Convolutional neural networks

Part 3

Image Segmentation
Assignment 2
CNN visualization
Adversarial examples (in 2 weeks)

Remaining lectures:

- 17.05 - Convnets part 3
- 24.05 - Embeddings
- 31.05 - Convnets part 4, Performance, Distributed learning
- 07.06 - Generative adversarial networks
- 14.06 - Projects’ presentations

Oral exam - probably 21.06 (maybe 22.06 as well)

Assignments 2 and 3:

- Assignment 2 - Image Segmentation
  - Most important facts released today
  - Final problem statement and dataset by 23.05 (probably 19.05)
  - Deadline 06.06

- Assignment 3 (most likely involving RNNs and/or embeddings)
  - Problem statement and dataset released by 30.05
  - Deadline 13.06

Projects

if you submitted a project plan, you should received feedback by today

Computational resources:

EC2, ICM, Codilime's machine, Henryk Michalewski's machine

Computer Vision Tasks

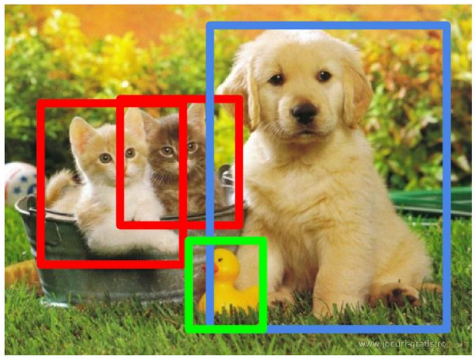
Classification

CAT

Classification + Localization

CAT

Object Detection



CAT, DOG, DUCK

Segmentation

CAT, DOG, DUCK

Single object

Multiple objects

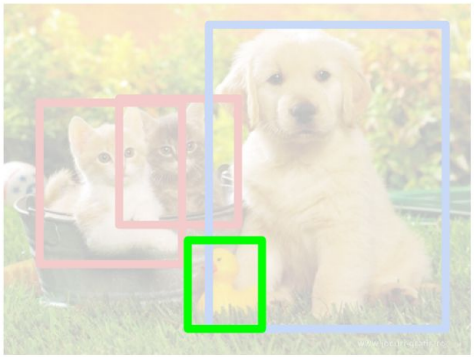
Slide taken from CS231n@stanford

Computer Vision Tasks

Classification

Classification + Localization

Object Detection



Segmentation

Today

Slide taken from CS231n@stanford

Semantic Segmentation

Label every pixel!

Don’t differentiate instances (cows)

Classic computer vision problem

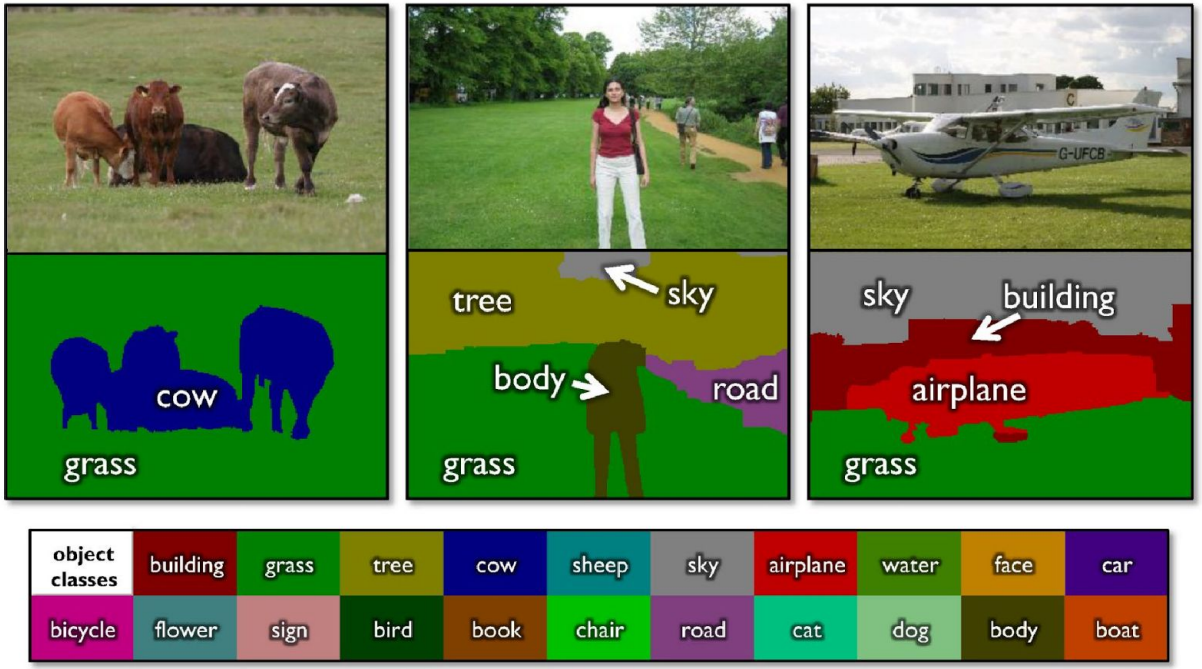


Figure credit: Shotton et al, “TextonBoost for Image Understanding: Multi-Class Object Recognition and Segmentation by Jointly Modeling Texture, Layout, and Context”, IJCV 2007

Slide taken from CS231n@stanford

Instance Segmentation

Detect instances, give category, label pixels

“simultaneous detection and segmentation” (SDS)

Lots of recent work (MS-COCO)

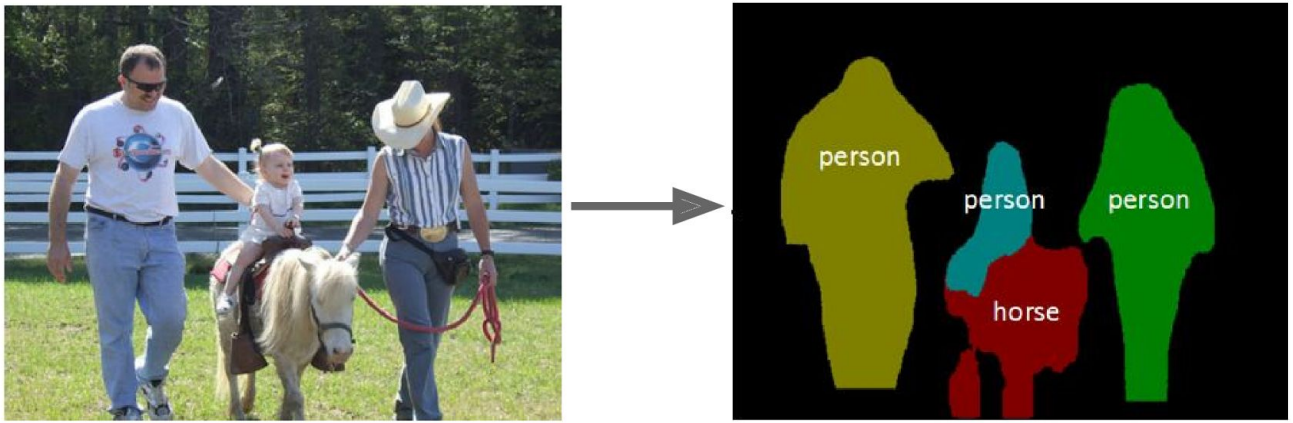
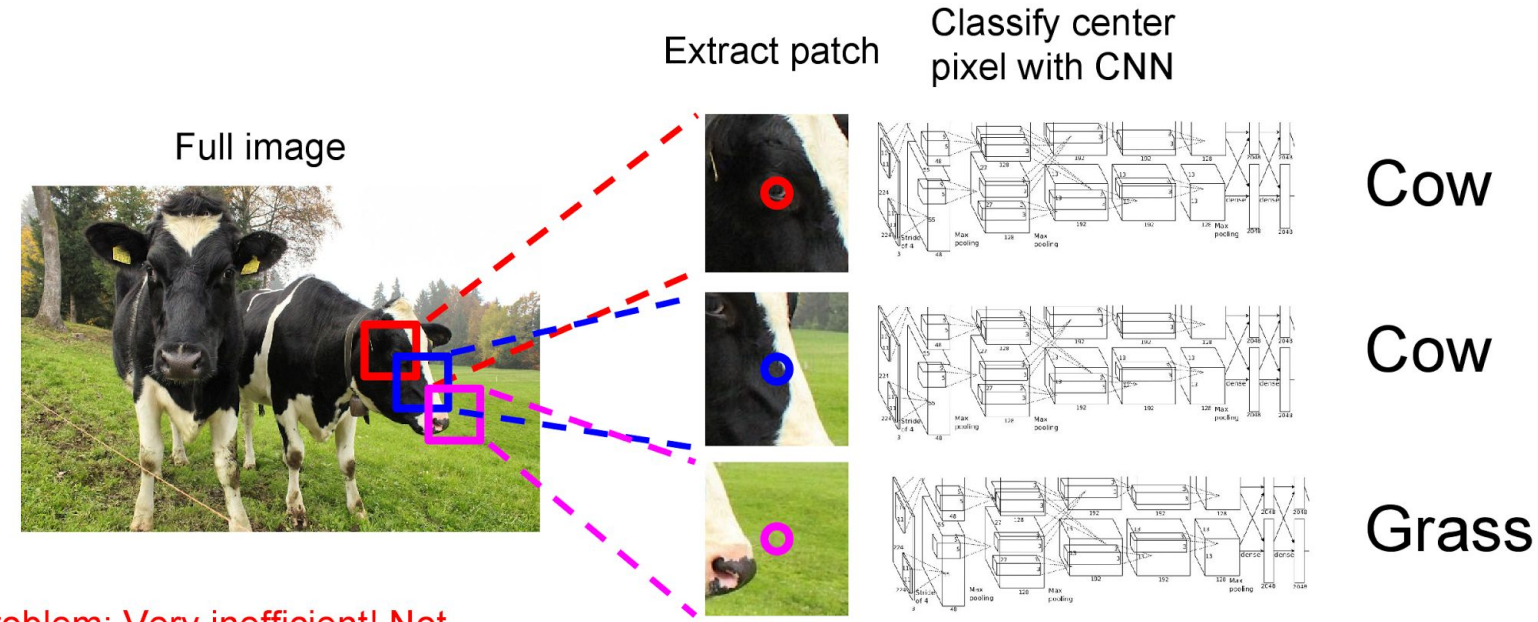


Figure credit: Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Slide taken from CS231n@stanford

Semantic segmentation

Semantic Segmentation Idea: Sliding Window

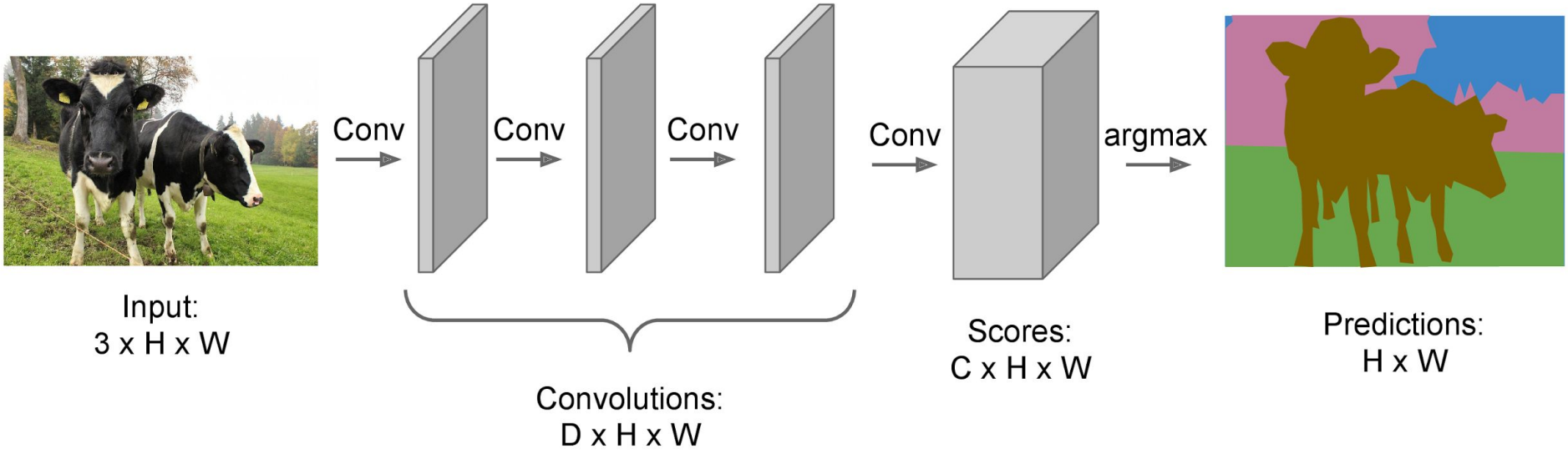


Problem: Very inefficient! Not reusing shared features between overlapping patches

Slide taken from CS231n@stanford

Semantic Segmentation Idea: Fully Convolutional

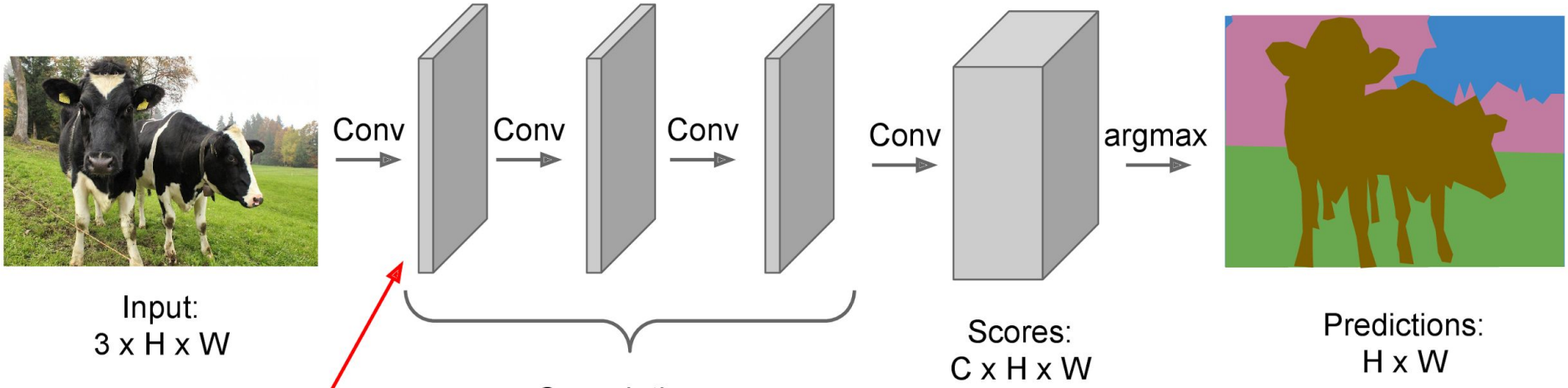
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Slide taken from CS231n@stanford

Semantic Segmentation Idea: Fully Convolutional

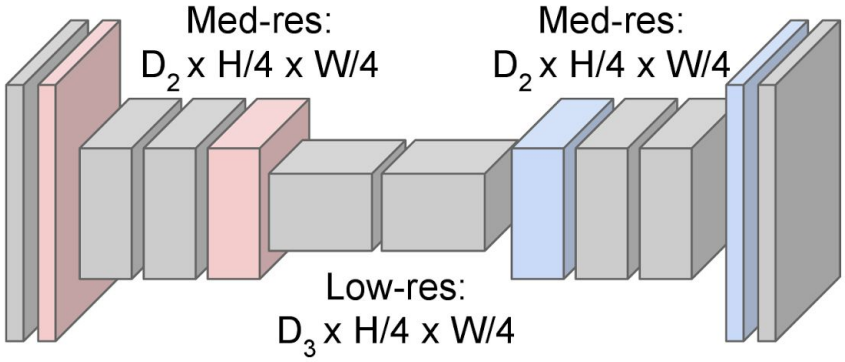
Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Slide taken from CS231n@stanford

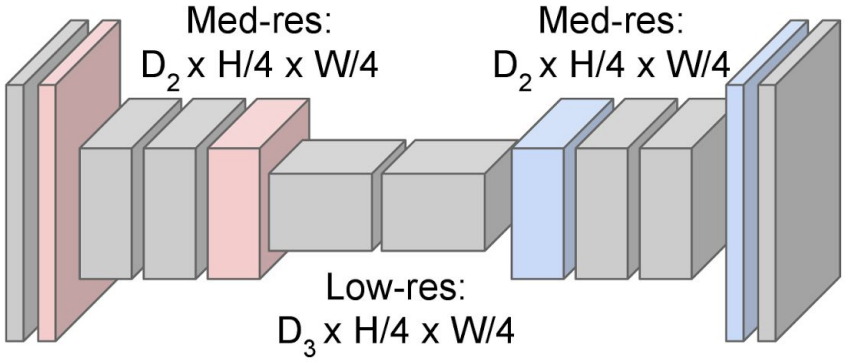
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

High-res:
D_1 x H/2 x W/2

Predictions:
H x W

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

Slide taken from CS231n@stanford

Most relevant papers for semantic segmentation in the last 5 years

- Multi-scale approach
  Farabet et al., *Learning Hierarchical Features for Scene Labeling,* **TPAMI** 2013

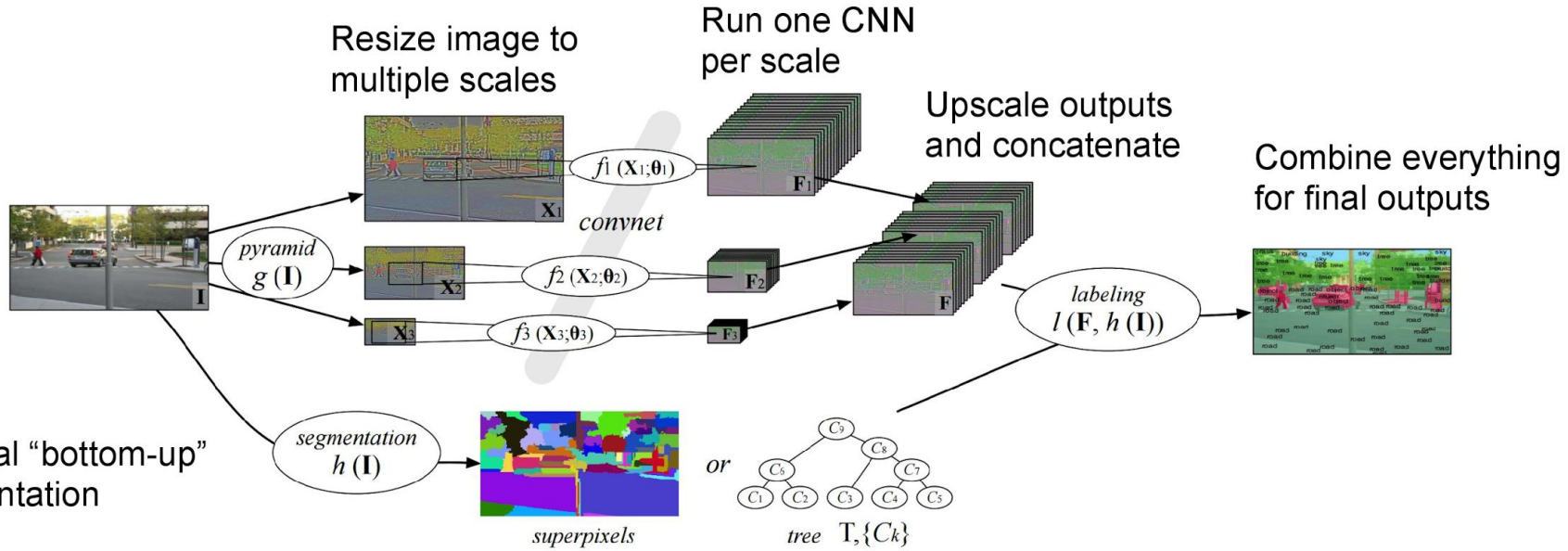
- Recursive refinement
  Pinheiro and Collobert, *Recurrent Convolutional Neural Networks for Scene Labeling*, **ICML** 2014

- Learnable upsampling and skip connections (U-net)
  Long, Shelhamer and Darrell, *Fully Convolutional Networks for Semantic Segmentation*, **CVPR** 2015

- Switch variables in unpooling
  Noh, Hong and Han, *Learning Deconvolution Network for Semantic Segmentation,* **ICCV** 2015

- Dilated convolutions (called atrous_conv2d in tensorflow)
  Yu and Koltun, *Multi-Scale Context Aggregation by Dilated Convolutions,* **ICLR** 2016

Semantic Segmentation: Multi-Scale



Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013

Slide taken from CS231n@stanford

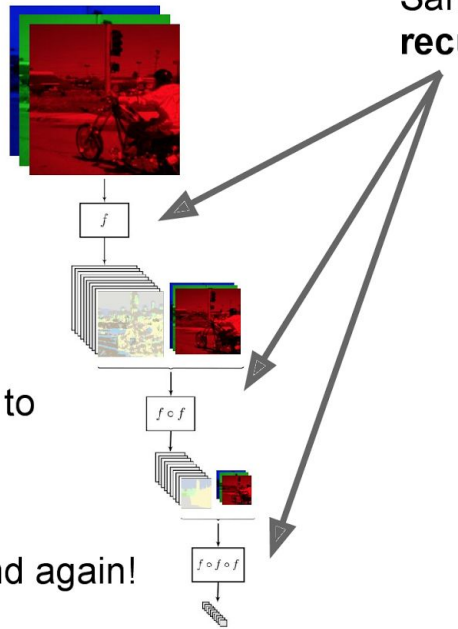
Semantic Segmentation: Refinement

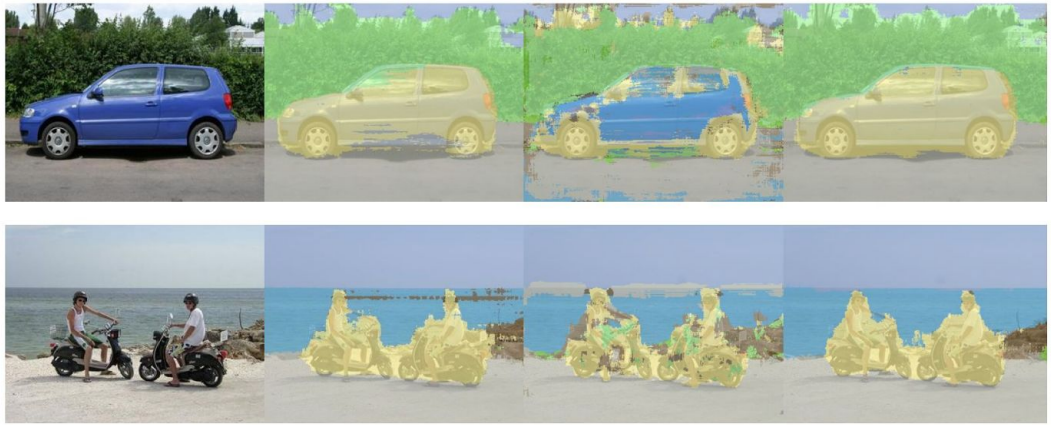
Same CNN weights:
recurrent convolutional network

Apply CNN once to get labels

Apply AGAIN to refine labels

And again!



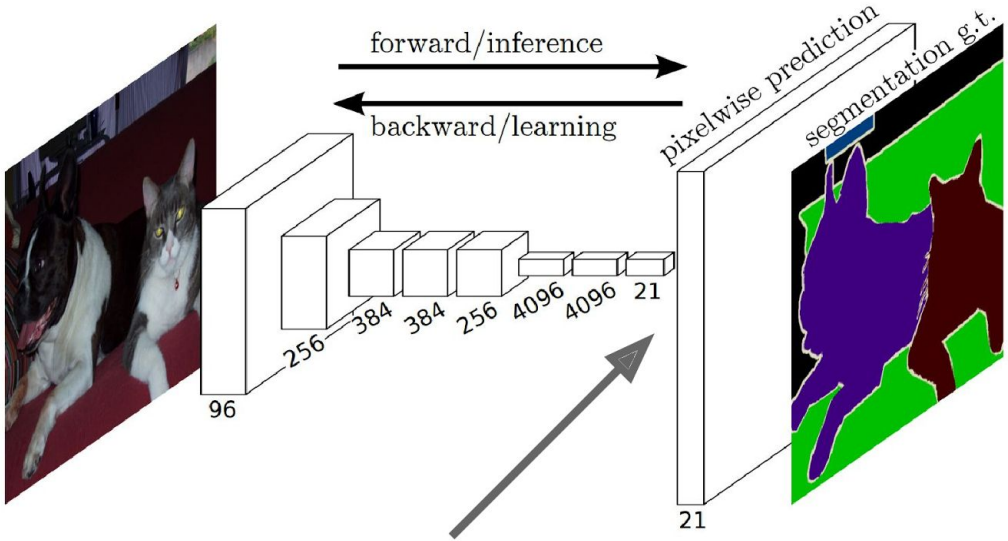


More iterations improve results

Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

Slide taken from CS231n@stanford

Semantic Segmentation: Upsampling

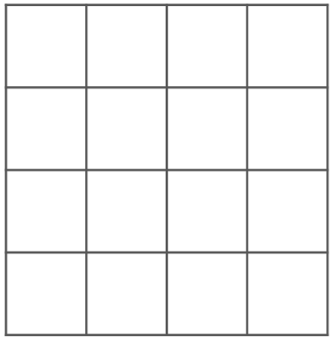


Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

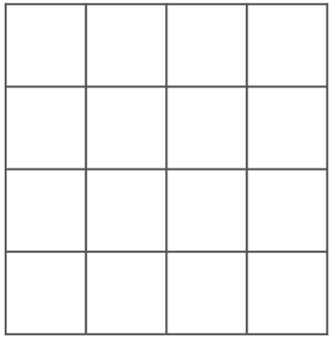
Slide taken from CS231n@stanford

Learnable upsampling - explanation

Typical 3 x 3 convolution, stride 1 pad 1

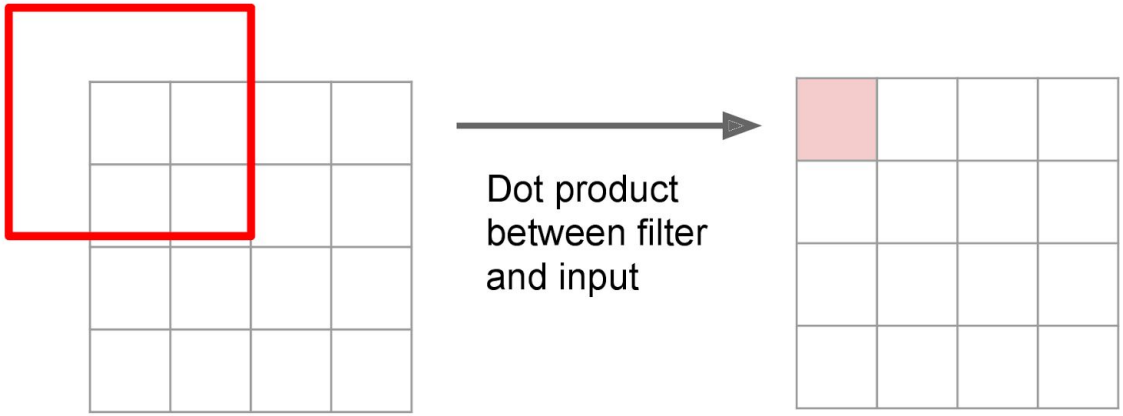


Input: 4 x 4

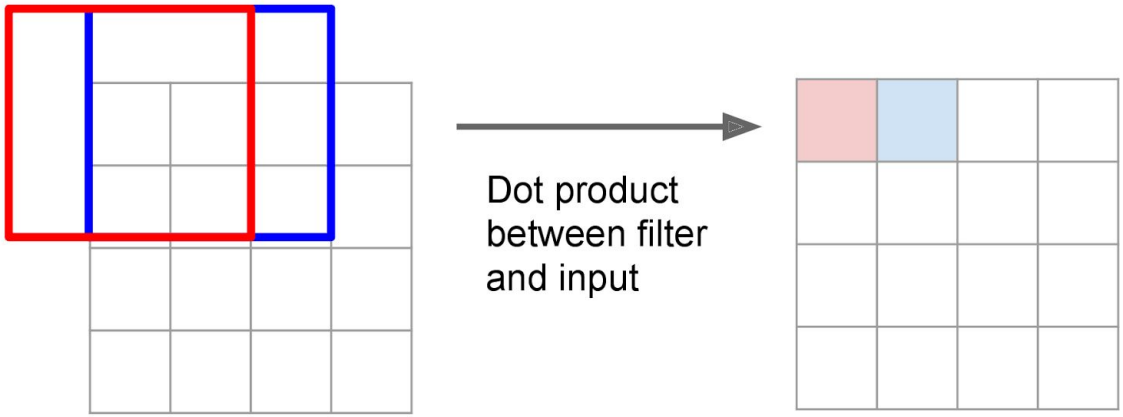


Output: 4 x 4

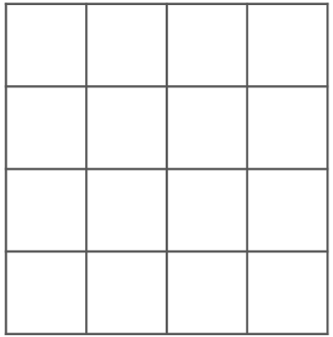
Typical 3 x 3 convolution, stride 1 pad 1



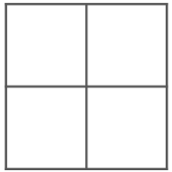
Typical 3 x 3 convolution, stride 1 pad 1



Typical 3 x 3 convolution, **stride 2** pad 1

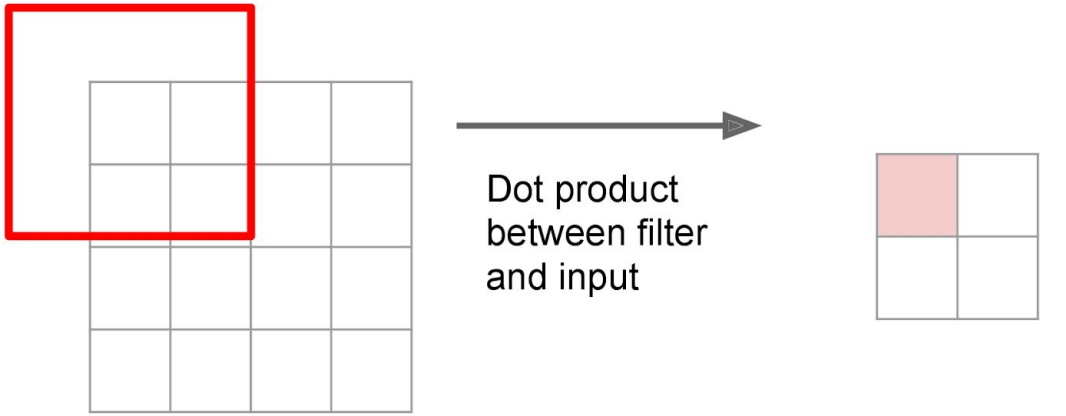


Input: 4 x 4



Output: 2 x 2

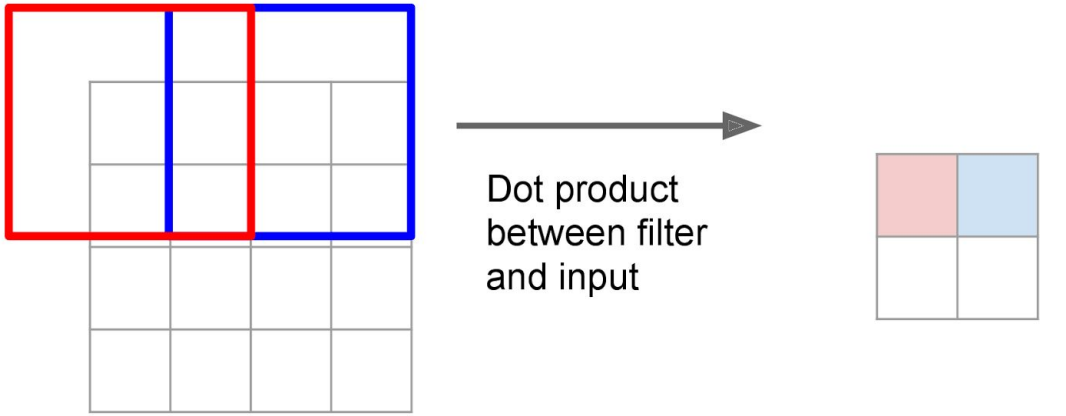
Typical 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Output: 2 x 2

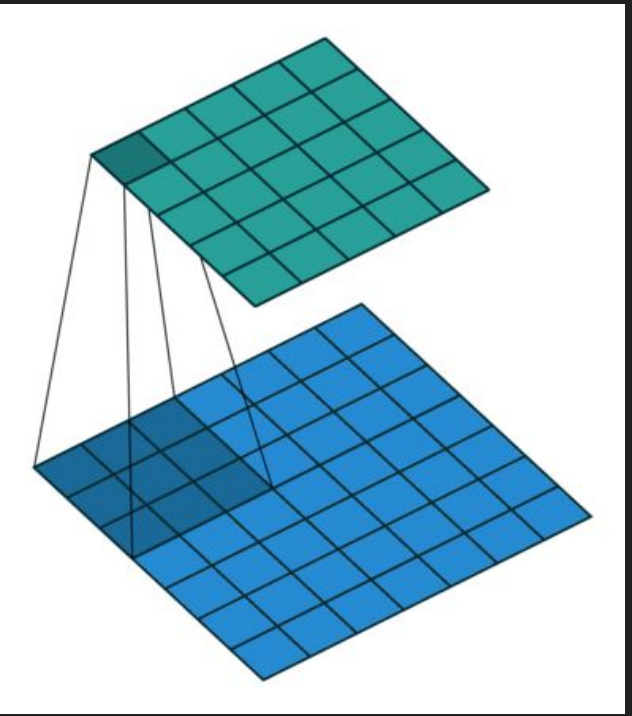
Typical 3 x 3 convolution, stride 2 pad 1

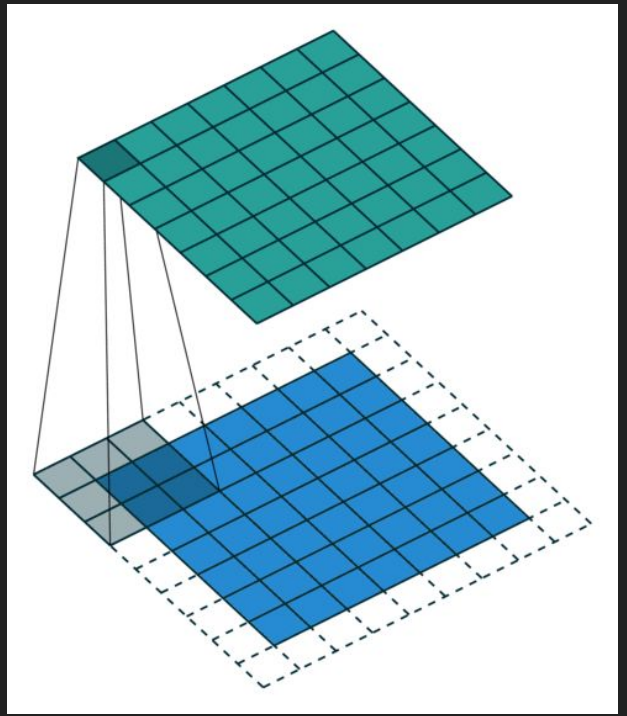


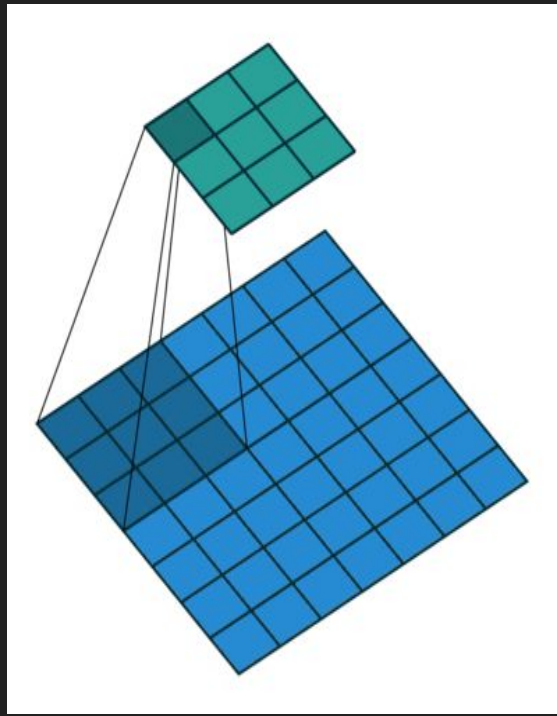
Input: 4 x 4

Output: 2 x 2

Regular convolution



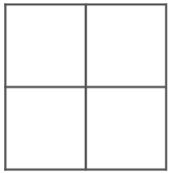




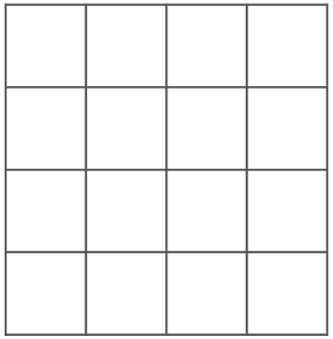
Images generated by using https://github.com/vdumoulin/conv_arithmetic

Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1



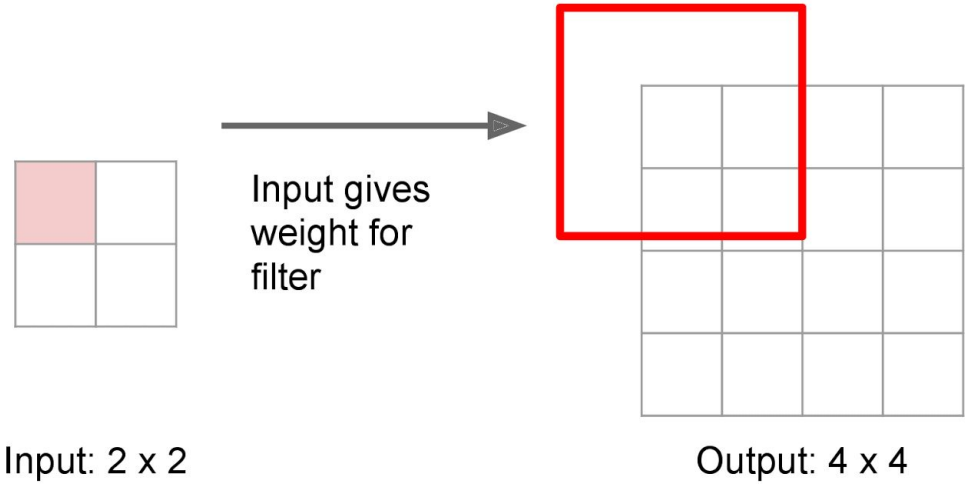
Input: 2 x 2



Output: 4 x 4

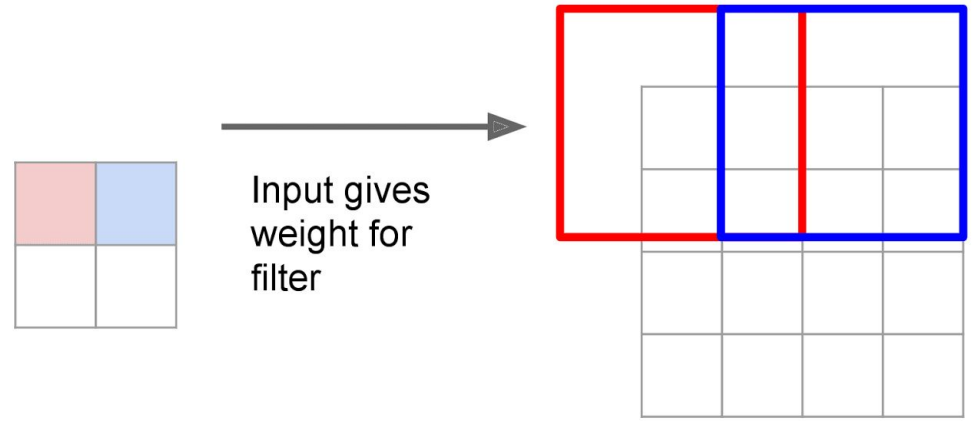
Learnable Upsampling: “Deconvolution”

3 x 3 “deconvolution”, stride 2 pad 1

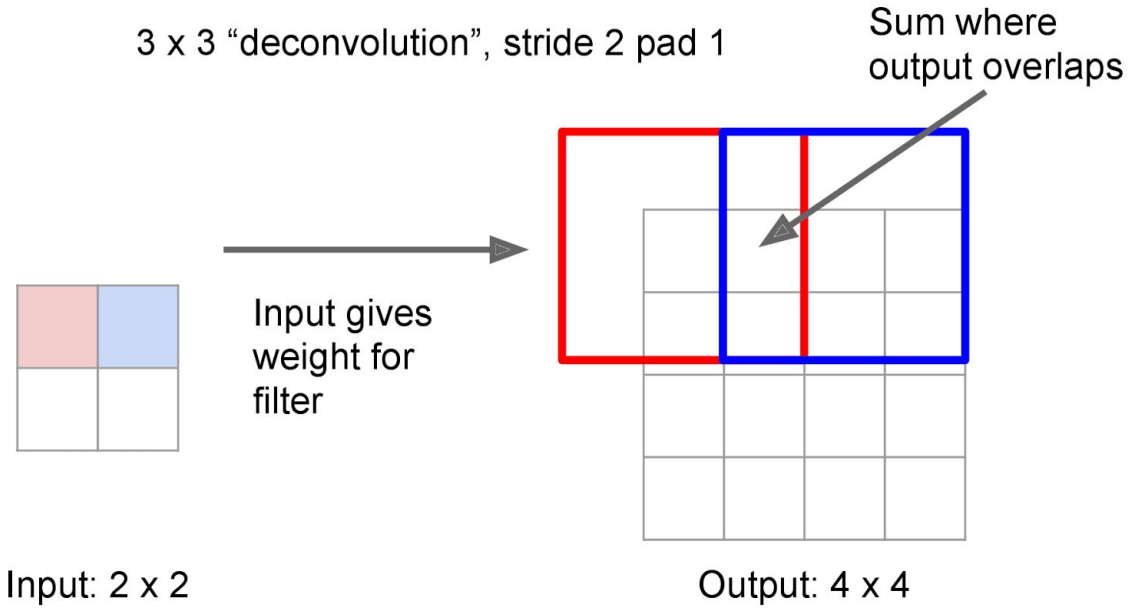


Learnable Upsampling: “Deconvolution”

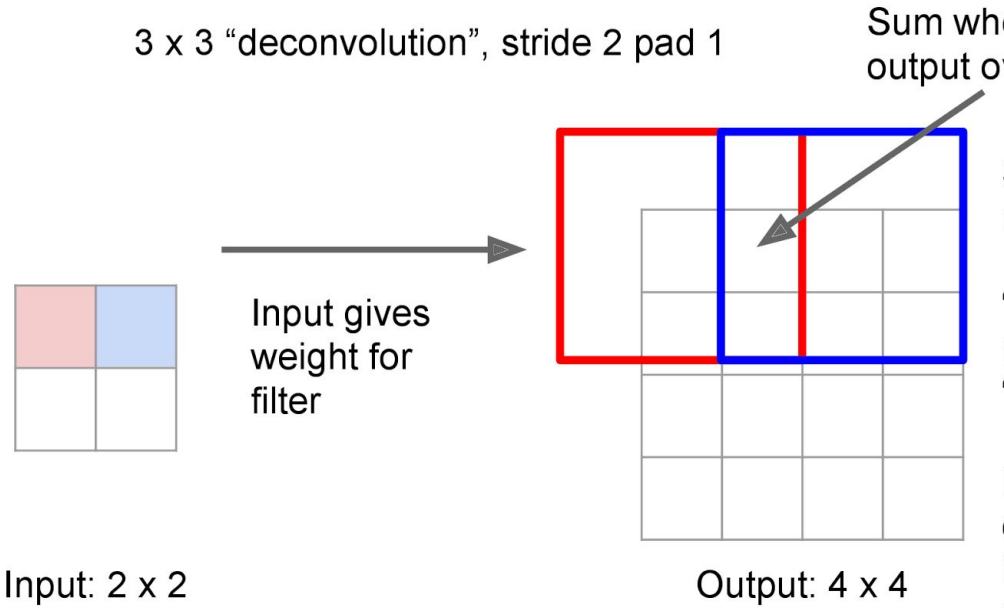
3 x 3 “deconvolution”, stride 2 pad 1



Learnable Upsampling: “Deconvolution”



Learnable Upsampling: “Deconvolution”



Same as backward pass for normal convolution!

“Deconvolution” is a bad name, already defined as “inverse of convolution”

Better names:
convolution transpose, backward strided convolution, 1/2 strided convolution, upconvolution

Slide taken from CS231n@stanford

Learnable upsampling, alternative view

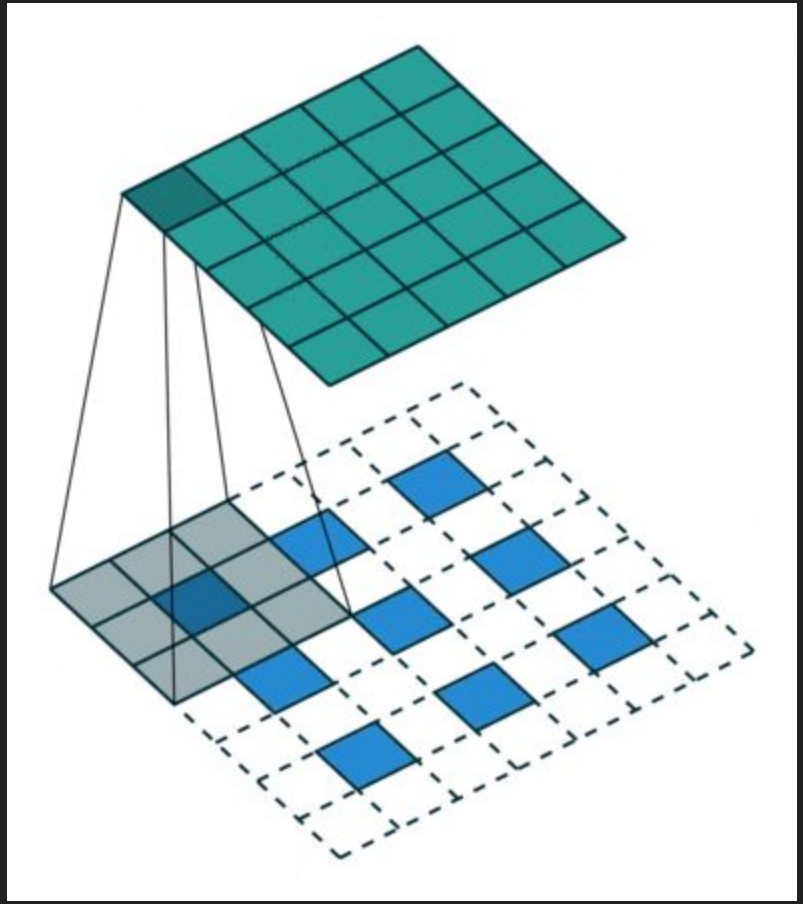
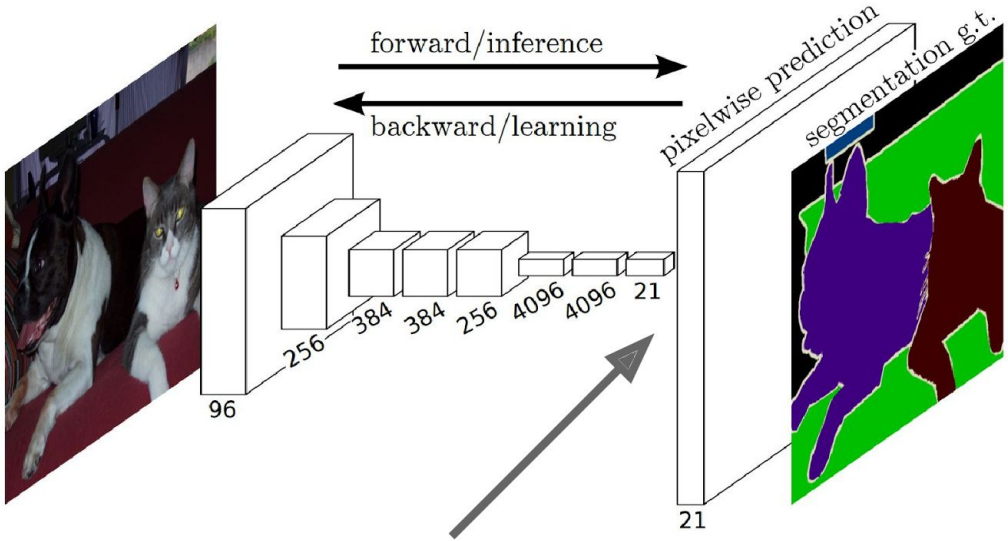


Image generated by using https://github.com/vdumoulin/conv_arithmetic

Learnable upsampling - explanation end

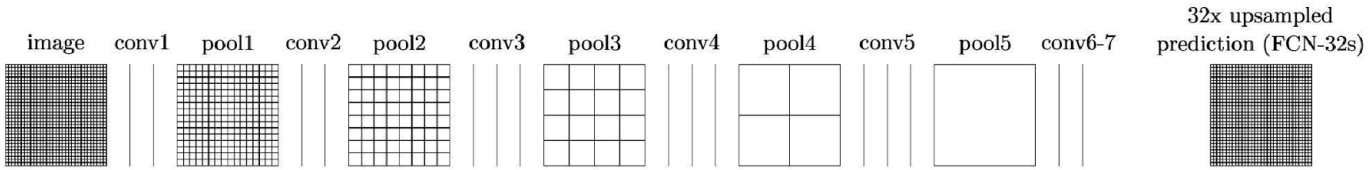
Semantic Segmentation: Upsampling



Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

Slide taken from CS231n@stanford

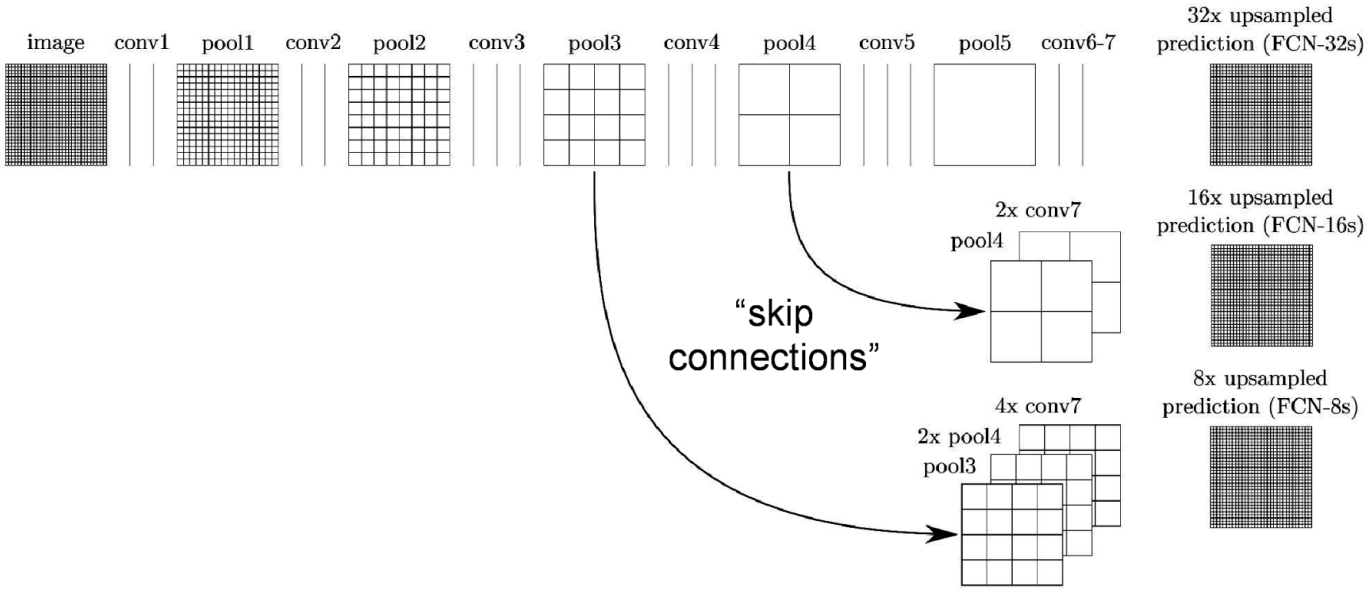
Semantic Segmentation: Upsampling



Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015

Slide taken from CS231n@stanford

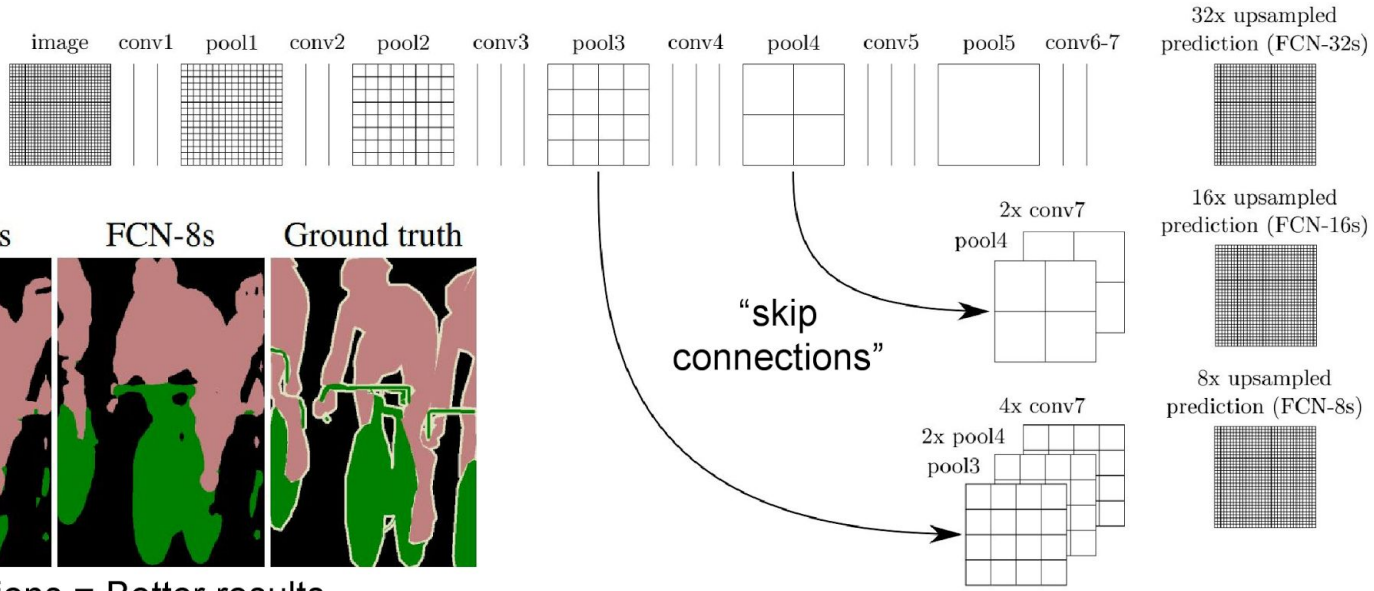
Semantic Segmentation: Upsampling



Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Slide taken from CS231n@stanford

Semantic Segmentation: Upsampling

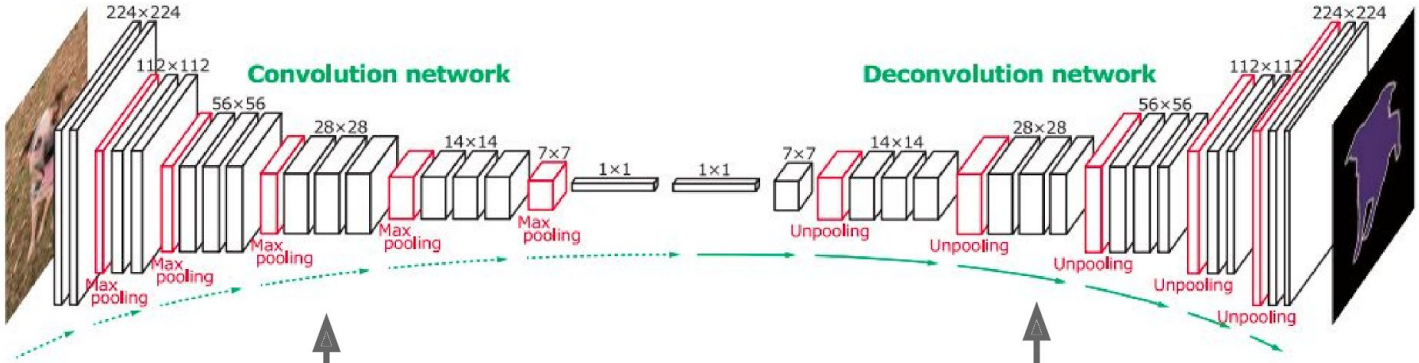


Skip connections = Better results

Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015

Slide taken from CS231n@stanford

Semantic Segmentation: Upsampling



Normal VGG

"Upside down" VGG

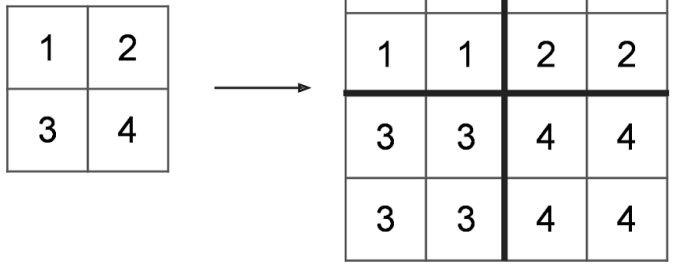
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

6 days of training on Titan X…

Slide taken from CS231n@stanford

In-Network upsampling: “Unpooling”

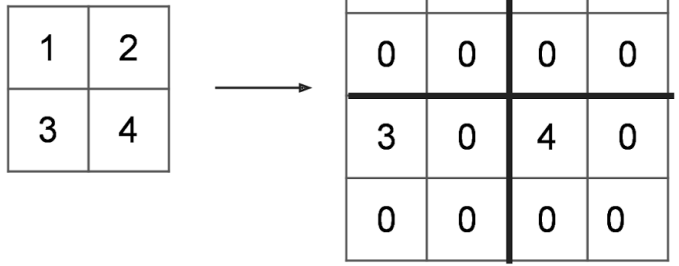
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



Input: 2 x 2

Output: 4 x 4

Slide taken from CS231n@stanford

In-Network upsampling: “Max Unpooling”

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

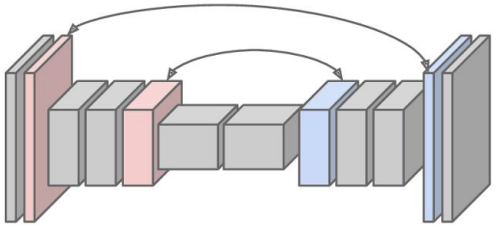
1	2
3	4

Input: 2 x 2

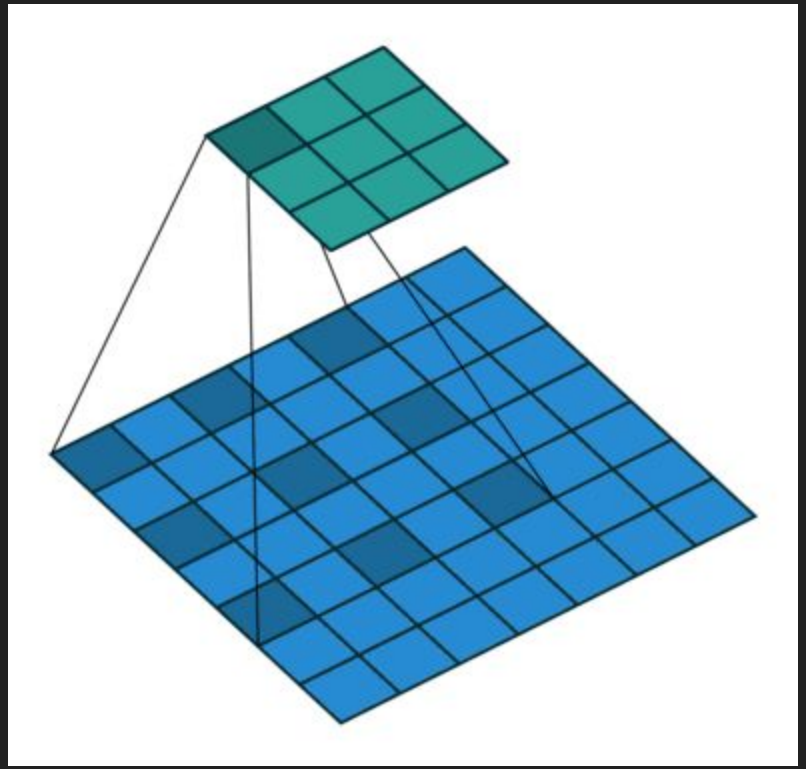
0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



Dilated convolutions



Yu and Koltun, *Multi-Scale Context Aggregation by Dilated Convolutions*, **ICLR 2016**

A way of increasing the receptive field without decreasing the image resolution, while keeping the number of parameters small.

Implemented in TensorFlow (atrous_conv2d).

Image generated by using https://github.com/vdumoulin/conv_arithmetic

Dilated convolutions

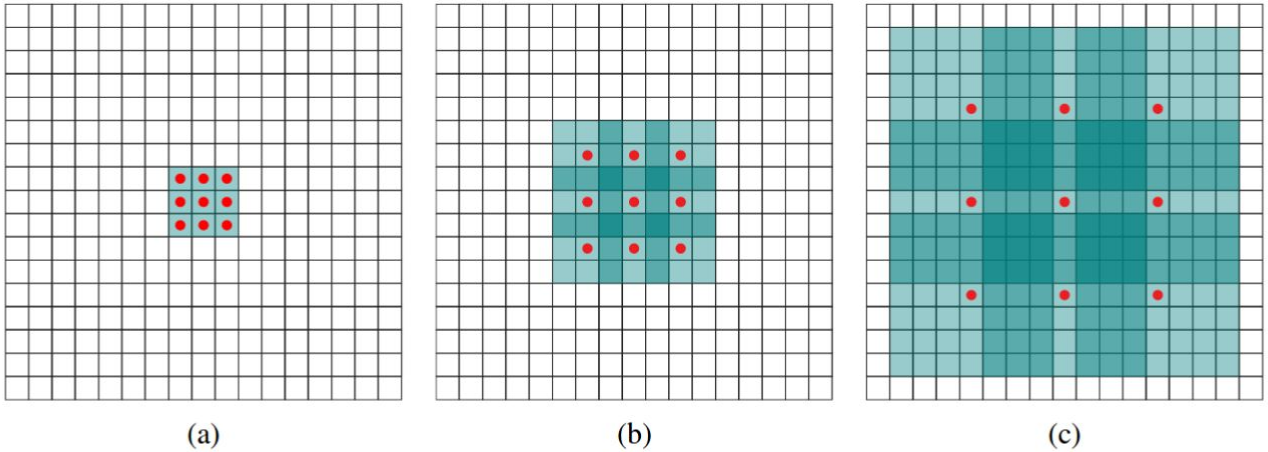


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3. (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7. (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15. The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

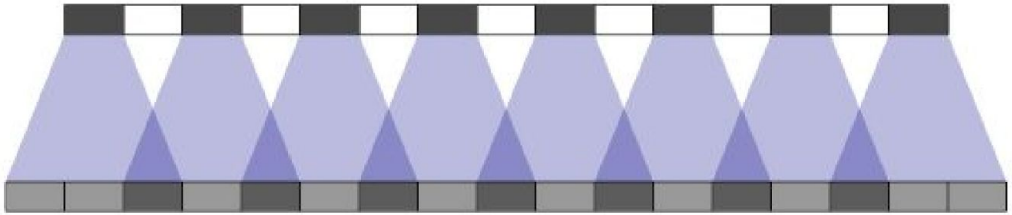
Yu and Koltun, *Multi-Scale Context Aggregation by Dilated Convolutions*, **ICLR 2016**

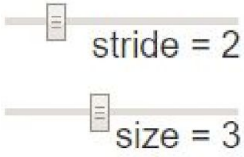
The checkerboard effect of conv transpose

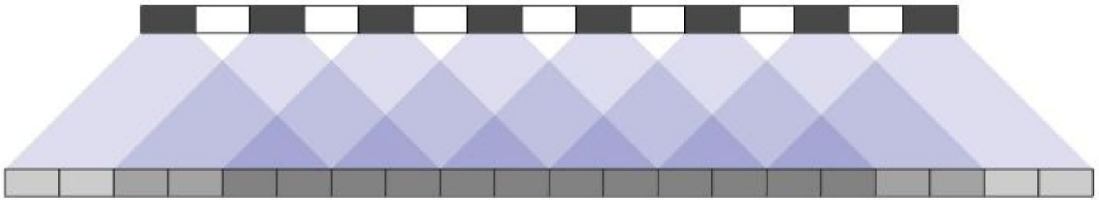
When we look very closely at images generated by neural networks, we often see a strange checkerboard pattern of artifacts. It's more obvious in some cases than others, but a large fraction of recent models exhibit this behavior.

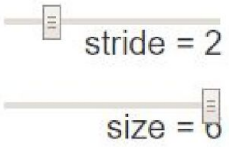
http://distill.pub/2016/deconv-checkerboard/

Warning: Checkerboard effect when kernel size is not divisible by the stride

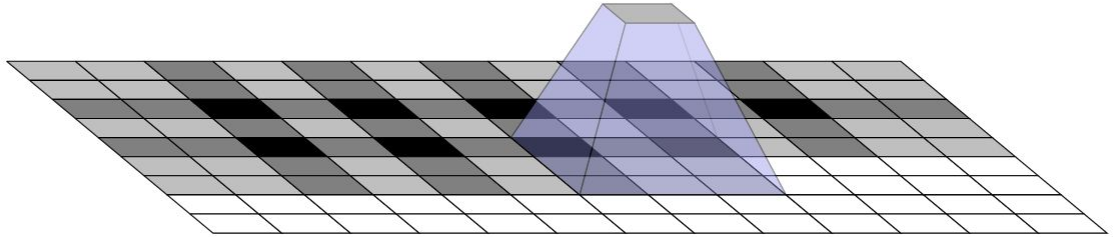








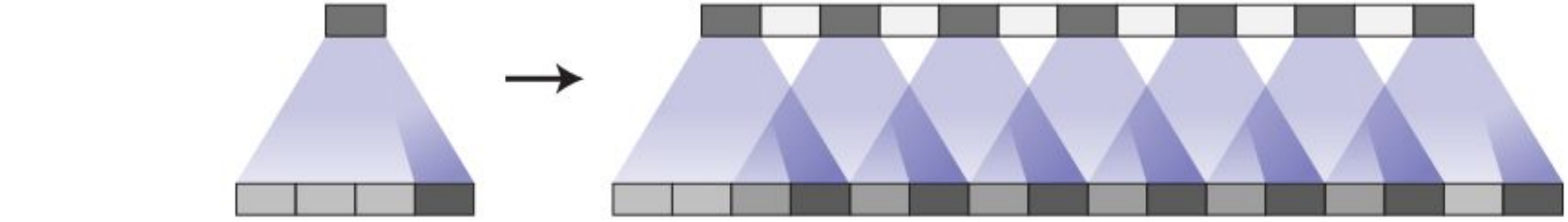
http://distill.pub/2016/deconv-checkerboard/



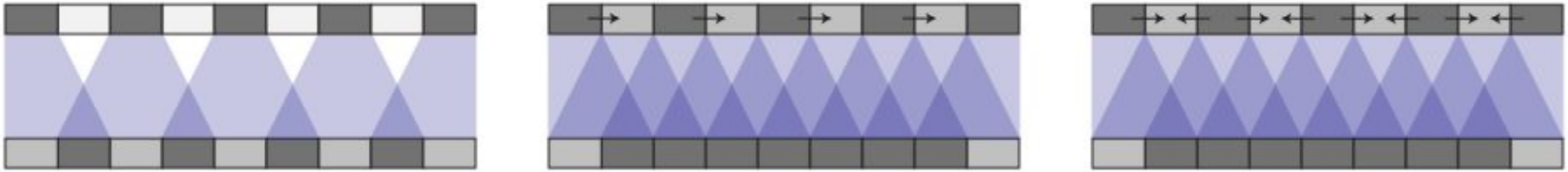
In fact, the uneven overlap tends to be more extreme in two dimensions! Because the two patterns are multiplied together, the unevenness gets squared. For example, in one dimension, a stride 2, size 3 deconvolution has some outputs with twice the number of inputs as others, but in two dimensions this becomes a factor of four.

http://distill.pub/2016/deconv-checkerboard/

Completely avoiding artifacts is still a significant restriction on filters, and in practice the artifacts are still present in these models, although they seem milder.

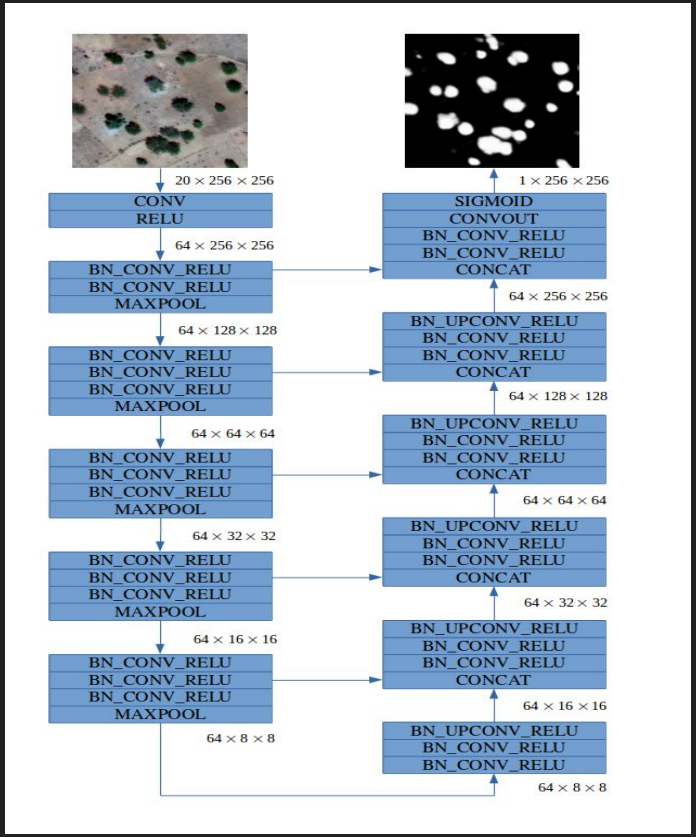


Instead of strided convolution, it might be a better idea to first do upsampling (nearest neighbor or bilinear), and then do a convolution without changing the size.



http://distill.pub/2016/deconv-checkerboard/

Semantic segmentation, take-home-message



- Use U-net as a default starting point for semantic segmentation (sample architecture on the left)
- Fully convolutional solution (no region proposals)
- For conv transpose use kernel size divisible by stride, or (possibly better option) upsample by nearest neighbour and then do regular convolution
- Dilated convolution often gives improved results, but at the cost of longer training and more space consumption.

https://deepsense.io/deep-learning-for-satellite-imagery-via-image-segmentation/

Instance segmentation

Instance Segmentation

Detect instances, give category, label pixels

“simultaneous detection and segmentation” (SDS)

Lots of recent work (MS-COCO)

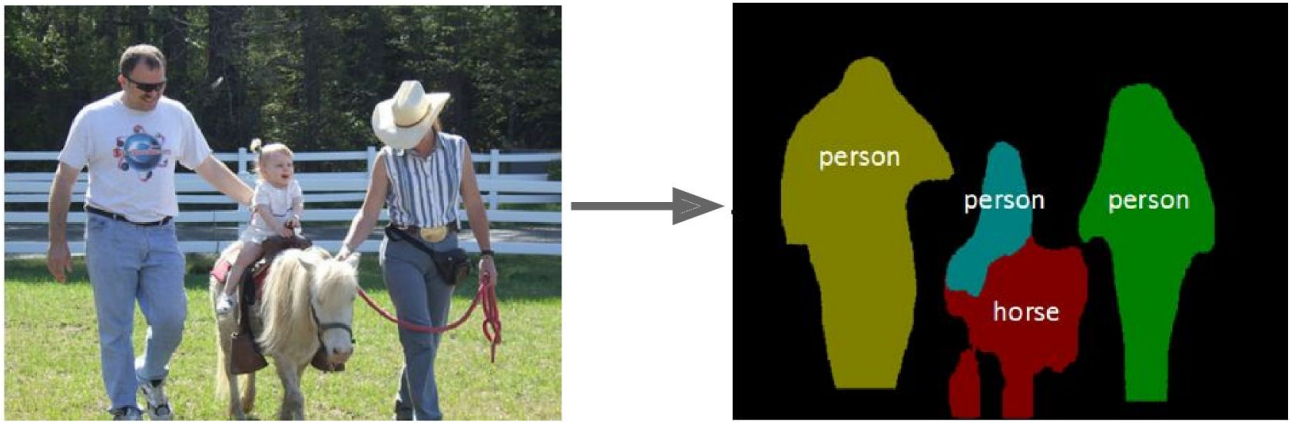
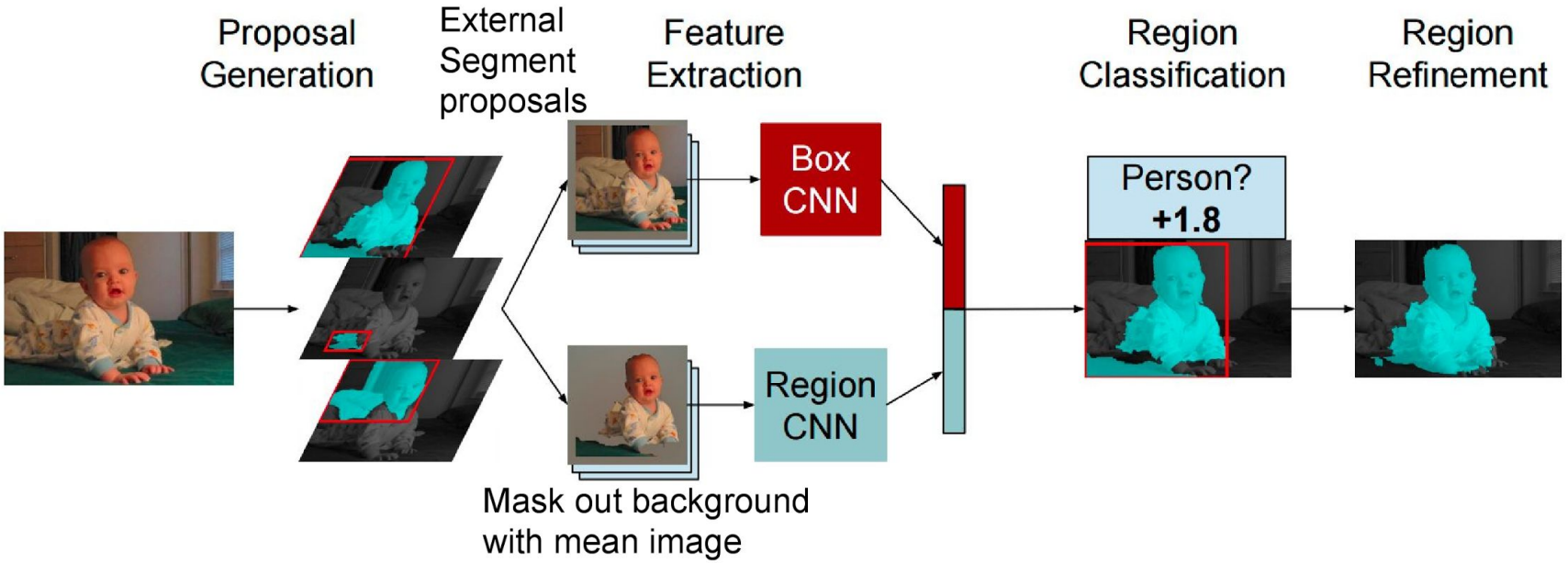


Figure credit: Dai et al, “Instance-aware Semantic Segmentation via Multi-task Network Cascades”, arXiv 2015

Instance Segmentation

Similar to R-CNN, but with segments



Hariharan et al, “Simultaneous Detection and Segmentation”, ECCV 2014

Slide taken from CS231n@stanford

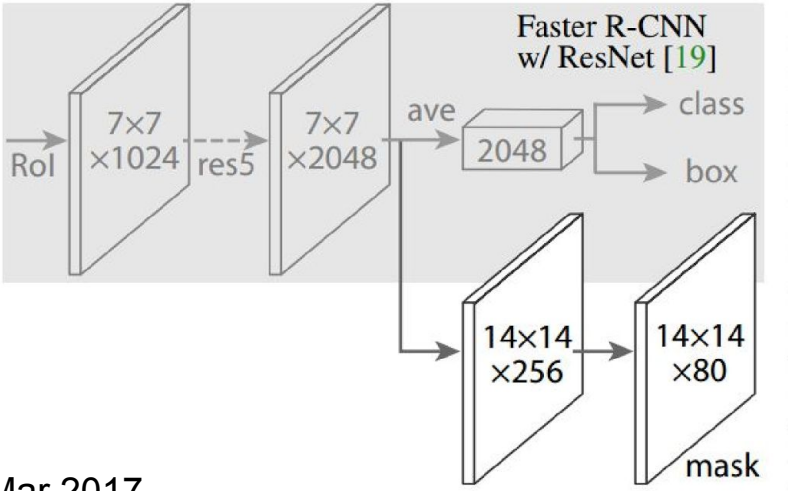
Instance segmentation, fast forward to 2017:

- Resnets
- Tricks from Fast and Faster RCNN
- ROI Allign instead of ROI Pooling
- State of the art: He et al. *Mask R-CNN*, arxiv Mar 2017

Mask R-CNN

- Classification & box detection losses are identical to those in Faster R-CNN
- Addition of a new loss term for mask prediction:

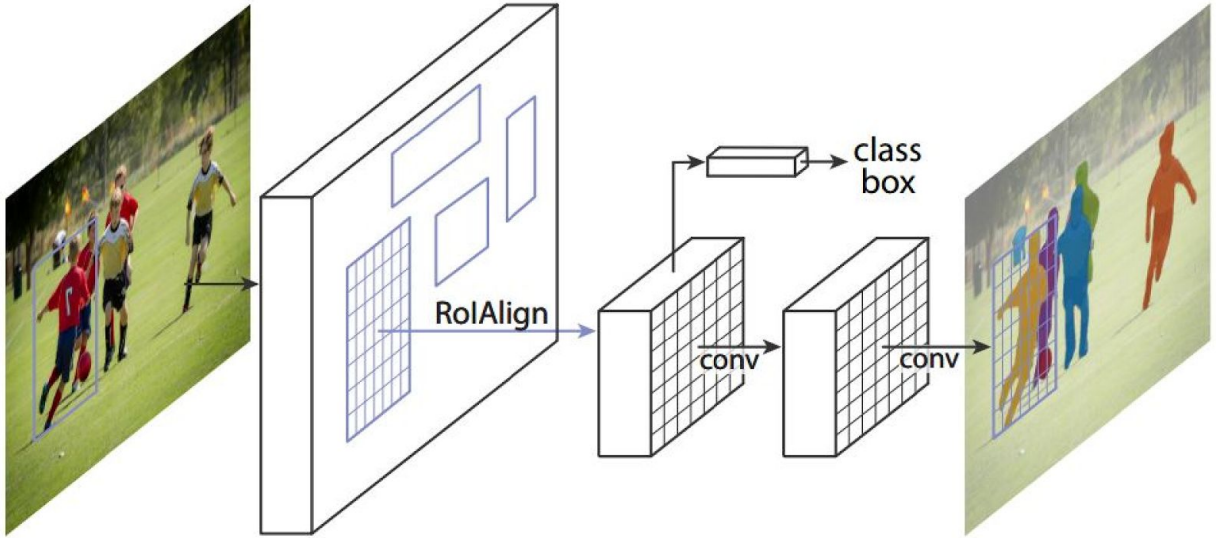
The network outputs a *K x m x m* volume for mask prediction, where *K* is the number of categories and *m* is the size of the mask (square)



He et al. Mask R-CNN, arxiv Mar 2017

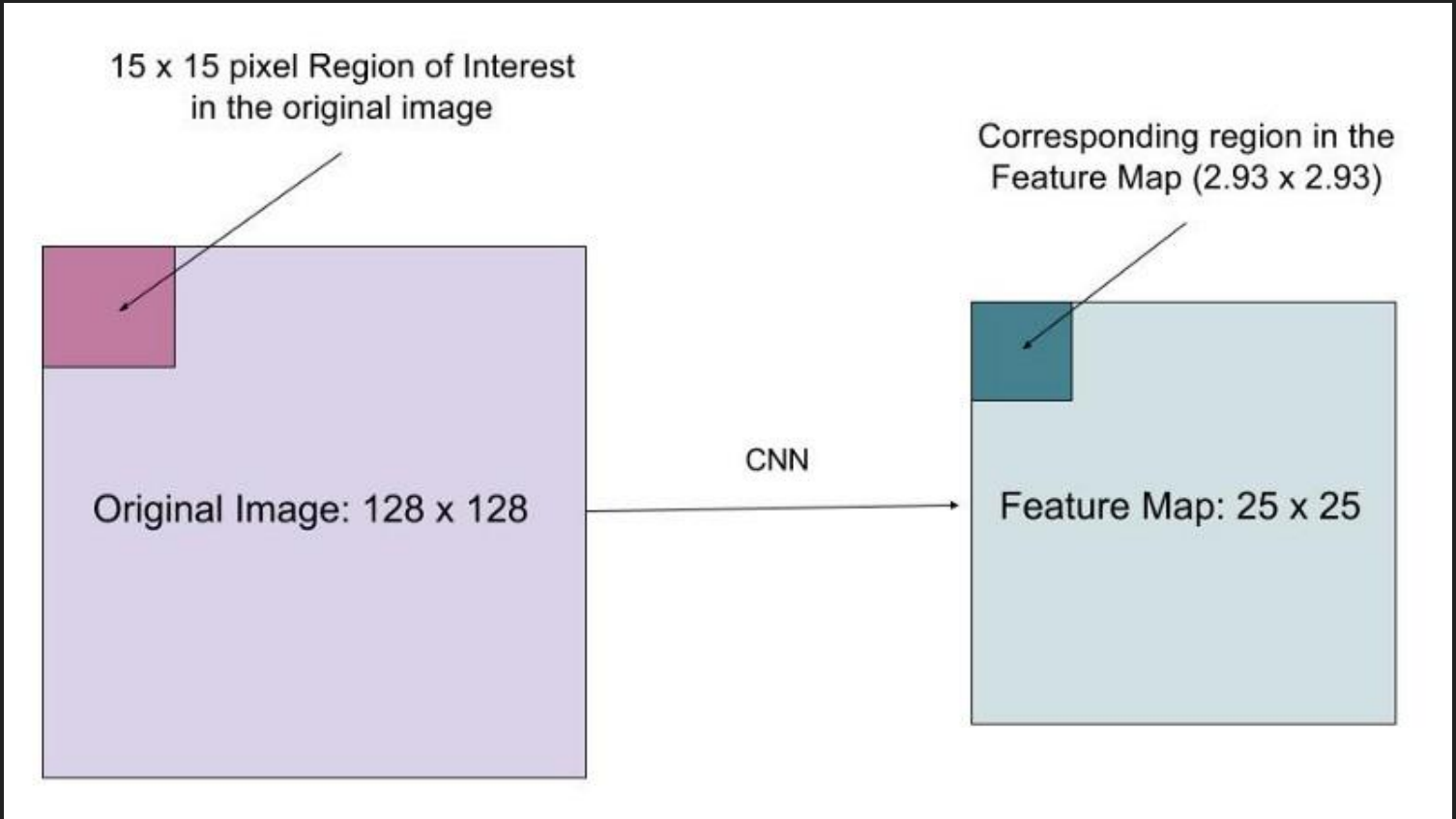
Proposal-based Instance Segmentation: Mask R-CNN

Faster R-CNN for Pixel Level Segmentation as a **parallel prediction of masks and class labels**

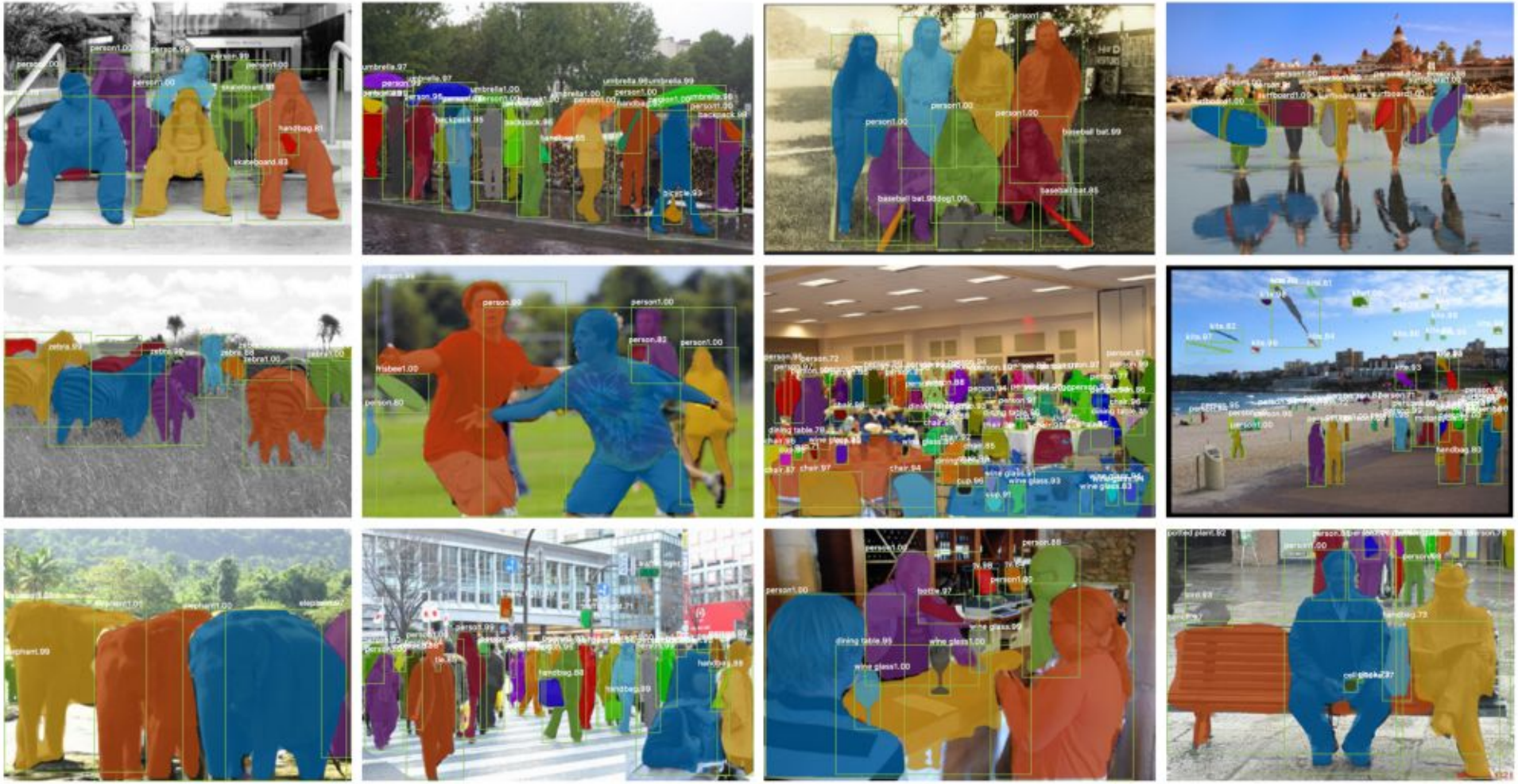


He et al. Mask R-CNN, arxiv Mar 2017

ROI Allign



https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

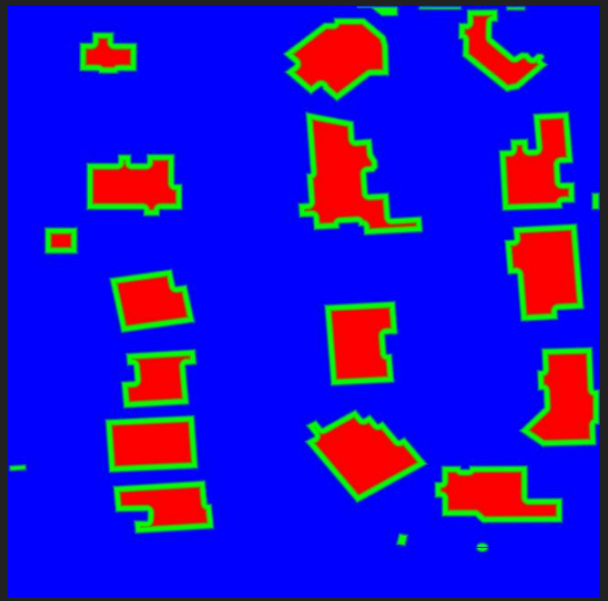


He et al. Mask R-CNN, arxiv Mar 2017

Segmentation Overview

- Semantic segmentation
  - Classify all pixels
  - Fully convolutional models, downsample then upsample
  - Learnable upsampling: fractionally strided convolution
  - Skip connections
- Instance Segmentation
  - Detect instance, generate mask
  - Similar pipelines to object detection

Assignment 2



- Classify each pixel into one of three categories.
- Optimize cross entropy loss (return probabilities).
- Implement some version of U-net (no copy-pasting from online resources).
- Data /data/spacenet2 @ICM (tital-lab-gw).

Assignment 2

Requested features:

- Split the data into train & validation set (randomly).
- Batch prepared on CPU, training on GPU (sample code for batch preparation will be provided). You can use feed_dict approach, even though it is suboptimal from the performance perspective.
- Implement data augmentation (rotation, cropping, flipping, scaling).
- When calculating predictions for the validation set, average a few augmented versions of an image, store the averaged predictions.
- Add context-module made of dilated convolutions (maybe).

Understanding convnets

opening the black-box

Understanding ConvNets

- Visualize patches that maximally activate neurons
- Visualize the weights
- Visualize the representation space (e.g. with t-SNE)
- Occlusion experiments
- Human experiment comparisons
- Deconv approaches (single backward pass)
- Optimization over image approaches (optimization)

Visualize patches that maximally activate neurons

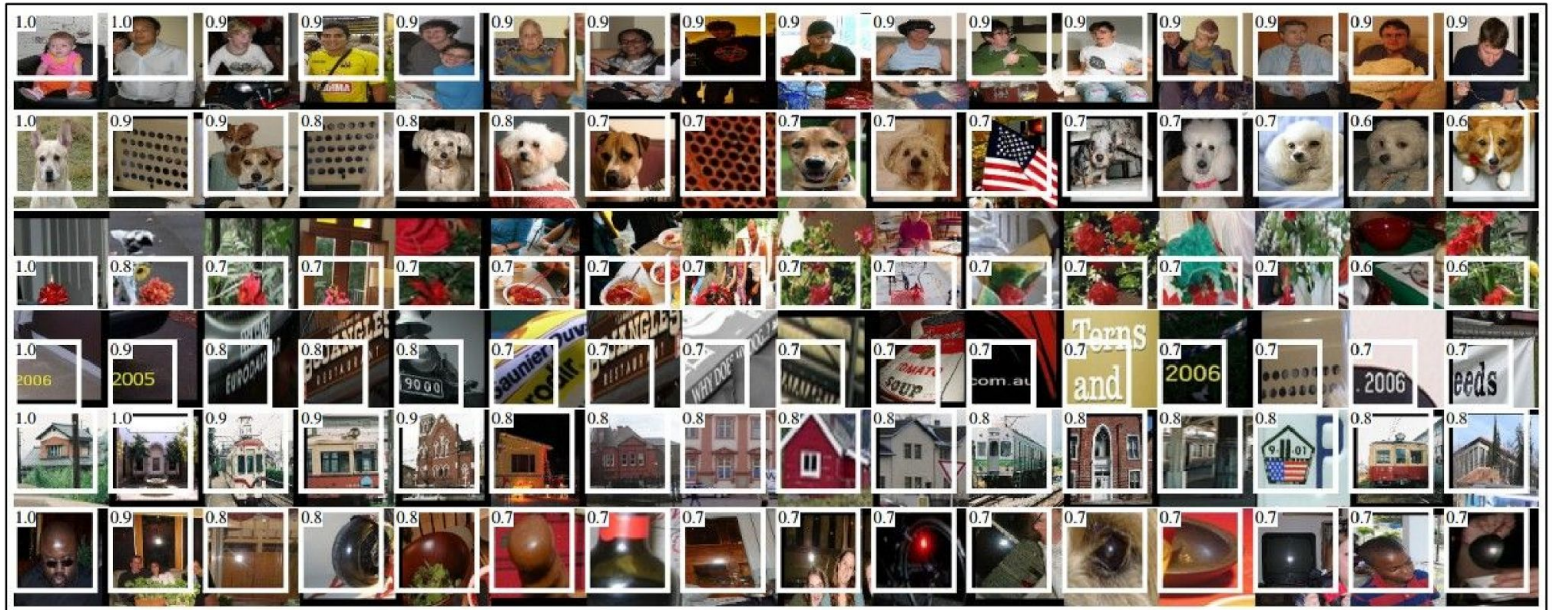
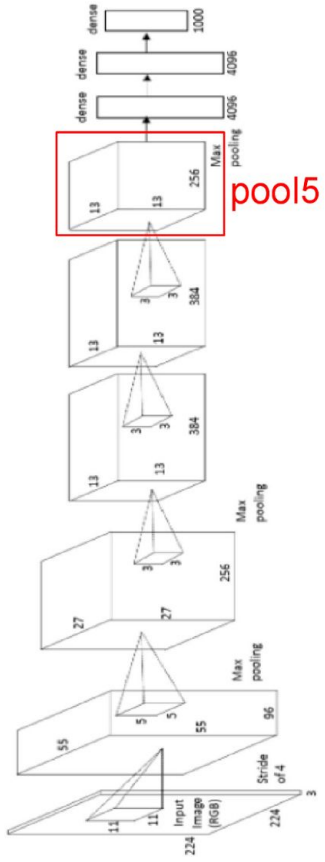


Figure 4: Top regions for six pool$_5$ units. Receptive fields and activation values are drawn in white. Some units are aligned to concepts, such as people (row 1) or text (4). Other units capture texture and material properties, such as dot arrays (2) and specular reflections (6).

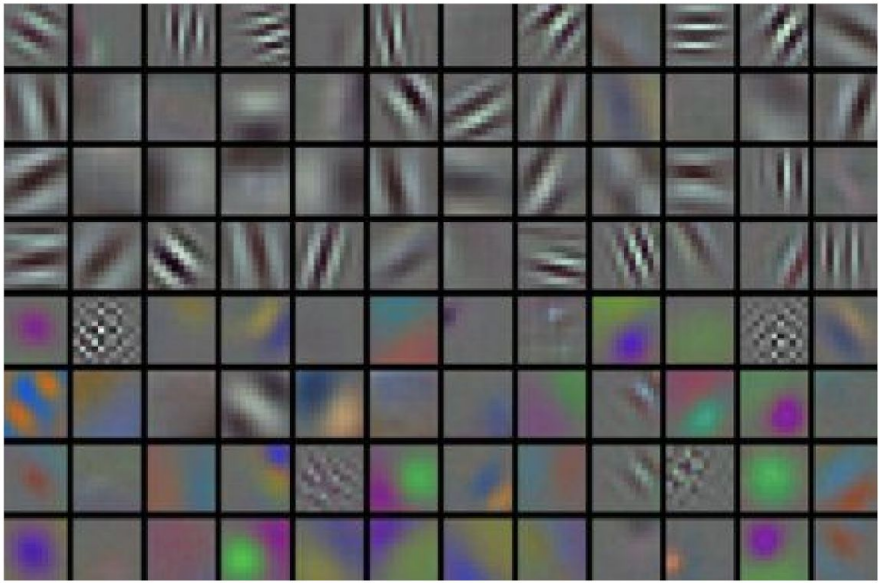
one-stream AlexNet

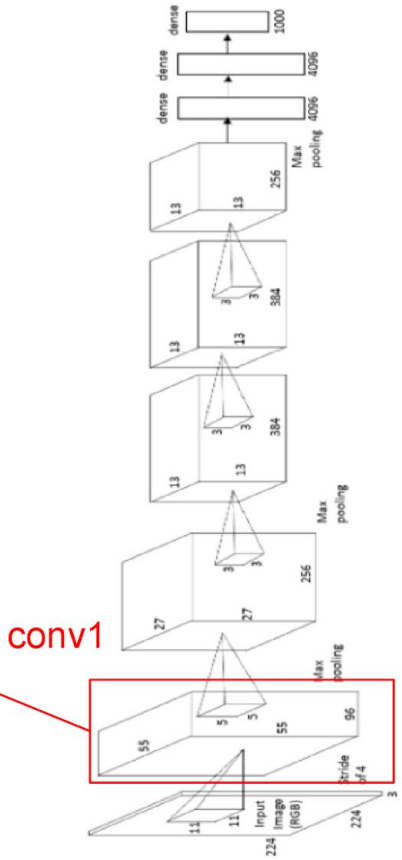


Rich feature hierarchies for accurate object detection and semantic segmentation
[Girshick, Donahue, Darrell, Malik]

Slide taken from CS231n@stanford

Visualize the filters/kernels (raw weights)





only interpretable on the first layer :(

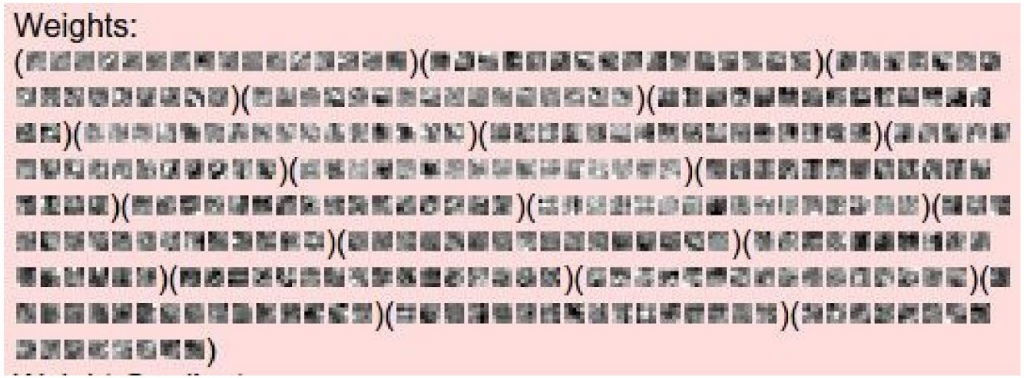
Slide taken from CS231n@stanford

Visualize the filters/kernels (raw weights)

you can still do it for higher layers, it's just not that interesting

(these are taken from ConvNetJS CIFAR-10 demo)

layer 1 weights

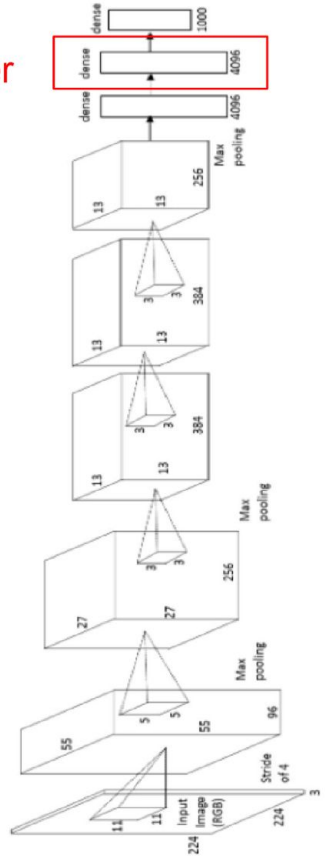


layer 2 weights

layer 3 weights

Visualizing the representation

fc7 layer



4096-dimensional “code” for an image
(layer immediately before the classifier)

can collect the code for many images

Slide taken from CS231n@stanford

Visualizing the representation

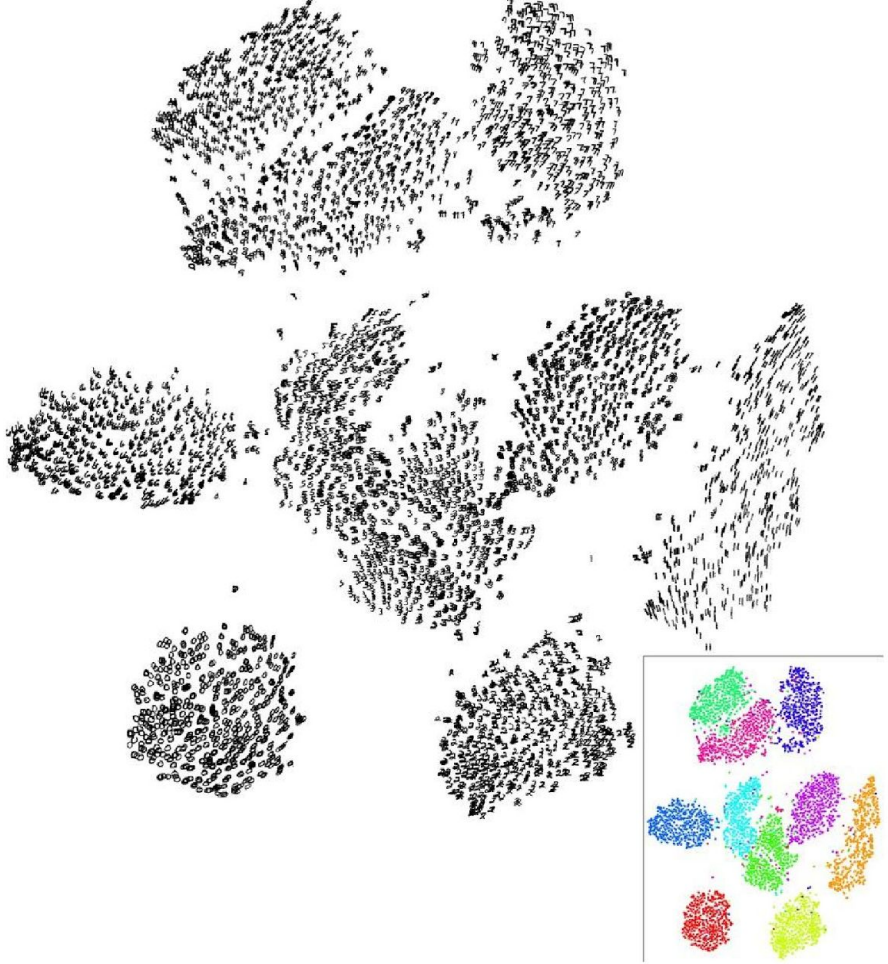
t-SNE visualization

[van der Maaten & Hinton]

Embed high-dimensional points so that locally, pairwise distances are conserved

i.e. similar things end up in similar places. dissimilar things end up wherever

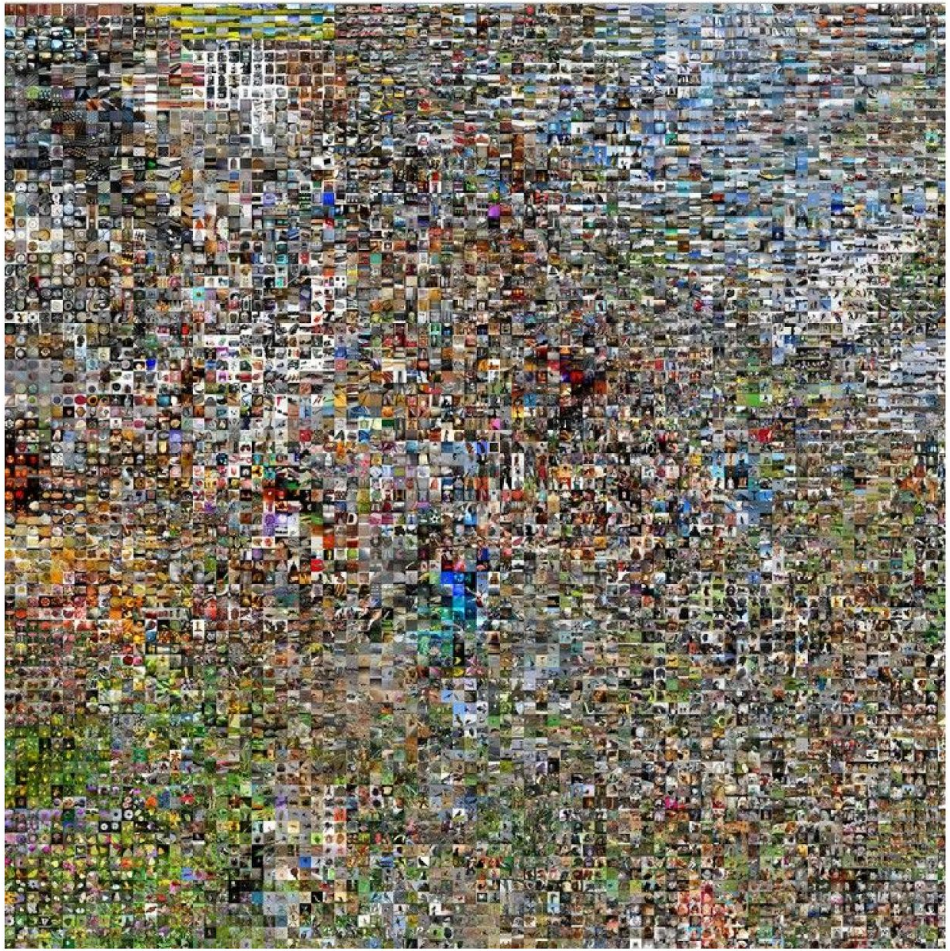
Right: Example embedding of MNIST digits (0-9) in 2D



t-SNE visualization:

two images are placed nearby if their CNN codes are close. See more:

http://cs.stanford.edu/people/karpathy/cnnembed



Occlusion experiments

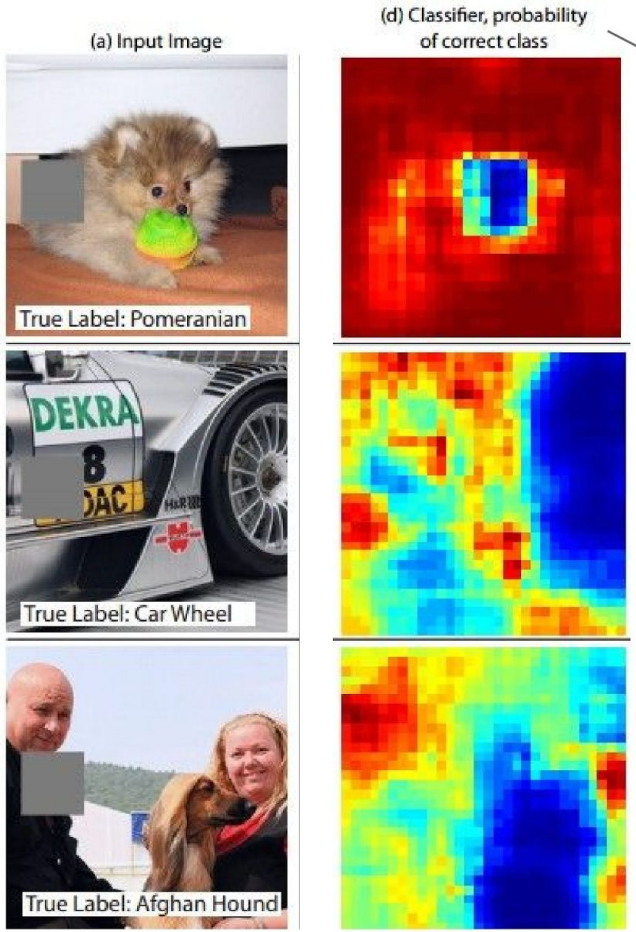
[Zeiler & Fergus 2013]

(as a function of the position of the square of zeros in the original image)

Slide taken from CS231n@stanford

Occlusion experiments

[Zeiler & Fergus 2013]



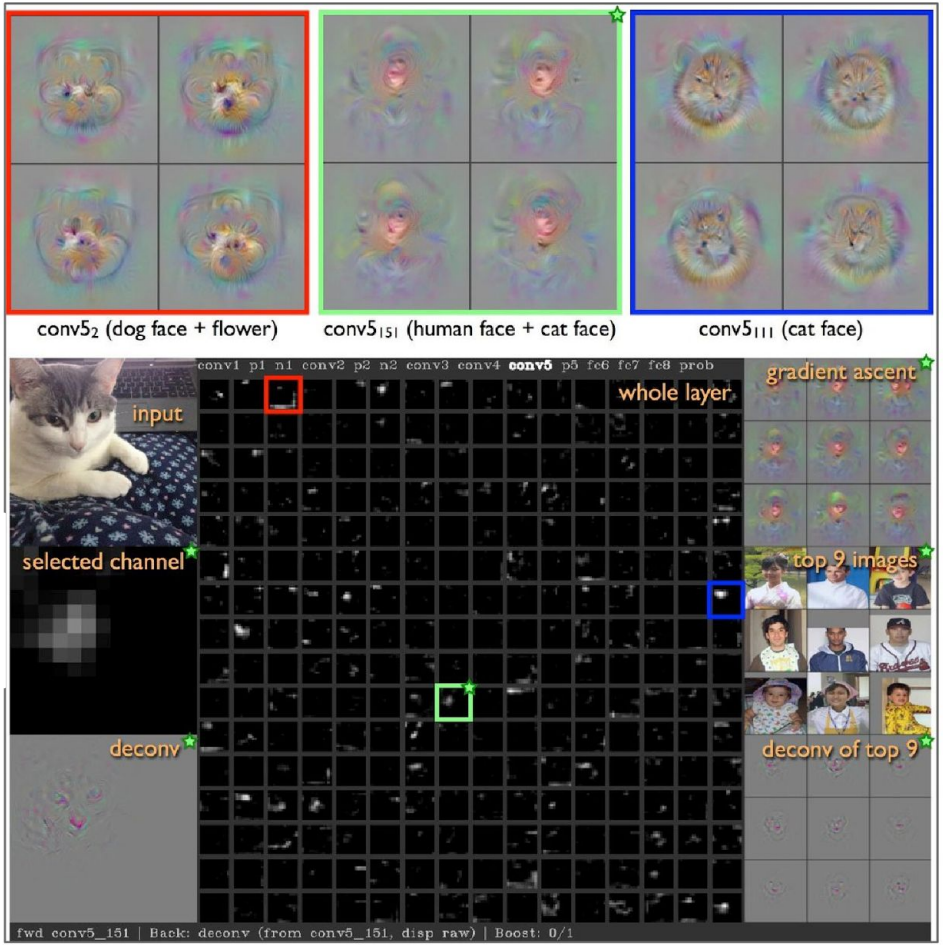
(as a function of the position of the square of zeros in the original image)

Slide taken from CS231n@stanford

Visualizing Activations

http://yosinski.com/deepvis

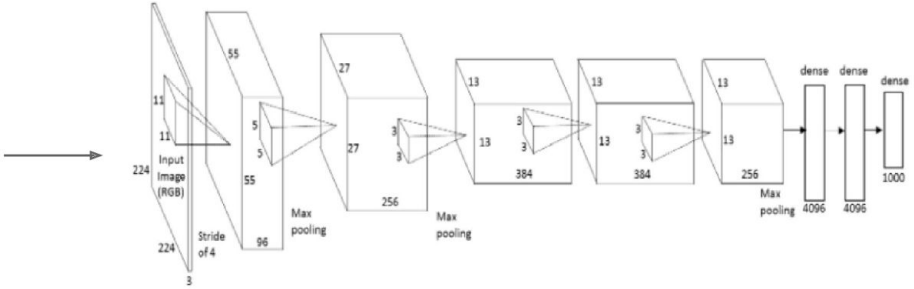
YouTube video
https://www.youtube.com/watch?&v=AgkfIQ4IGaM
(4min)



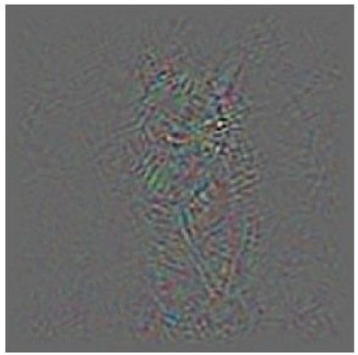
Slide taken from CS231n@stanford

Deconv approaches

1. Feed image into net



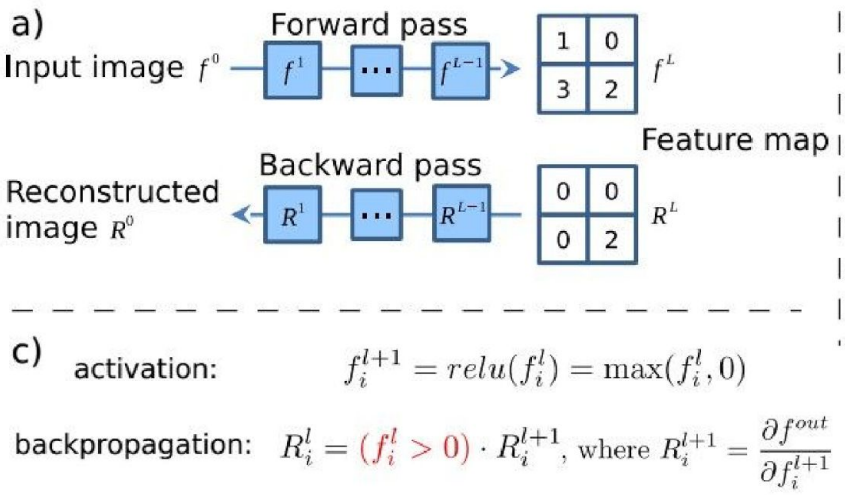
2. Pick a layer, set the gradient there to be all zero except for one 1 for some neuron of interest
3. Backprop to image:

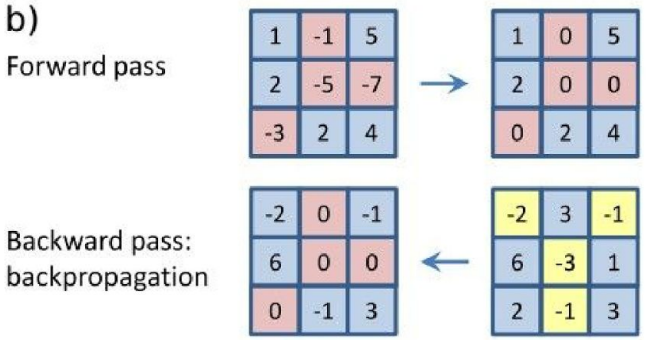


"Guided backpropagation:" instead

Deconv approaches

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]

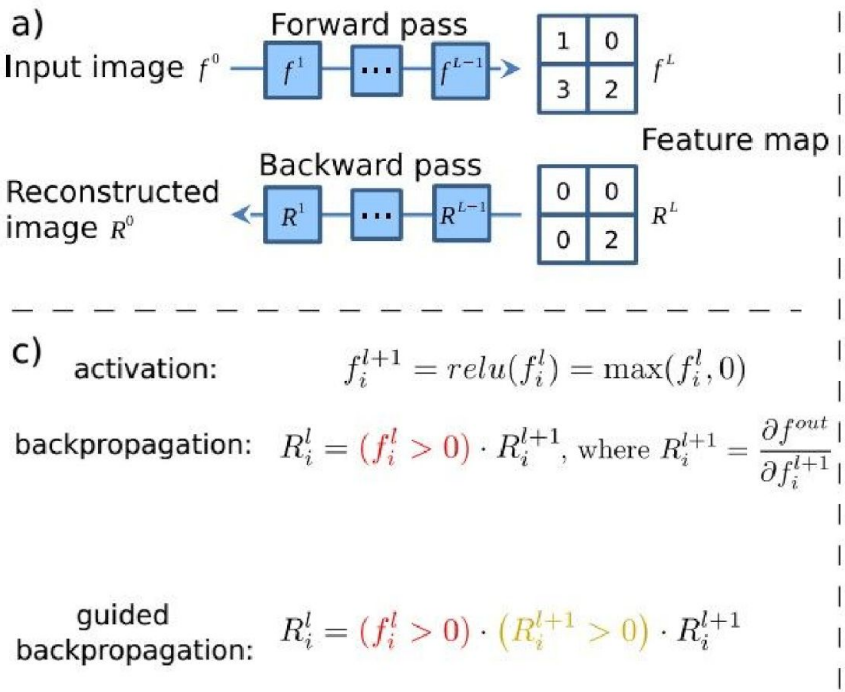




Backward pass for a ReLU (will be changed in Guided Backprop)

Deconv approaches

[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



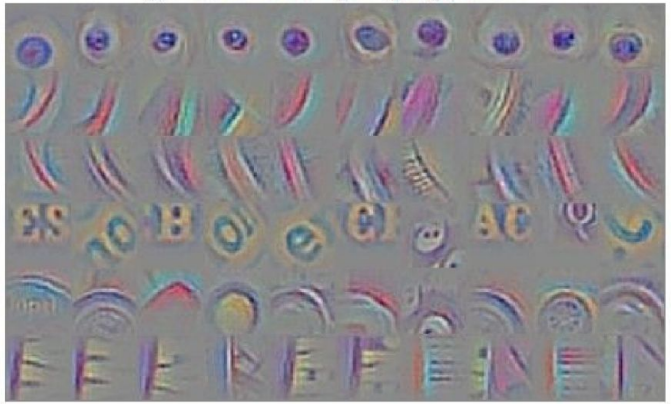
Slide taken from CS231n@stanford

Visualization of patterns learned by the layer **conv6** (top) and layer **conv9** (bottom) of the network trained on ImageNet.

Each row corresponds to one filter.

The visualization using "guided backpropagation" is based on the top 10 image patches activating this filter taken from the ImageNet dataset.

guided backpropagation



corresponding image crops

guided backpropagation

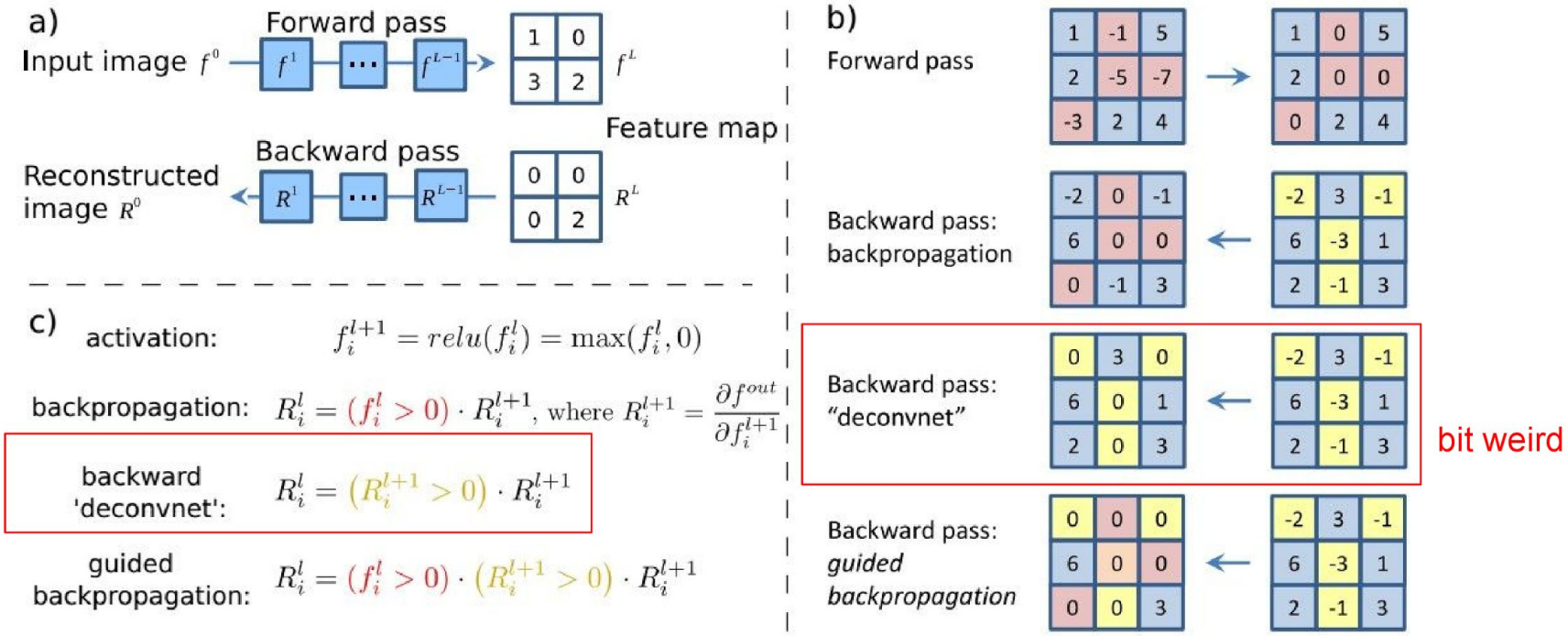
corresponding image crops

[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]

Slide taken from CS231n@stanford

Deconv approaches

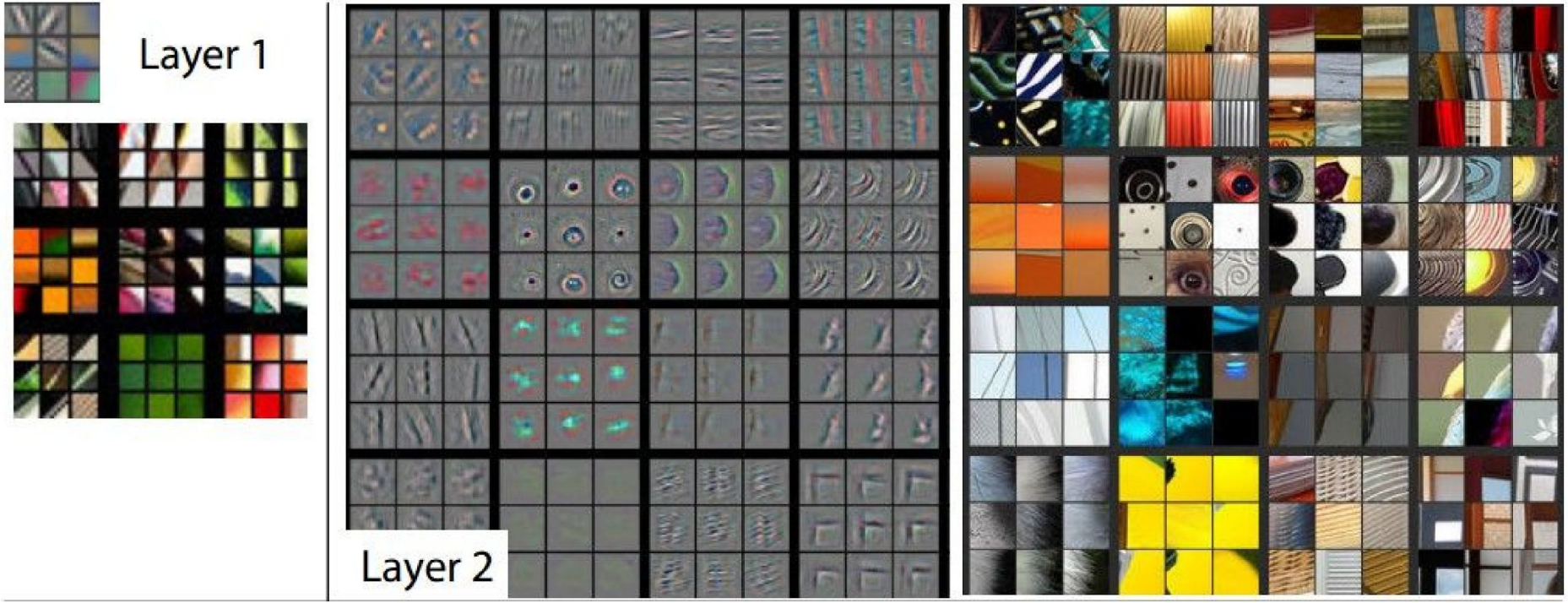
[Visualizing and Understanding Convolutional Networks, Zeiler and Fergus 2013]
[Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps, Simonyan et al., 2014]
[Striving for Simplicity: The all convolutional net, Springenberg, Dosovitskiy, et al., 2015]



Slide taken from CS231n@stanford

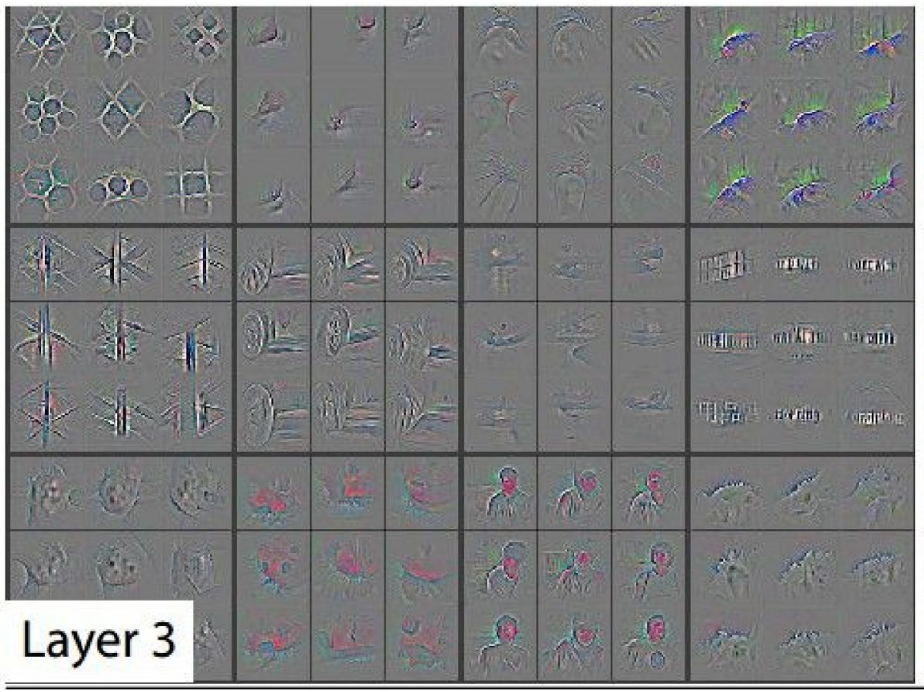
Visualizing and Understanding Convolutional Networks
Zeiler & Fergus, 2013

Visualizing arbitrary neurons along the way to the top...



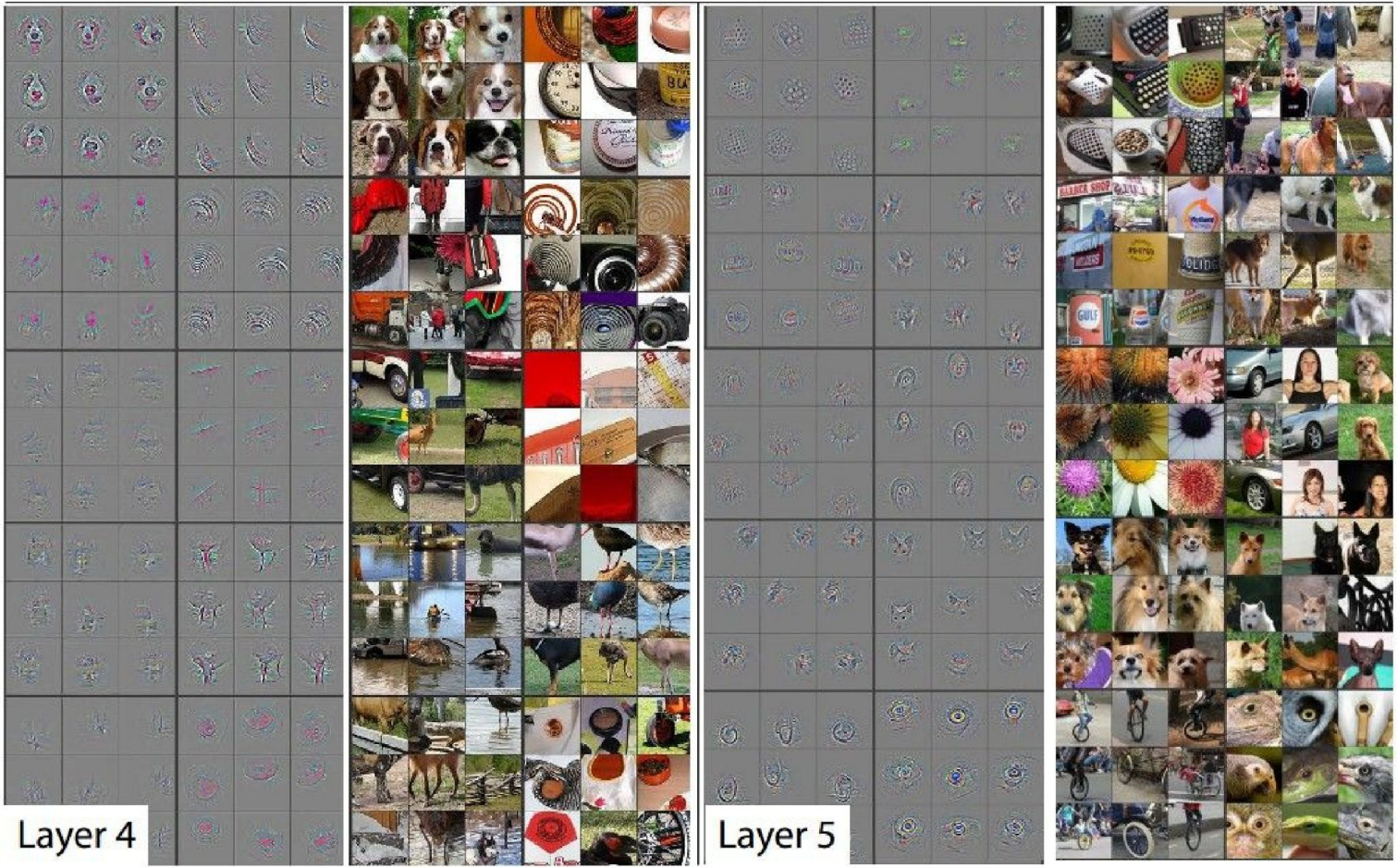
Slide taken from CS231n@stanford

Visualizing arbitrary neurons along the way to the top...



Slide taken from CS231n@stanford

Visualizing arbitrary neurons along the way to the top...



Slide taken from CS231n@stanford

Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

Visualize the Data gradient:



M = ?

(note that the gradient on data has three channels. Here they visualize M, s.t.:

$$M_{ij} = \max_c |w_{h(i,j,c)}|$$

(at each pixel take abs val, and max over channels)

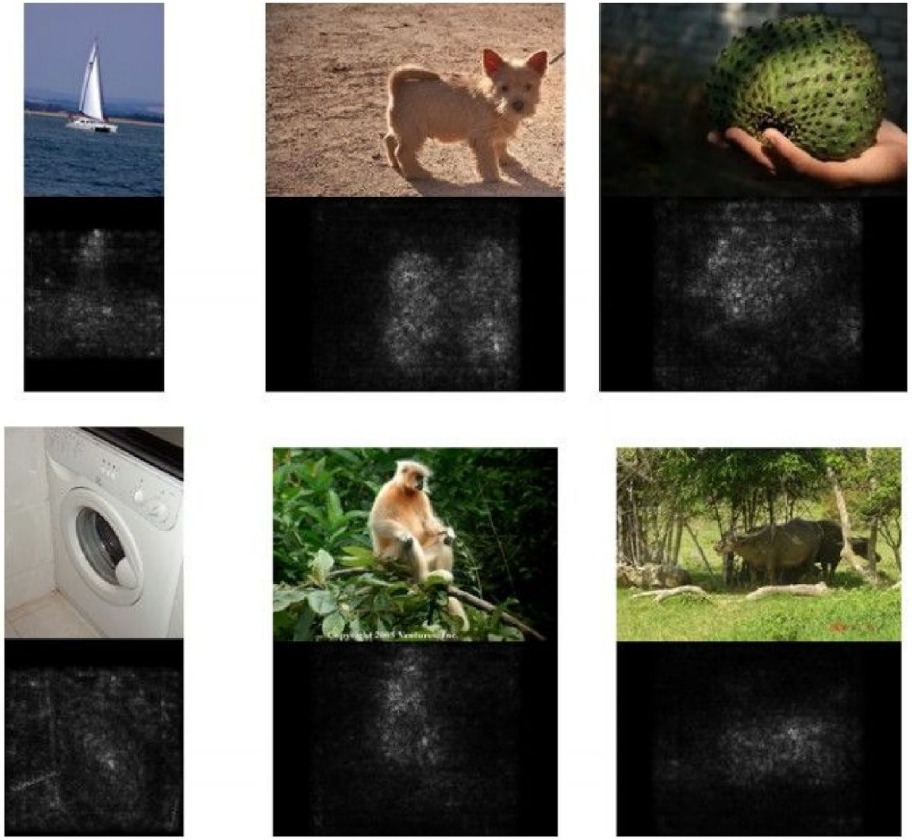
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

2. Visualize the Data gradient:

(note that the gradient on data has three channels. Here they visualize M, s.t.:

$$M_{ij} = \max_c |w_{h(i,j,c)}|$$

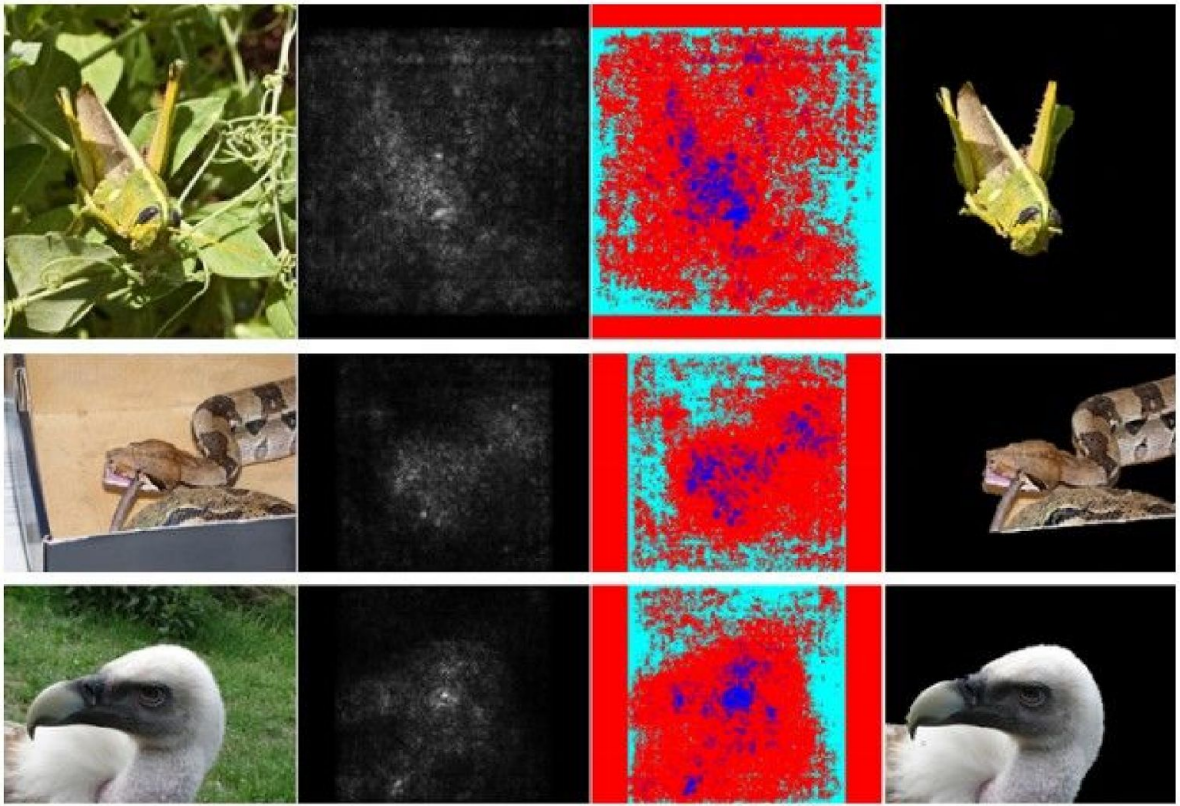
(at each pixel take abs val, and max over channels)



Slide taken from CS231n@stanford

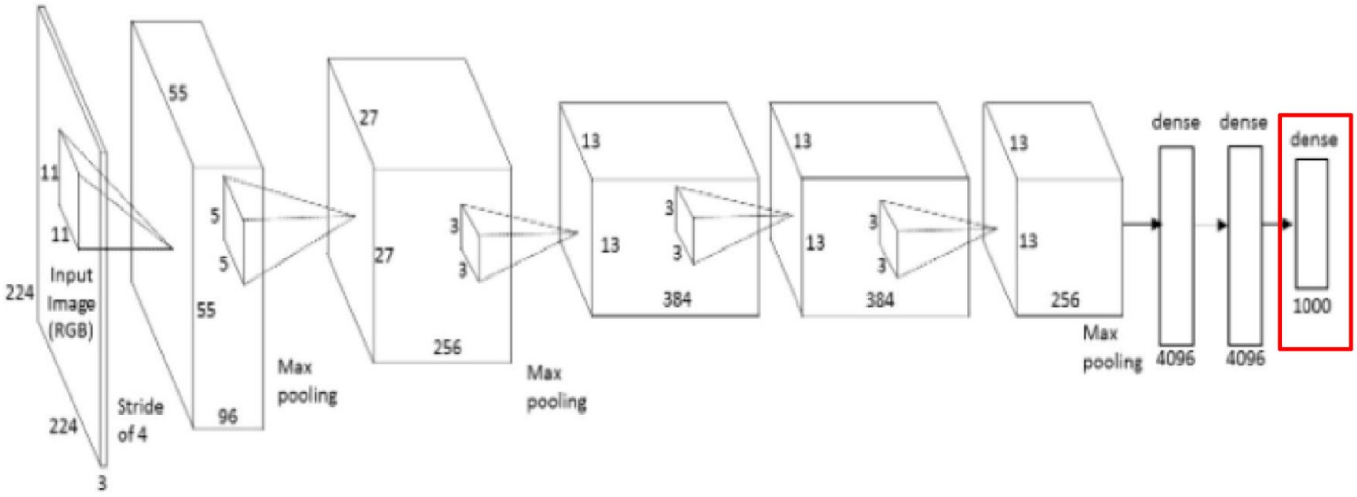
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

- Use **grabcut** for segmentation



Slide taken from CS231n@stanford

Optimization to Image



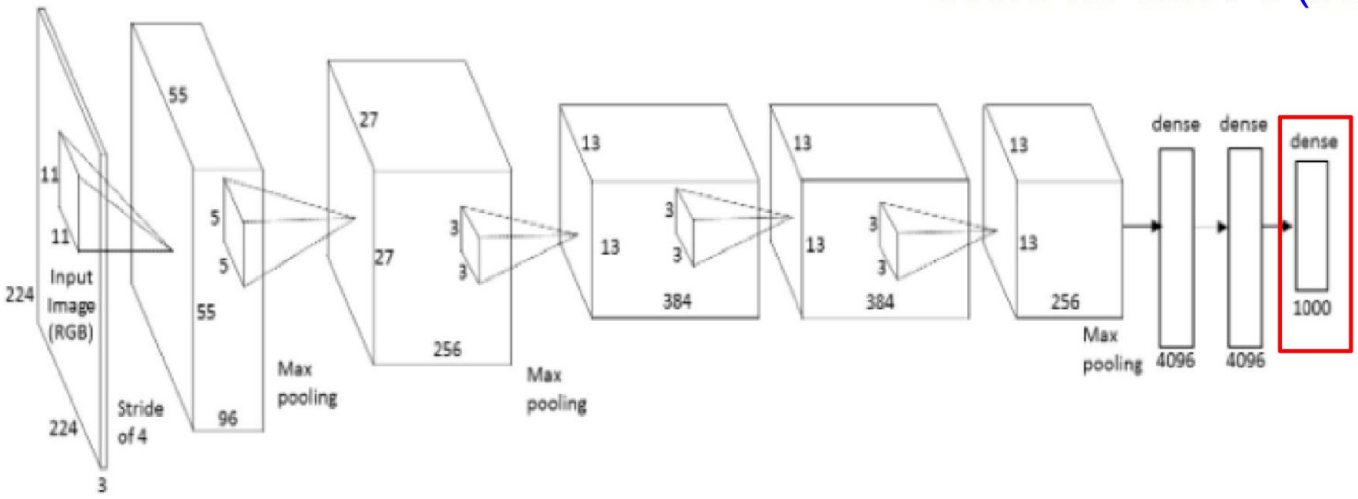
Q: can we find an image that maximizes some class score?

Slide taken from CS231n@stanford

Optimization to Image

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

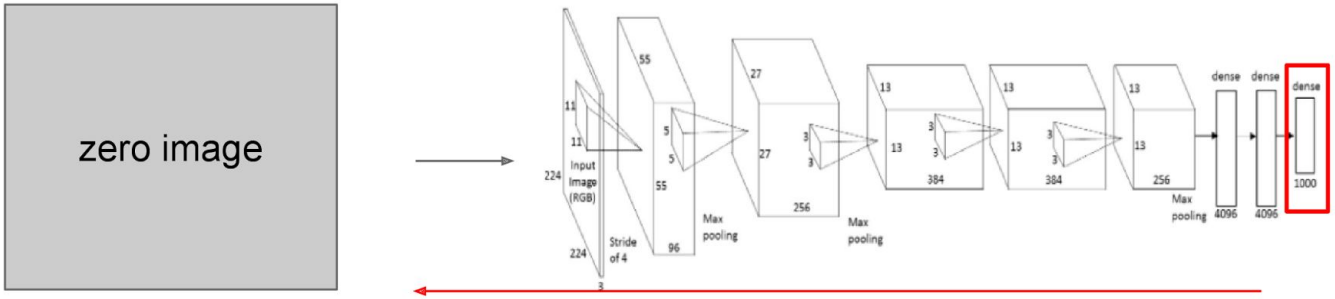


Q: can we find an image that maximizes some class score?

Slide taken from CS231n@stanford

Optimization to Image

1. feed in zeros.

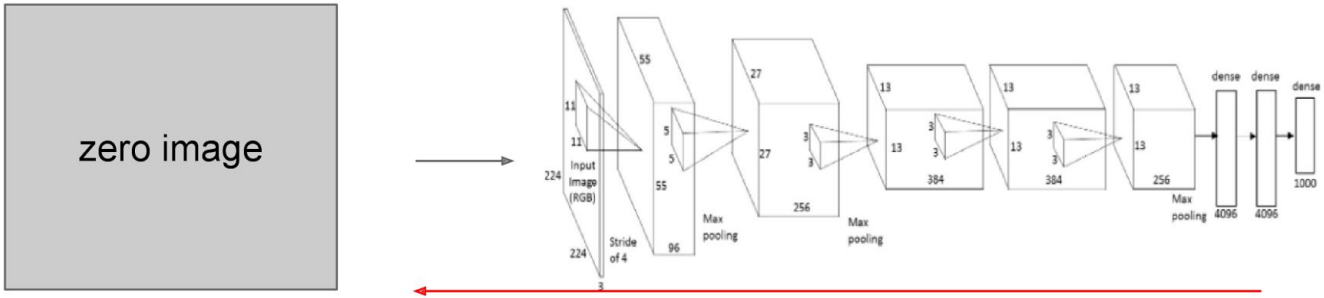


2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image

Slide taken from CS231n@stanford

Optimization to Image

1. feed in zeros.



2. set the gradient of the scores vector to be [0,0,....1,....,0], then backprop to image
3. do a small “image update”
4. forward the image through the network.
5. go back to 2.

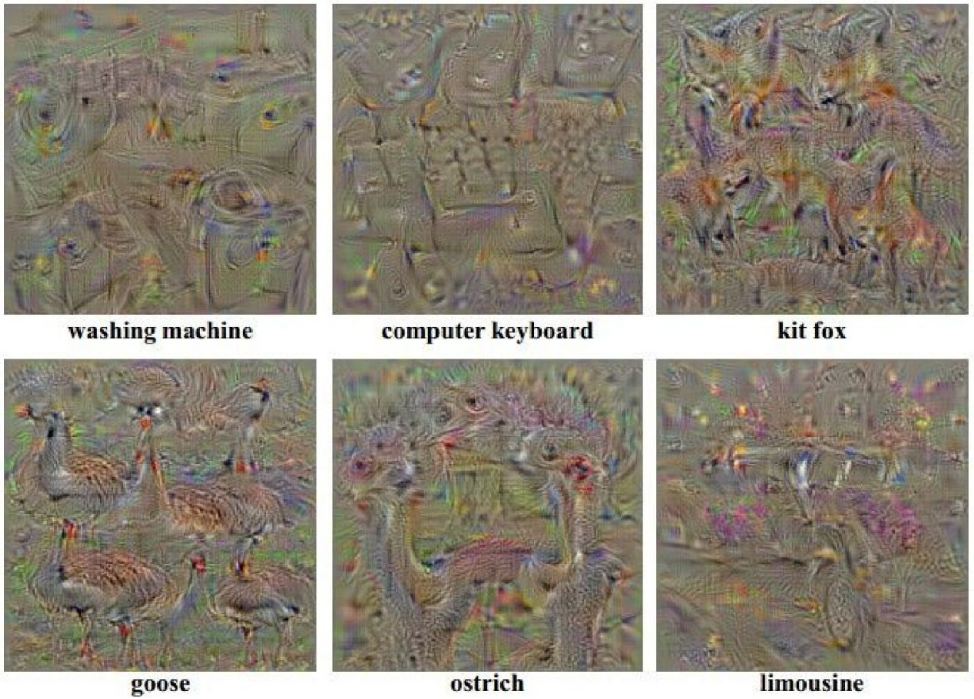
$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

score for class c (before Softmax)

Slide taken from CS231n@stanford

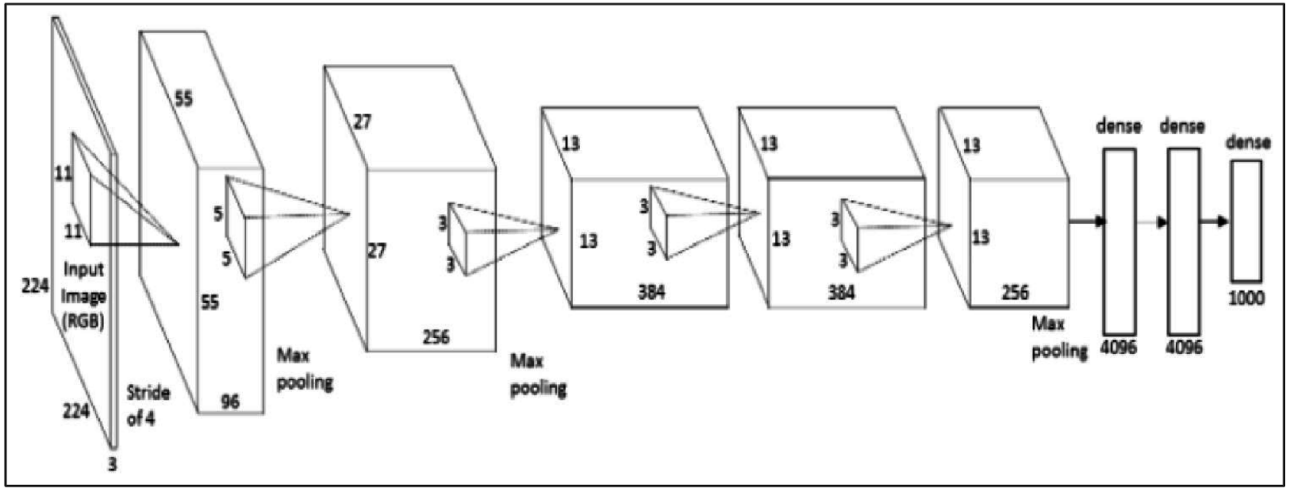
Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps
Karen Simonyan, Andrea Vedaldi, Andrew Zisserman, 2014

Find images that maximize some class score:



Slide taken from CS231n@stanford

We can in fact do this for arbitrary neurons along the ConvNet



Repeat:

1. Forward an image
2. Set activations in layer of interest to all zero, except for a 1.0 for a neuron of interest
3. Backprop to image
4. Do an “image update”

Slide taken from CS231n@stanford

[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]

Proposed a different form of regularizing the image

$$\arg\max_I S_c(I) - \lambda \|I\|_2^2$$

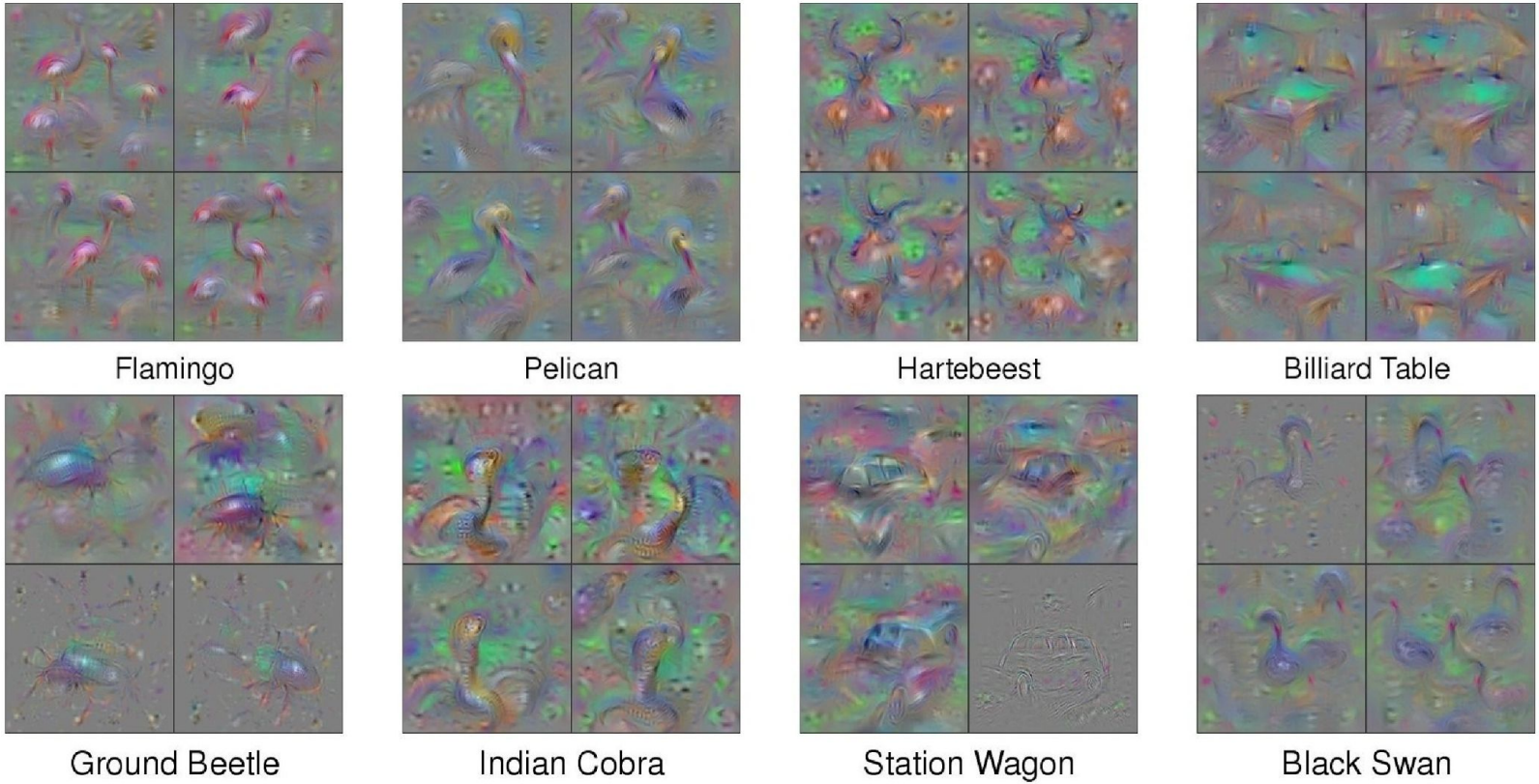
More explicit scheme:

Repeat:

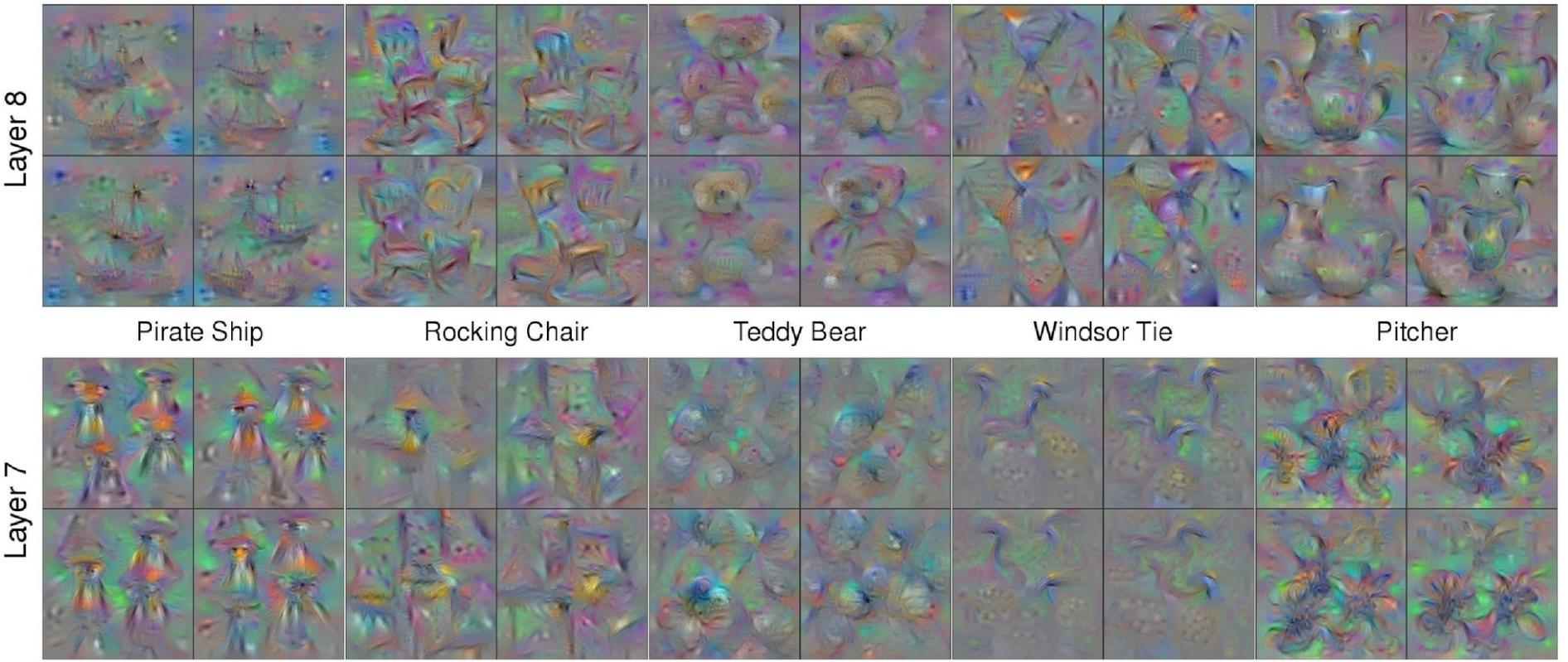
- Update the image **x** with gradient from some unit of interest
- Blur x a bit
- Take any pixel with small norm to zero (to encourage sparsity)

[Understanding Neural Networks Through Deep Visualization, Yosinski et al. , 2015]

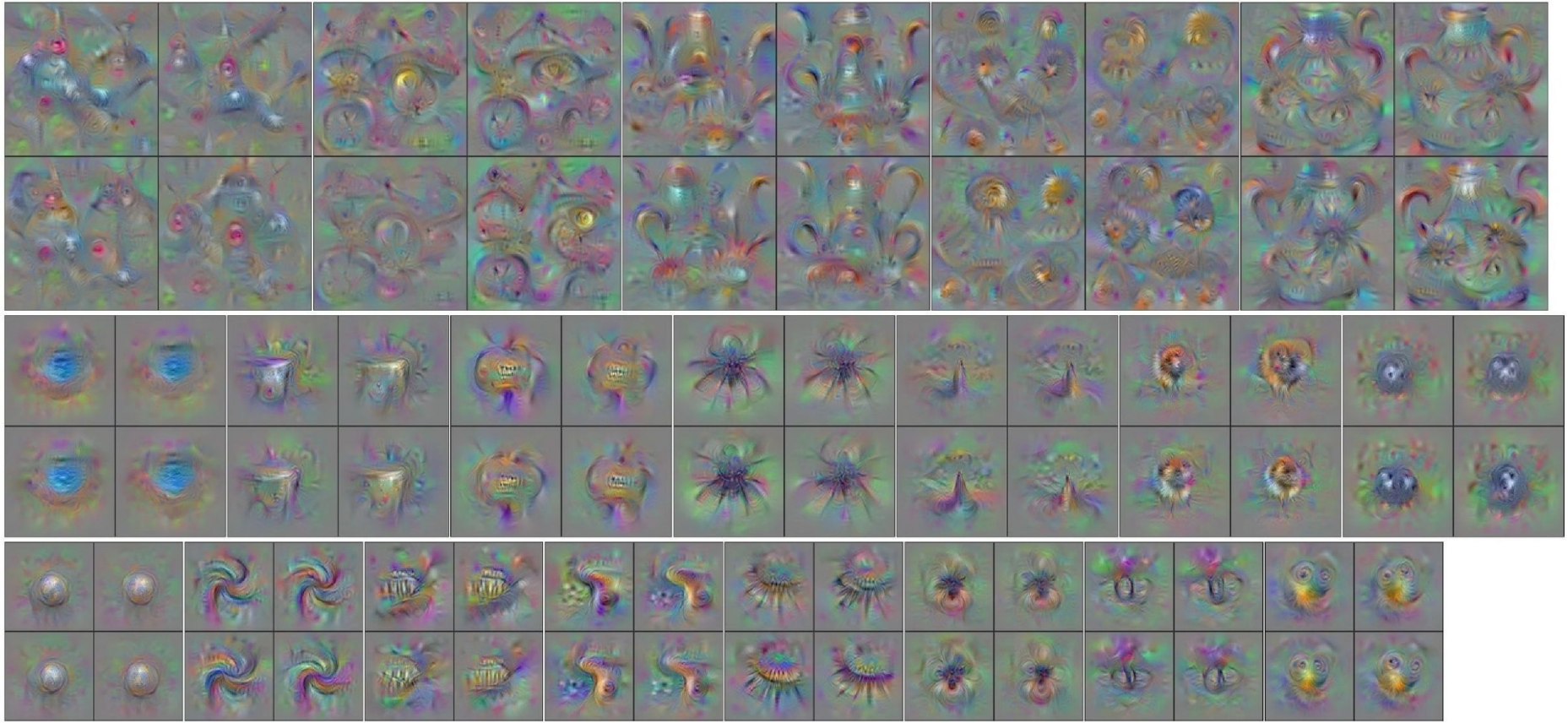
http://yosinski.com/deepvis



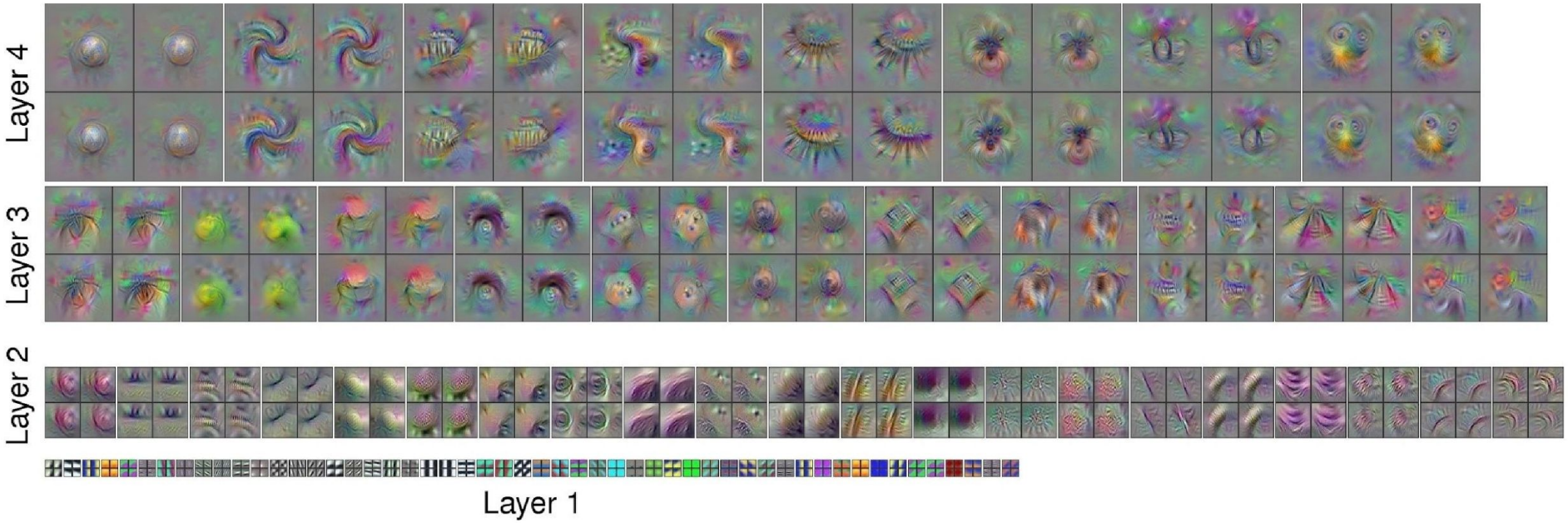
Slide taken from CS231n@stanford



Slide taken from CS231n@stanford



Slide taken from CS231n@stanford



Slide taken from CS231n@stanford

Adversarial examples

We can pose an optimization over the input image to maximize any class score.
That seems useful.

Question: Can we use this to “fool” ConvNets?

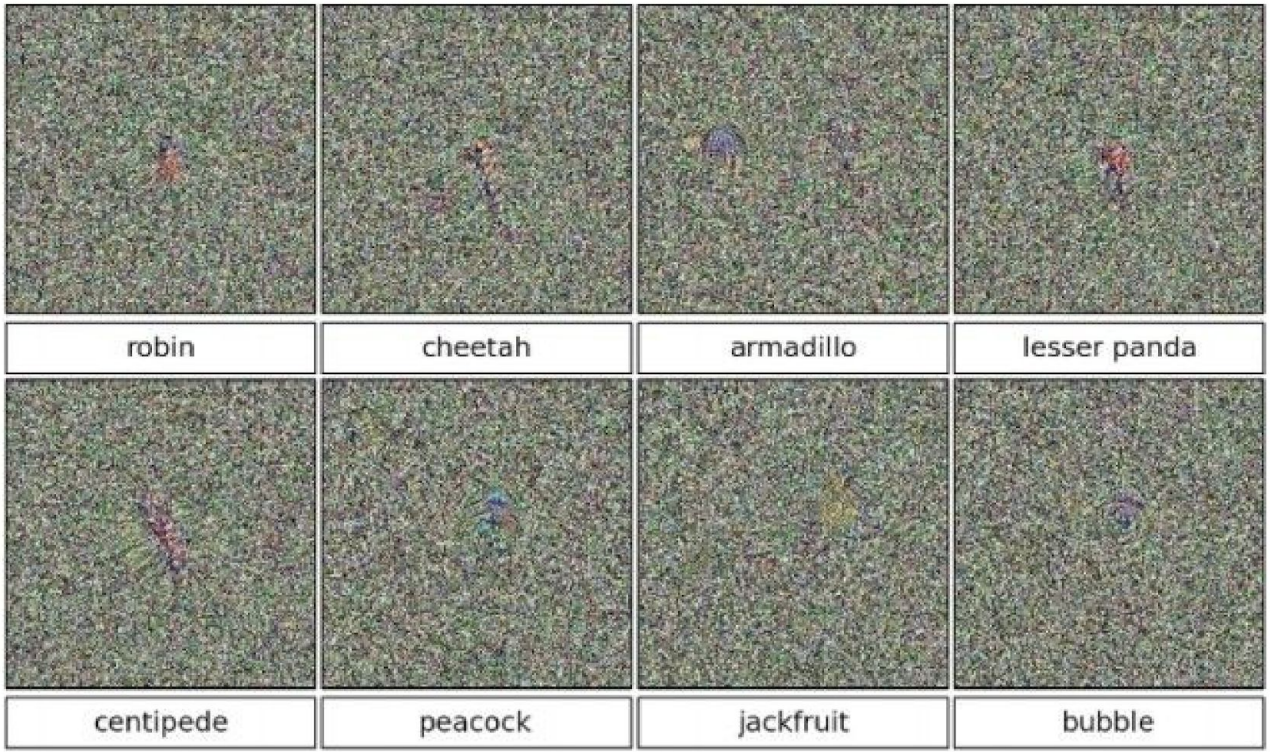
spoiler alert: yeah

[Intriguing properties of neural networks, Szegedy et al., 2013]

Slide taken from CS231n@stanford

[Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images Nguyen, Yosinski, Clune, 2014]

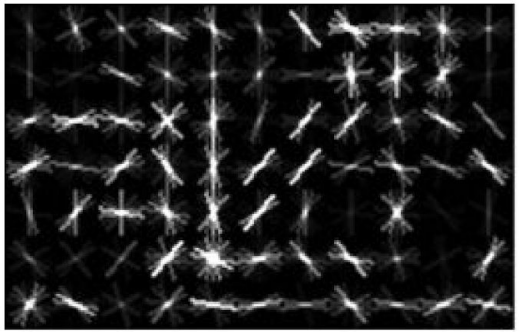
>99.6% confidences



Slide taken from CS231n@stanford

These kinds of results were around even before ConvNets…

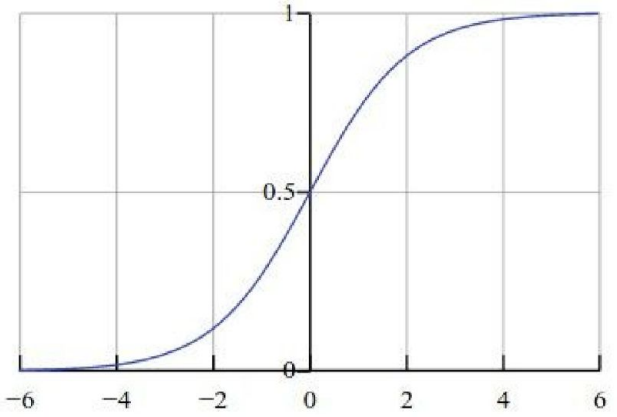
[Exploring the Representation Capabilities of the HOG Descriptor, Tatu et al., 2011]



Identical HOG represention

Lets fool a binary linear classifier: (logistic regression)

$$P(y = 1 \mid x; w, b) = \frac{1}{1 + e^{-(w^T x + b)}} = \sigma(w^T x + b)$$



Since the probabilities of class 1 and 0 sum to one, the probability for class 0 is $P(y = 0 \mid x; w, b) = 1 - P(y = 1 \mid x; w, b)$. Hence, an example is classified as a positive example (y = 1) if $\sigma(w^T x + b) > 0.5$, or equivalently if the score $w^T x + b > 0$.

Slide taken from CS231n@stanford

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1	← input example
w	-1	-1	1	-1	1	-1	1	1	-1	1	← weights

class 1 score = dot product:
= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474
i.e. the classifier is **95%** certain that this is class 0 example.

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Slide taken from CS231n@stanford

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	?	?	?	?	?	?	?	?	?	?

← input example

← weights

class 1 score = dot product:
= -2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3
=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474
i.e. the classifier is **95%** certain that this is class 0 example.

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Slide taken from CS231n@stanford

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

← input example

← weights

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2

=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

i.e. we improved the class 1 probability from 5% to 88%

$$P(y=1 \mid x; w, b) = \frac{1}{1+e^{-(w^T x+b)}} = \sigma(w^T x + b)$$

Slide taken from CS231n@stanford

Lets fool a binary linear classifier:

x	2	-1	3	-2	2	2	1	-4	5	1
w	-1	-1	1	-1	1	-1	1	1	-1	1
adversarial x	1.5	-1.5	3.5	-2.5	2.5	1.5	1.5	-3.5	4.5	1.5

← input example

← weights

class 1 score before:

-2 + 1 + 3 + 2 + 2 - 2 + 1 - 4 - 5 + 1 = -3

=> probability of class 1 is 1/(1+e^(-(-3))) = 0.0474

-1.5+1.5+3.5+2.5+2.5-1.5+1.5-3.5-4.5+1.5 = 2

=> probability of class 1 is now 1/(1+e^(-(2))) = 0.88

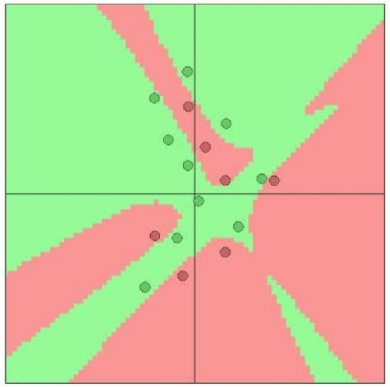
i.e. we improved the class 1 probability from 5% to 88%

This was only with 10 input dimensions. A 224x224 input image has 150,528.

(It's significantly easier with more numbers, need smaller nudge for each)

Slide taken from CS231n@stanford

EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES
[Goodfellow, Shlens & Szegedy, 2014]



“primary cause of neural networks’ vulnerability to adversarial perturbation is their **linear nature**“
(and very high-dimensional, sparsely-populated input spaces)

In particular, this is not a problem with Deep Learning, and has little to do with ConvNets specifically. Same issue would come up with Neural Nets in any other modalities.