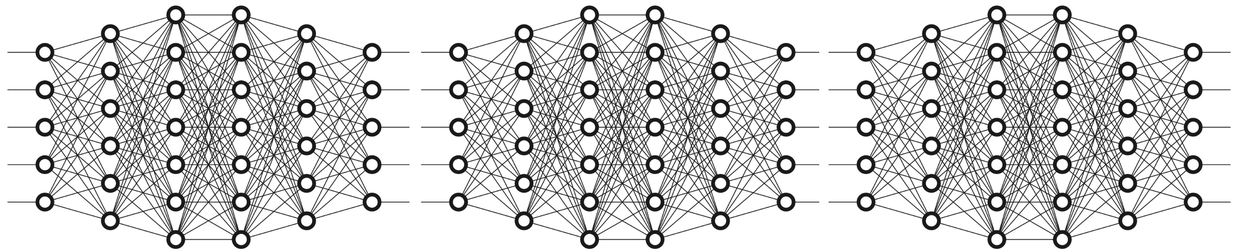
THE UNIVERSITY OF BRITISH COLUMBIA

CPSC 425: Computer Vision



Lecture 23: Neural Network Vision Applications

Menu for Today

Topics:

- Object Detection
- Segmentation
- Image Generation
- Quiz 5 and 6 **online** (see Canvas)

Note: grading same 0.5 point for participation

Redings:

- **Today's** Lecture: N/A

Reminders:

- **Assignment 6**: Deep Learning is out and due next **Thursday**
- **No** lecture on **Monday** (holiday)
- **No** office hours on **Friday** (holiday)
- Additional **Office Hours** week after next (for Final Prep)

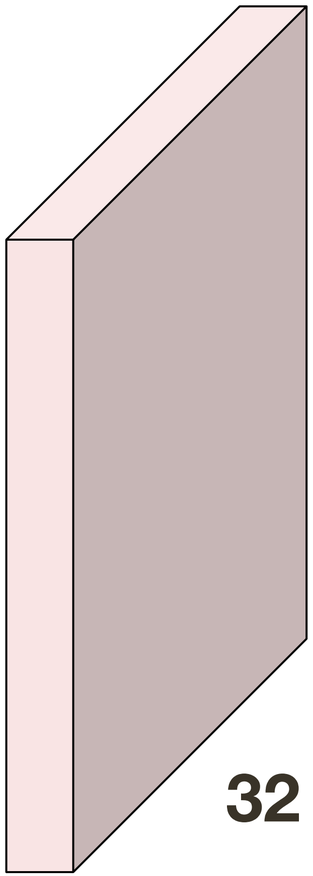
Please fill out

Student Evaluations

(on Canvas)

Review Lecture 22: **Convolutional** Layer

32 x 32 x 3 **image**



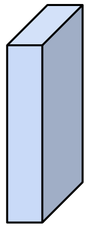
32 height

32 width

3 depth

Filters always extend the full depth of the input volume

5 x 5 x 3 **filter**

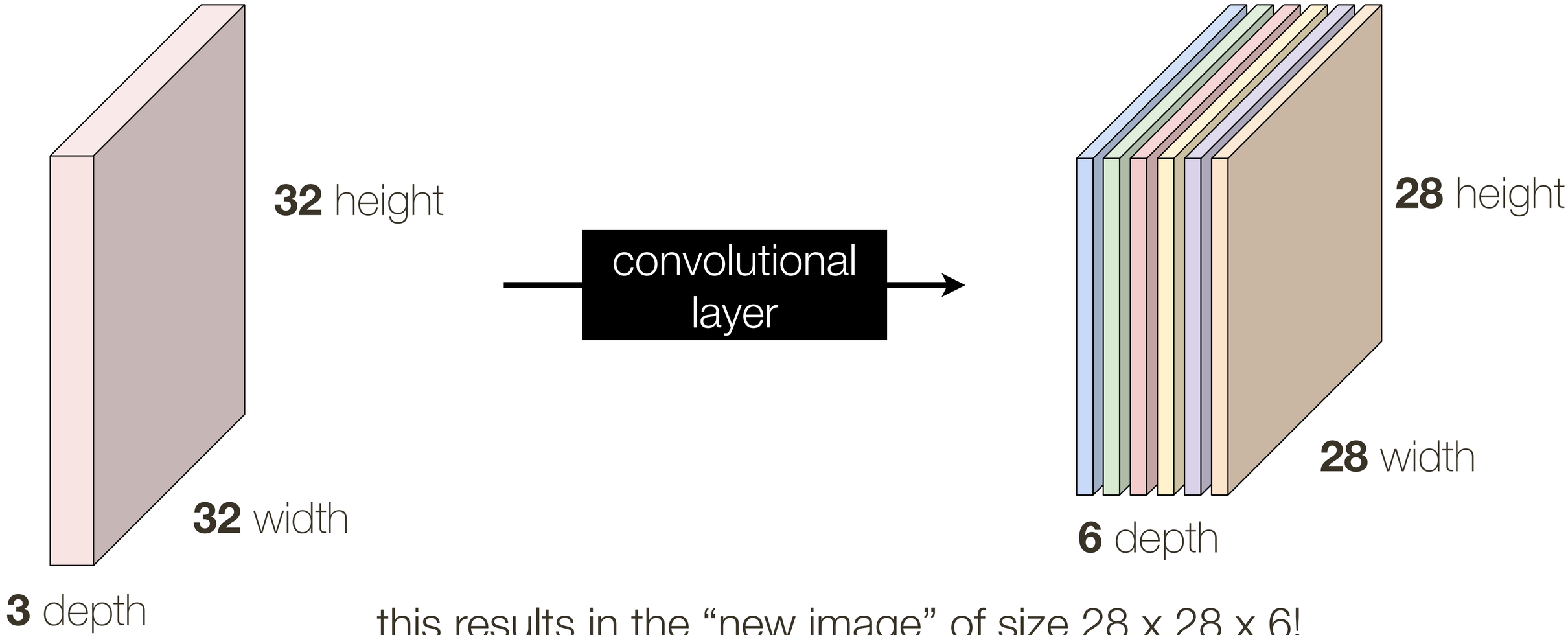


Convolve the filter with the image (i.e., “slide over the image spatially, computing dot products”

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Review Lecture 22: **Convolutional** Layer

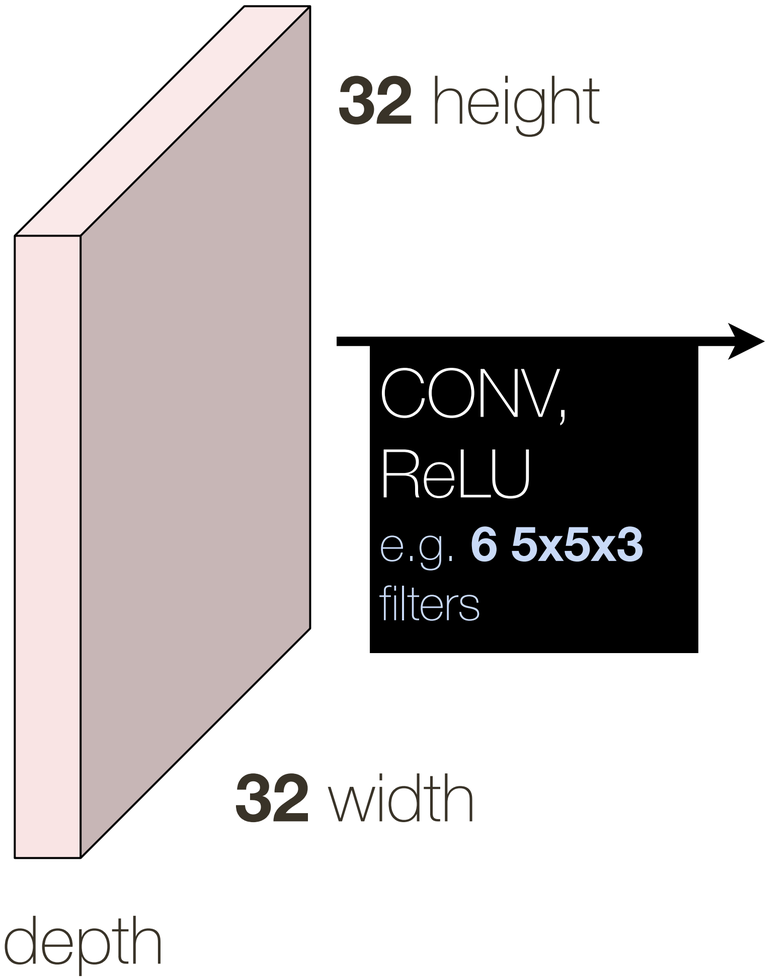
If we have 6 5x5 filter, we'll get 6 separate activation maps: **activation** map



this results in the "new image" of size 28 x 28 x 6!

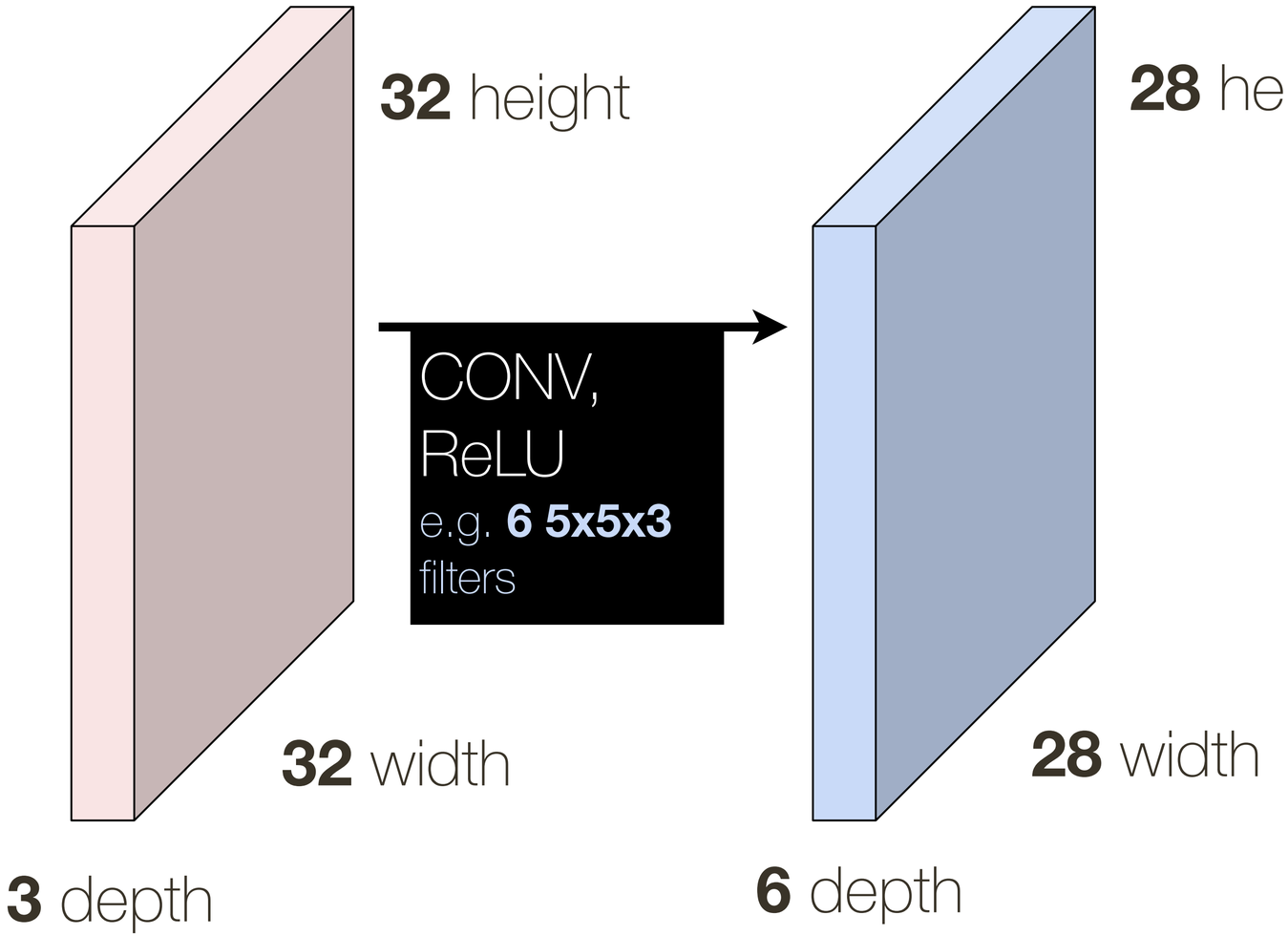
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Review Lecture 22: **Convolutional** Neural Network (ConvNet)



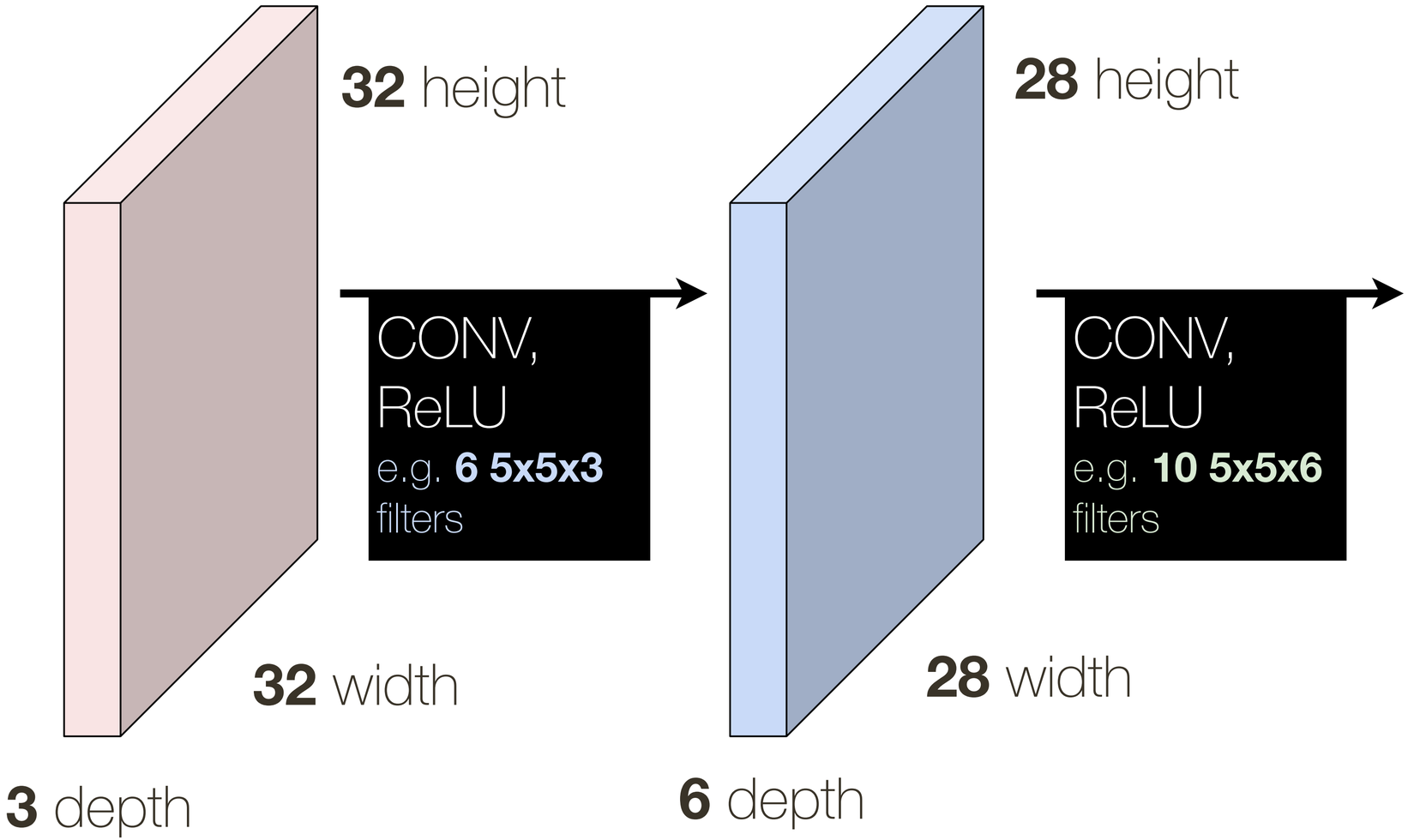
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Review Lecture 22: **Convolutional** Neural Network (ConvNet)



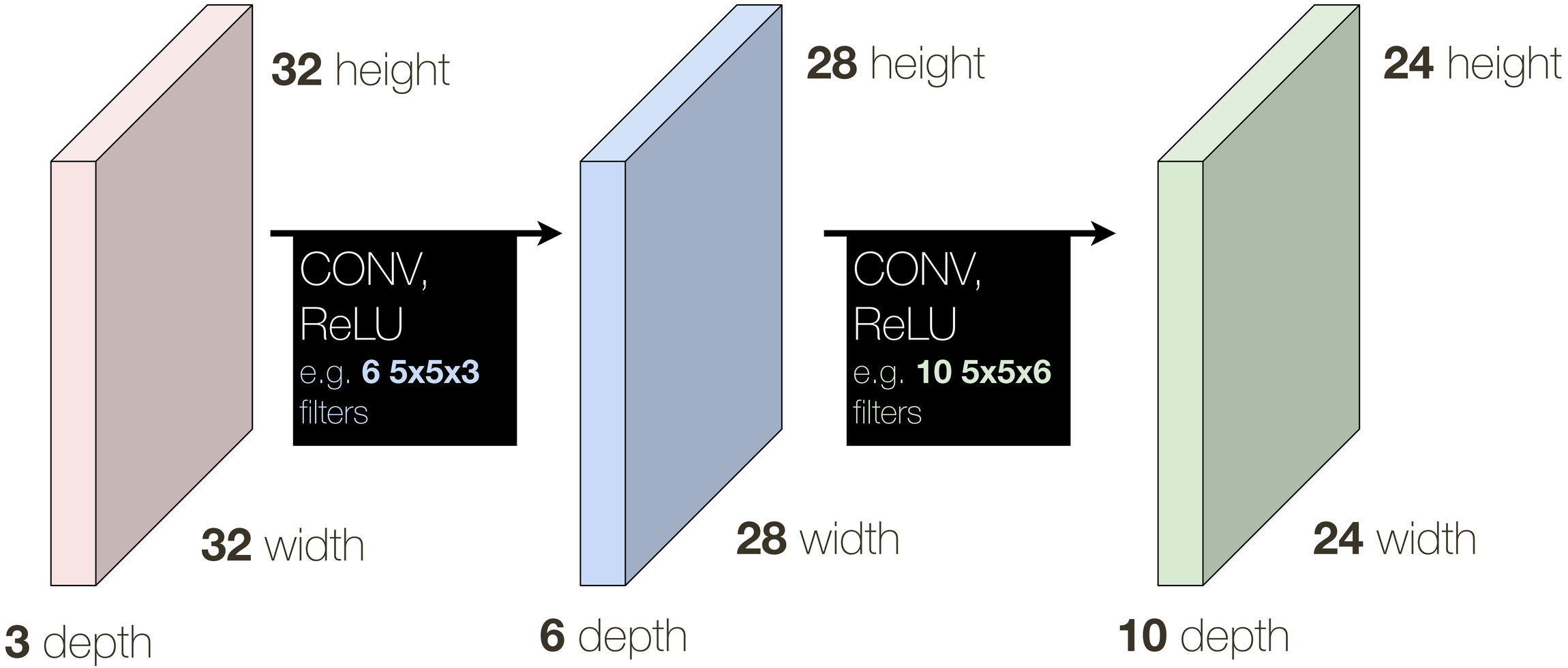
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Review Lecture 22: **Convolutional** Neural Network (ConvNet)



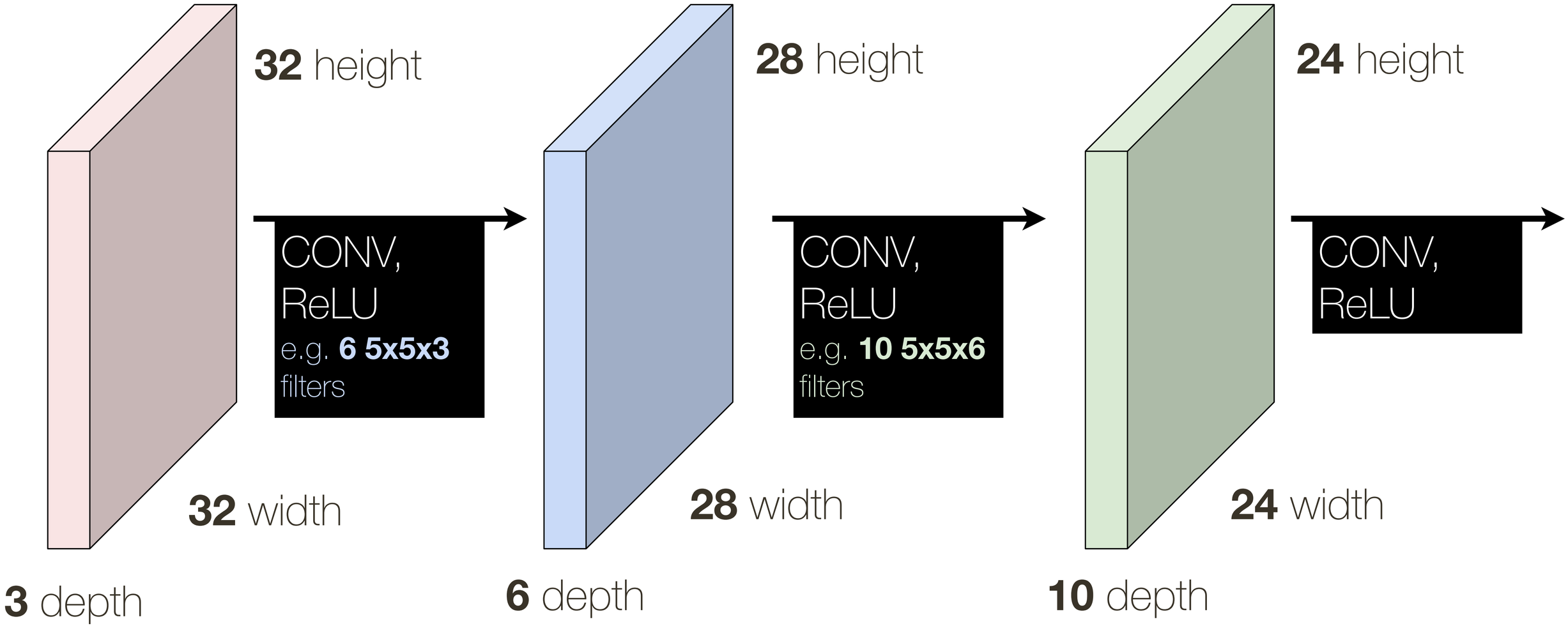
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Review Lecture 22: **Convolutional** Neural Network (ConvNet)



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

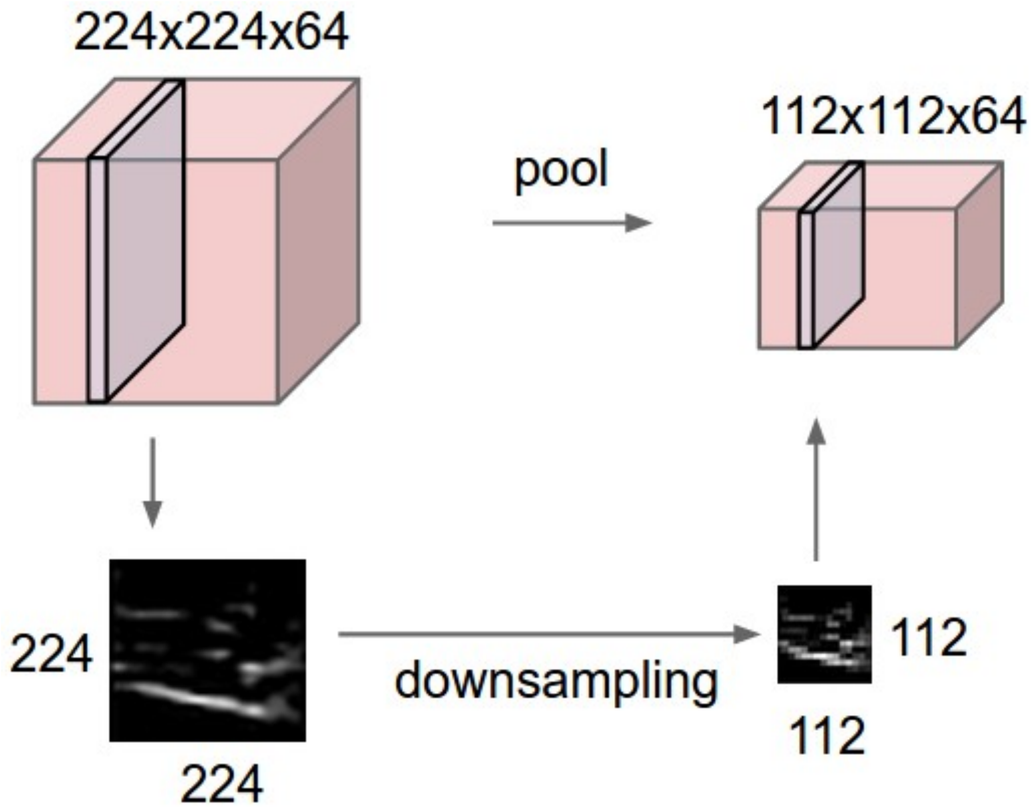
Review Lecture 22: **Convolutional** Neural Network (ConvNet)



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

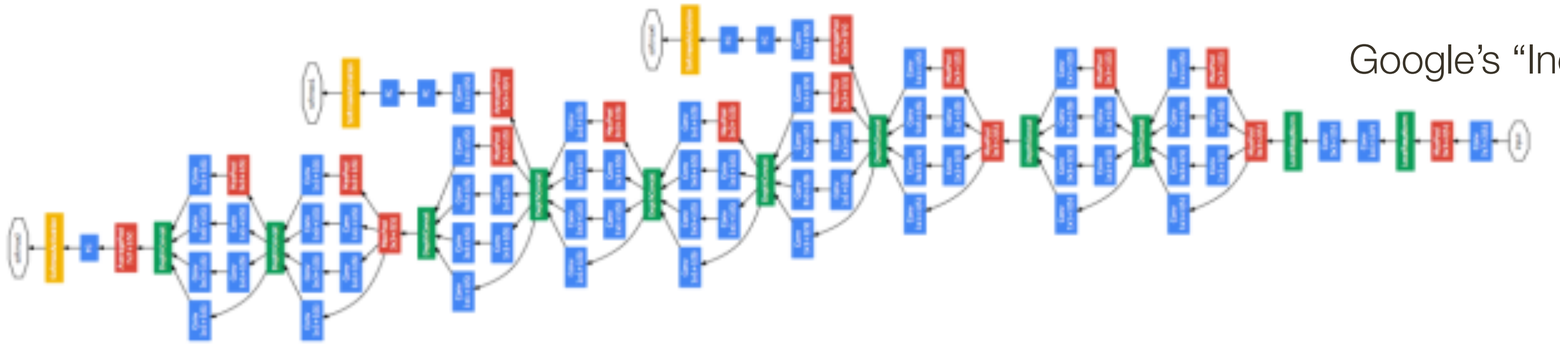
Review Lecture 22: **Pooling** Layer

- Makes representation smaller, more manageable and spatially invariant
- Operates over each activation map independently



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Review Lecture 22: Deep Learning **Terminology**



Google's "Inception" network

- **Network structure:** number and types of layers, forms of activation functions, dimensionality of each layer and connections (defines computational graph)

generally kept fixed, requires some knowledge of the problem and NN to sensibly set

deeper = better

- **Loss function:** objective function being optimized (`softmax`, `cross entropy`, *etc.*)

requires knowledge of the nature of the problem

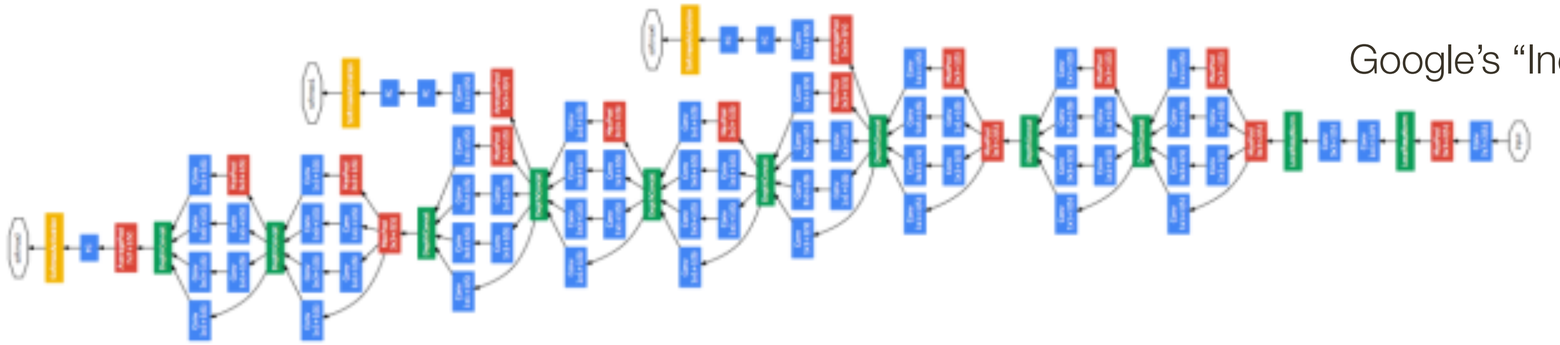
- **Parameters:** trainable parameters of the network, including weights/biases of linear/fc layers, parameters of the activation functions, *etc.*

optimized using SGD or variants

- **Hyper-parameters:** parameters, including for optimization, that are not optimized directly as part of training (*e.g.*, `learning rate`, `batch size`, `drop-out rate`)

grid search

Review Lecture 22: Deep Learning **Terminology**



Google's "Inception" network

- **Network structure:** number and types of layers, forms of activation functions, dimensionality of each layer and connections (defines computational graph)

generally kept fixed, requires some knowledge of the problem and NN to sensibly set

deeper = better

- **Loss function:** objective function being optimized (`softmax`, `cross entropy`, *etc.*)

requires knowledge of the nature of the problem

Specification of neural architecture will define a **computational** graph.

Computer **Vision Problems**

Computer **Vision Problems**

Categorization

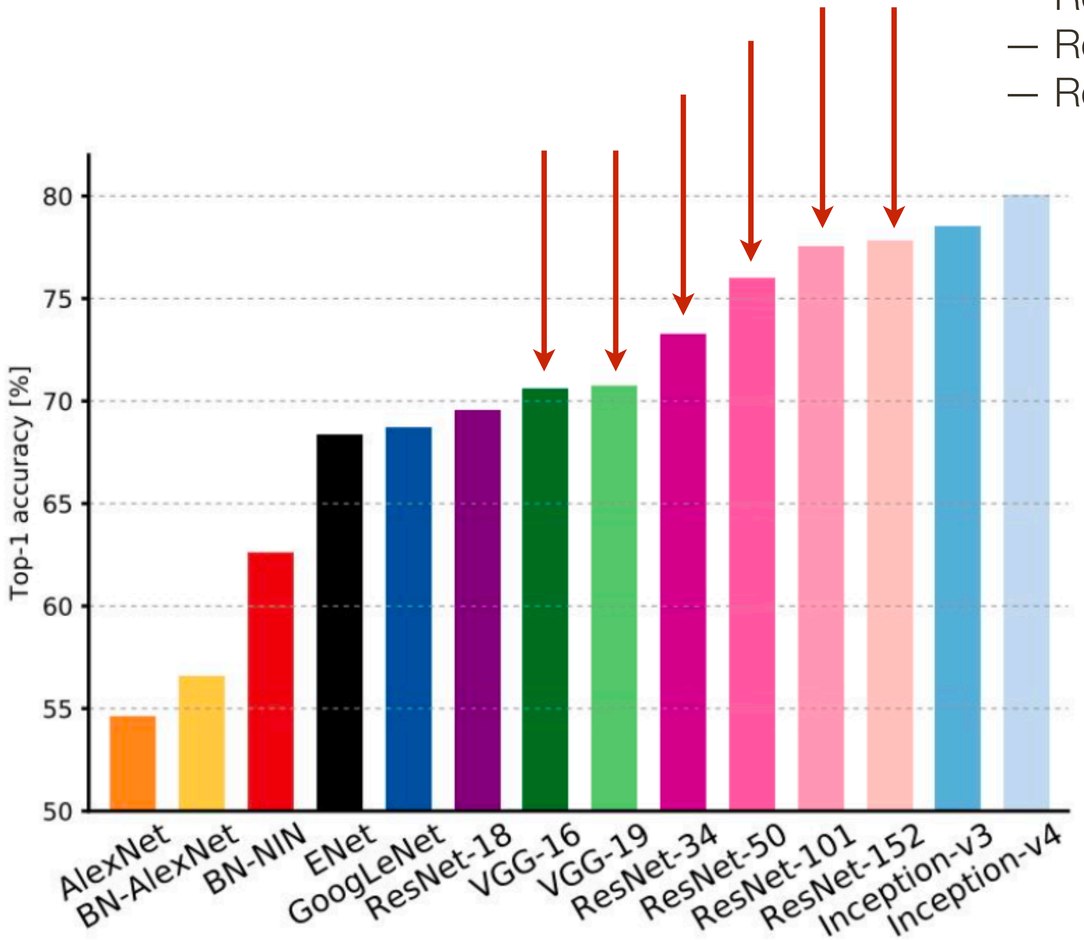
Multi-**class:**
Horse
Church
Toothbrush
Person

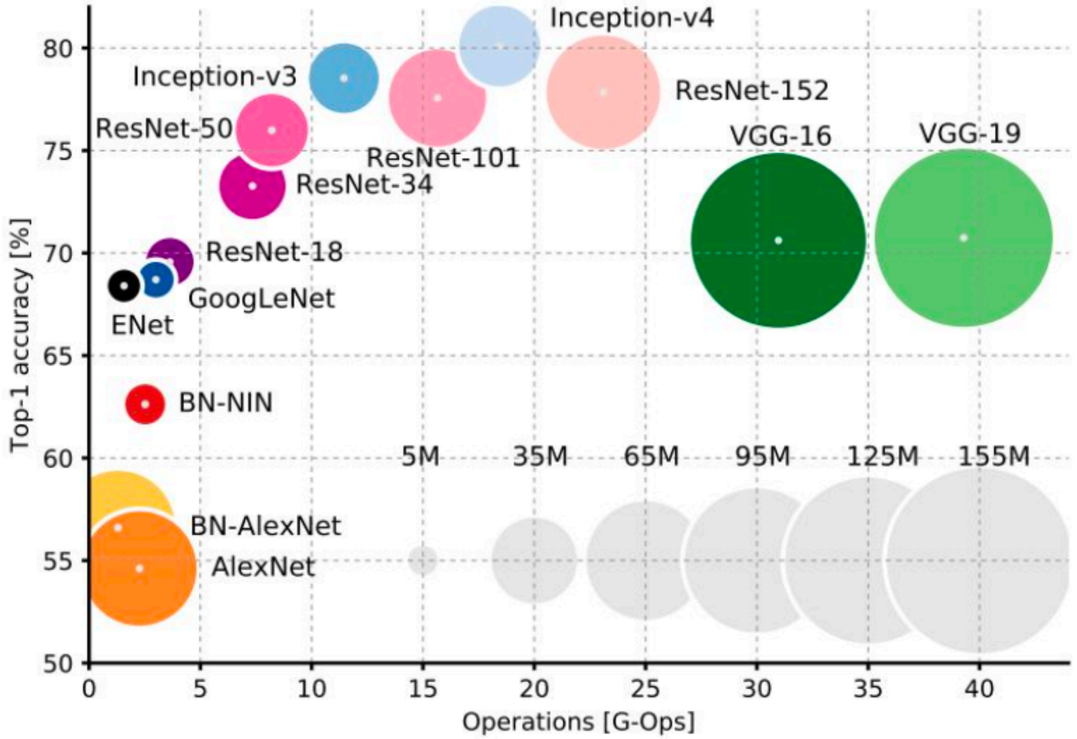
IMAGENET

Multi-**label**:
Horse
Church
Toothbrush
Person

Comparing **Complexity**

Trends:
- Deeper networks (more layers)
- Smaller filters (give same receptive fields with fewer parameters)
- Reduction of resolution /w increase of filters as you go deeper
- Residual connections for more stable training (avoid vanishing gradients)
- Reusable blocks (design a good structure for a small block use it recursively)





An Analysis of Deep Neural Network Models for Practical Applications, 2017.

* adopted from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Computer **Vision Problems** (no language for now)

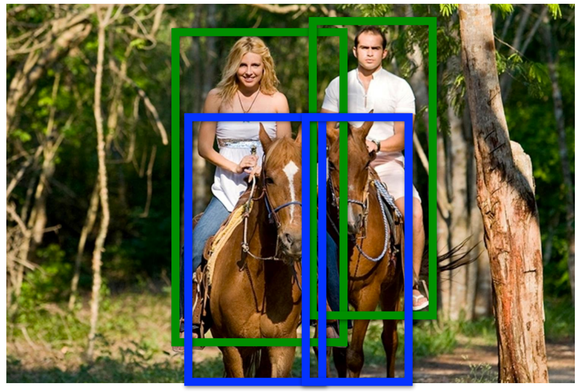
Categorization

Multi-**class:**
Horse
Church
Toothbrush
Person

IMAGENET

Multi-**label**:
Horse
Church
Toothbrush
Person

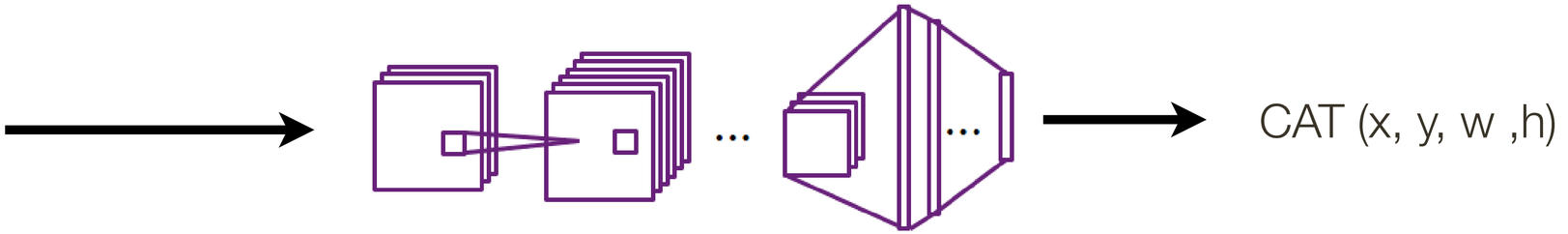
Detection



Horse (x, y, w, h)
Horse (x, y, w, h)
Person (x, y, w, h)
Person (x, y, w, h)

COCO
Common Objects in Context

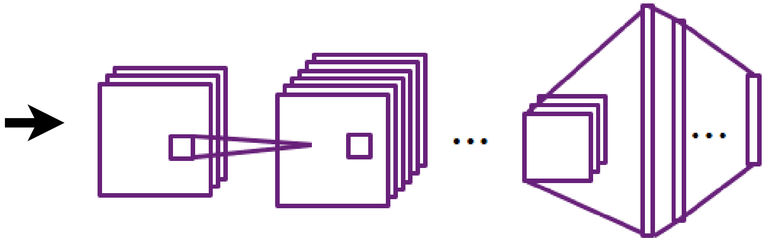
Object **Detection** as Regression Problem



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Regression Problem

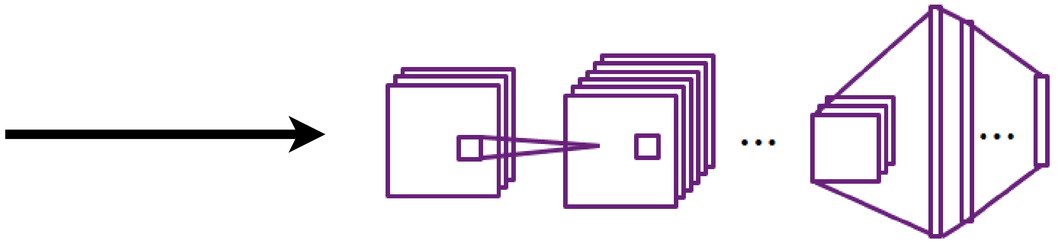
CAT (x, y, w ,h)



DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
…

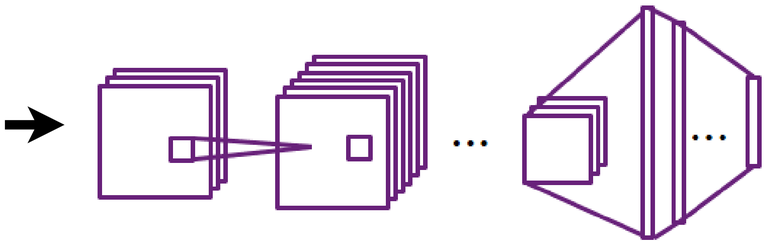
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Regression Problem



CAT (x, y, w ,h)

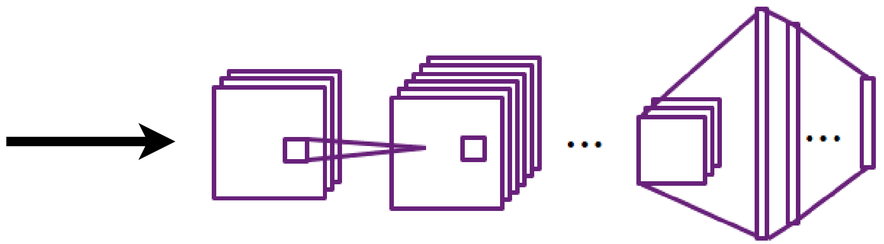
Problem: each image needs a different number of outputs



DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
DUCK (x, y, w ,h)
...

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem

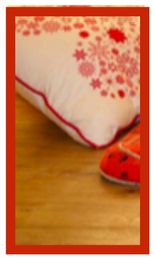


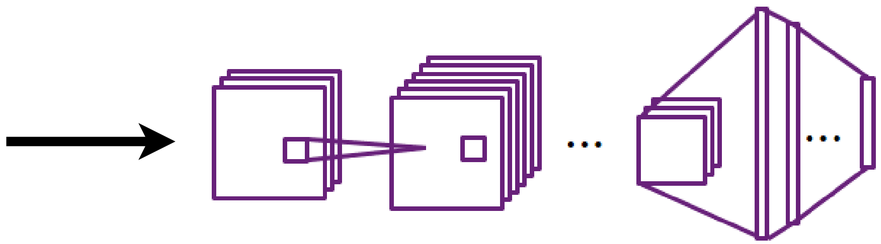
Category	Prediction
Dog	No
Cat	No
Couch	No
Flowers	No
Background	**Yes**
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem



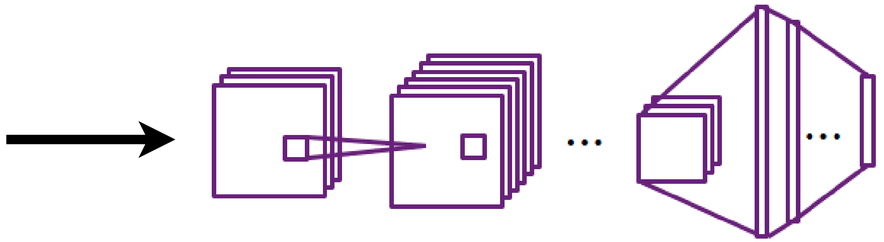


Category	Prediction
Dog	No
Cat	No
Couch	No
Flowers	No
Background	**Yes**
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem

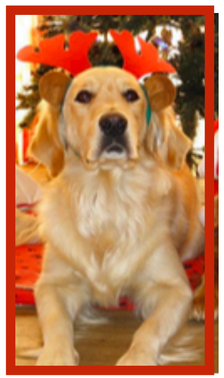


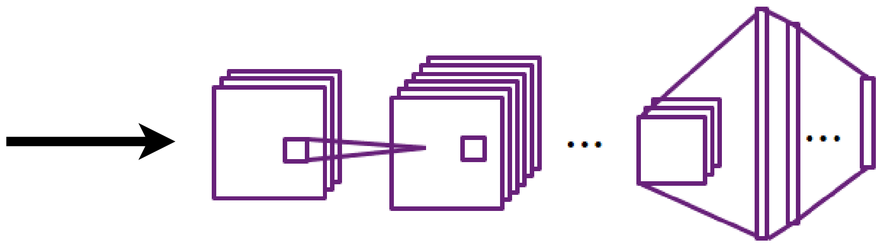
Category	Prediction
Dog	**Yes**
Cat	No
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem



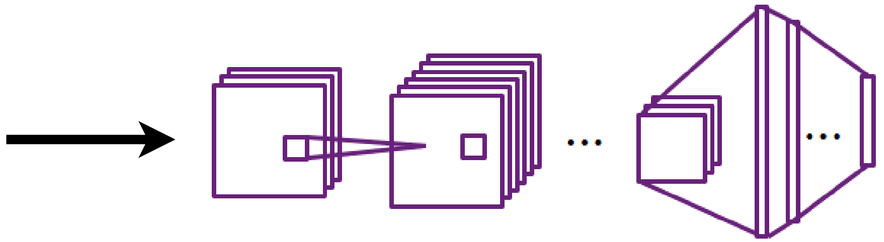


Category	Prediction
Dog	**Yes**
Cat	No
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem



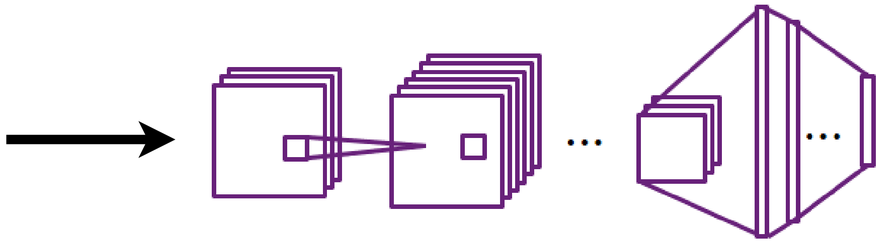
Category	Prediction
Dog	No
Cat	**Yes**
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Object **Detection** as Classification Problem

Problem: Need to apply CNN to **many** patches in each image



Category	Prediction
Dog	No
Cat	**Yes**
Couch	No
Flowers	No
Background	No
…	…

Apply CNN to many different crops in the image and (classification) CNN classifies each patch as object or background

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

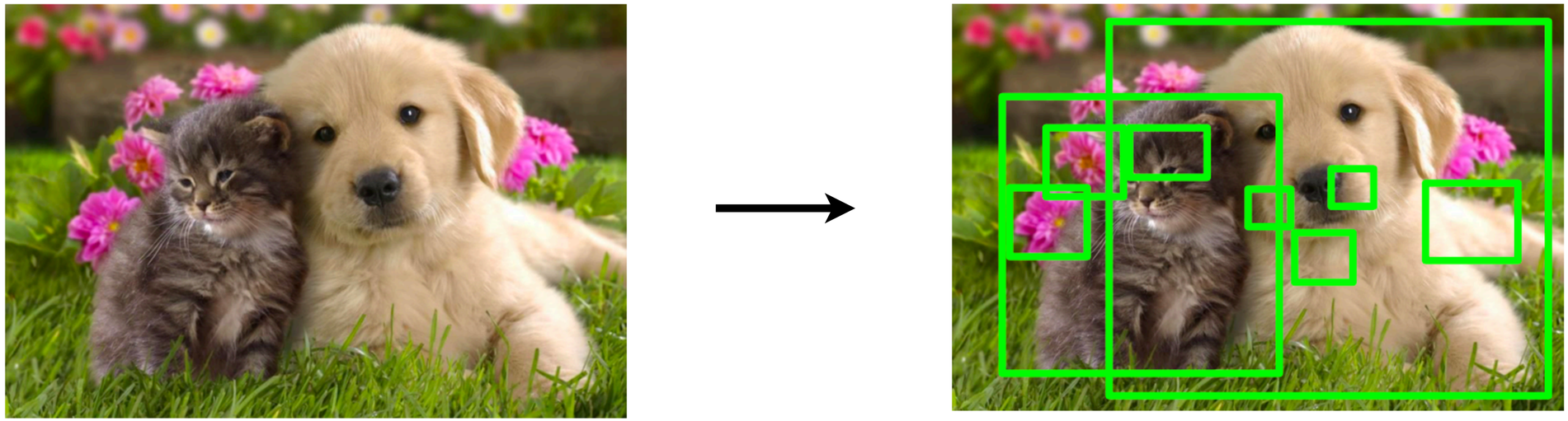
Region Proposals (older idea in vision)

[Alexe et al, TPAMI 2012]
[Uijkings et al, IJCV 2013]
[Cheng et al, CVPR 2014]
[Zitnick and Dollar, ECCV 2014]

Find image **regions that are likely contain objects** (any object at all)

- typically works by looking at histogram distributions, region aspect ratio, closed contours, coherent color

Relatively **fast to run** (Selective Search gives 1000 region proposals in a few seconds on a CPU)



Goal: Get “true” object regions to be in as few top K proposals as possible

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

R-CNN

[Girshick et al, CVPR 2014]

Input **Image**

* image from Ross Girshick

R-CNN

[Girshick et al, CVPR 2014]

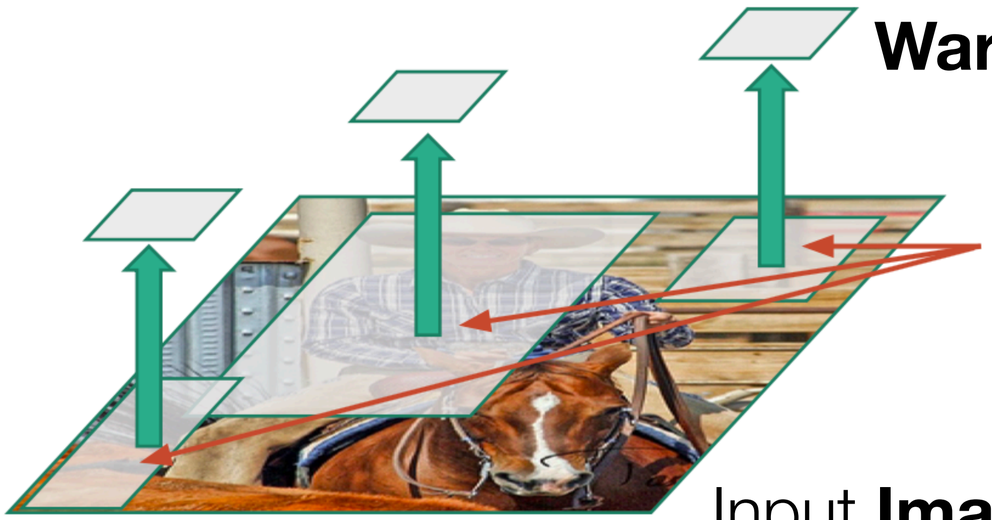
Regions of Interest from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

R-CNN

[Girshick et al, CVPR 2014]



Warped image regions

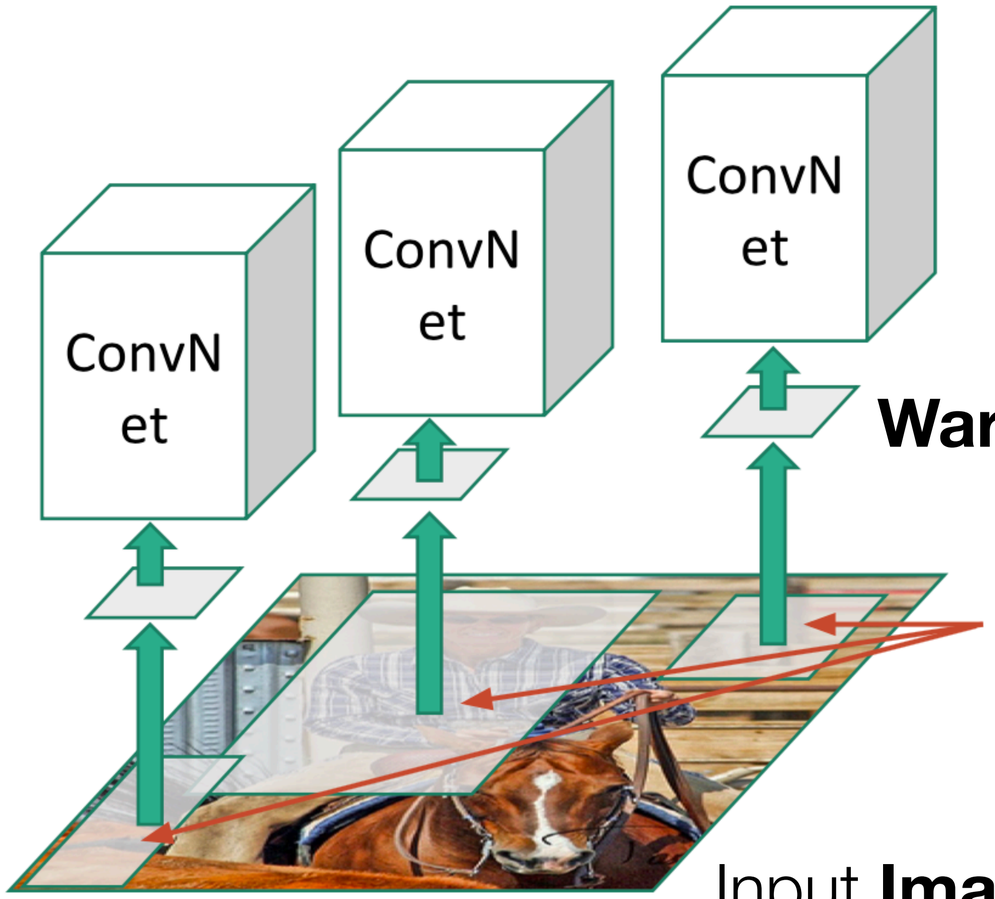
Regions of Interest from
a proposal method (~2k)

Input **Image**

* image from Ross Girshick

R-CNN

[Girshick et al, CVPR 2014]



Forward each region through a **CNN**

Warped image regions

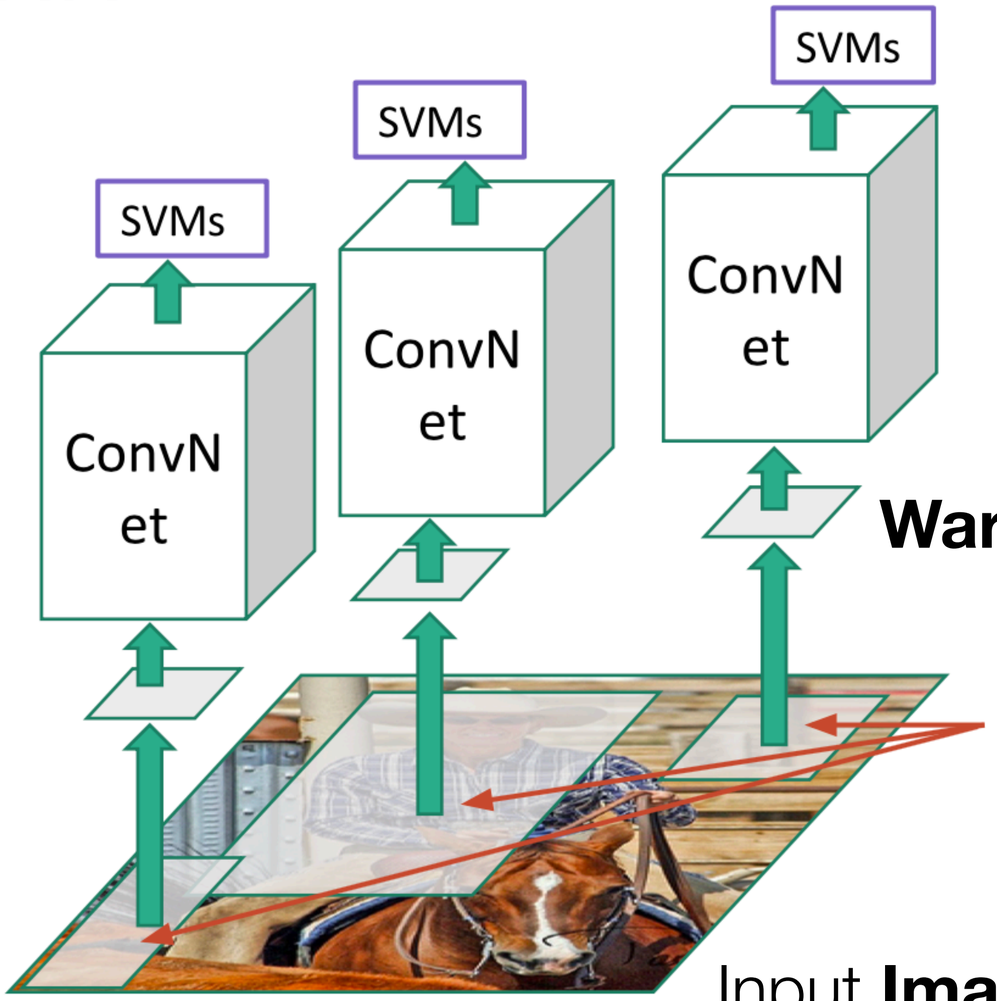
Regions of Interest from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

R-CNN

[Girshick et al, CVPR 2014]



Classify regions with SVM

Forward each region through a **CNN**

Warped image regions

Regions of Interest from a proposal method (~2k)

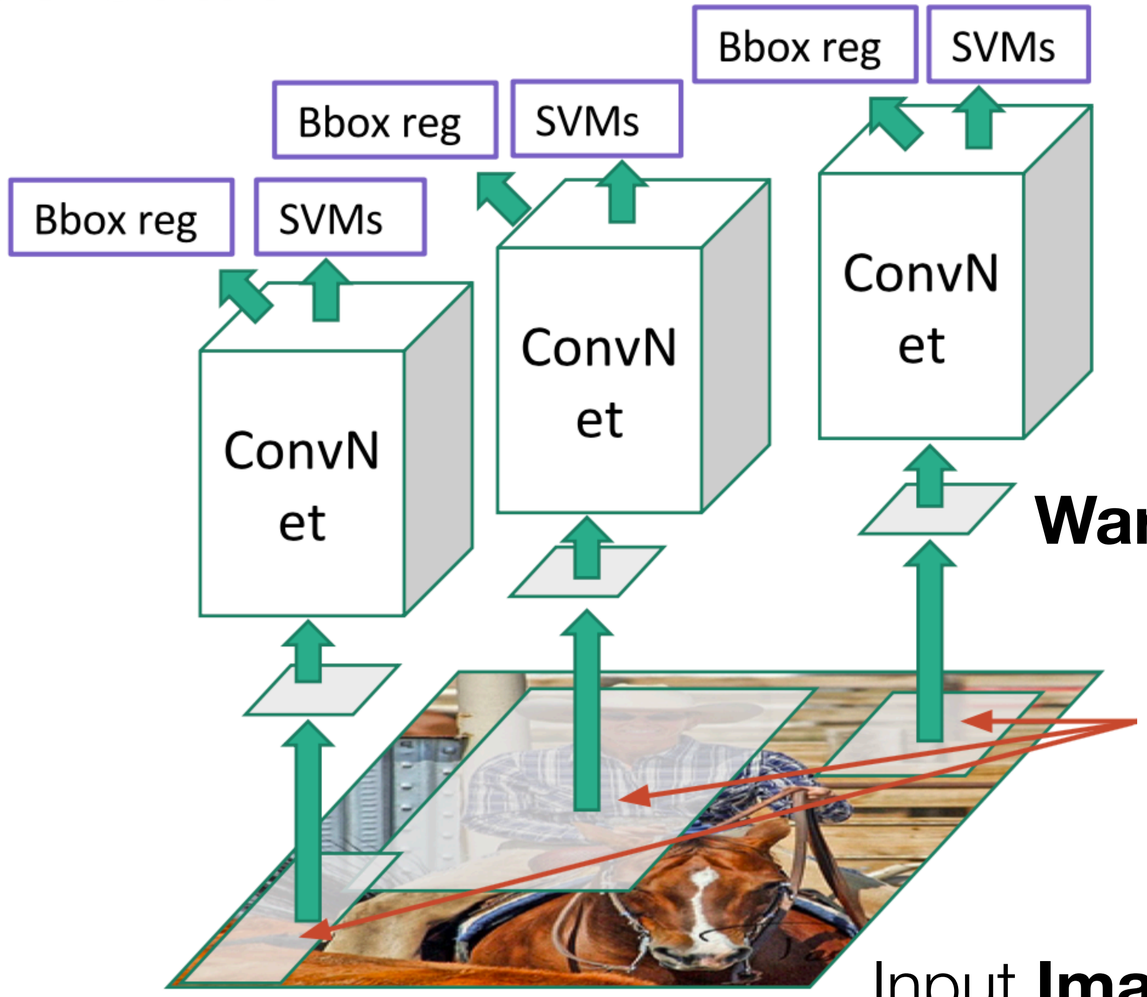
Input **Image**

* image from Ross Girshick

R-CNN

Linear Regression for bounding box offsets

[Girshick et al, CVPR 2014]



Classify regions with SVM

Forward each region through a **CNN**

Warped image regions

Regions of Interest from a proposal method (~2k)

Input **Image**

* image from Ross Girshick

R-CNN

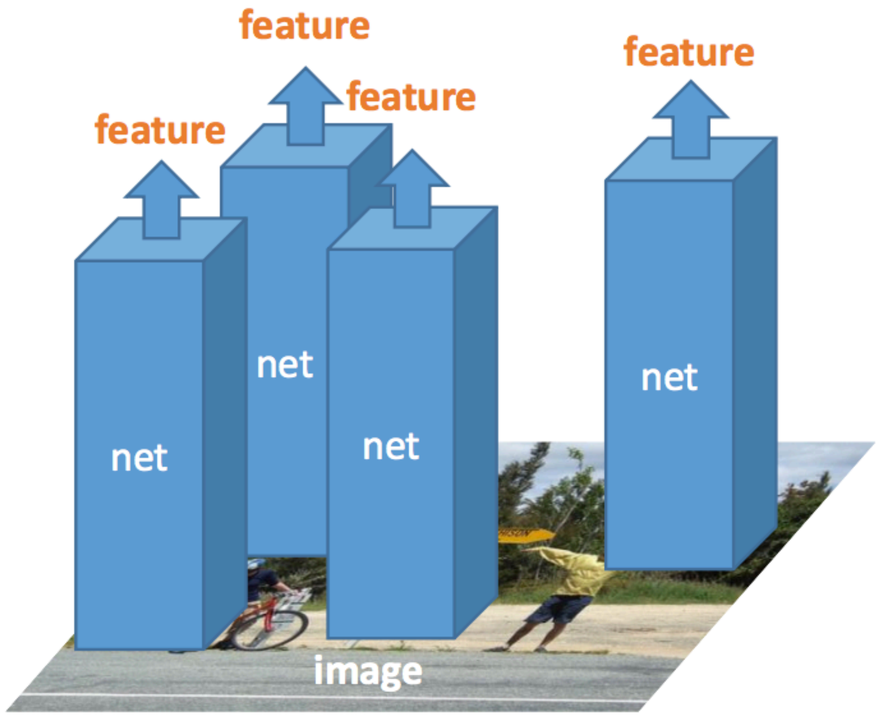
R-CNN (Regions with CNN features) algorithm:

- Extract promising candidate regions using an object proposals algorithm
- Resize each proposal window to the size of the input layer of a trained convolutional neural network
- Input each resized image patch to the convolutional neural network

Implementation detail: Instead of using the classification scores of the network directly, the output of the final fully-connected layer can be used as an input feature to a trained support vector machine (SVM)

R-CNN vs. SPP

[He et al, ECCV 2014]

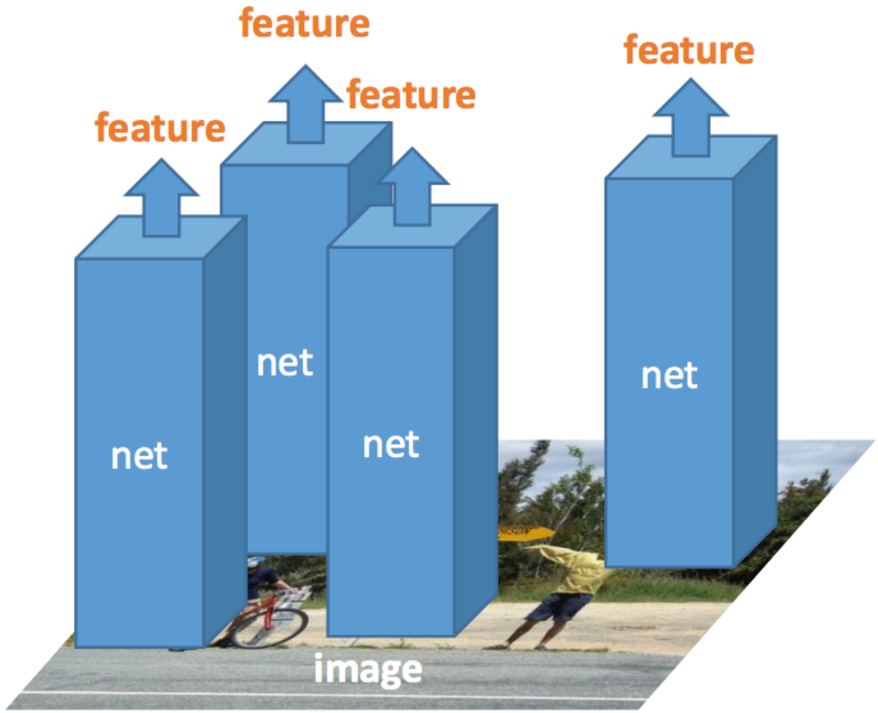


R-CNN

2000 nets on image regions

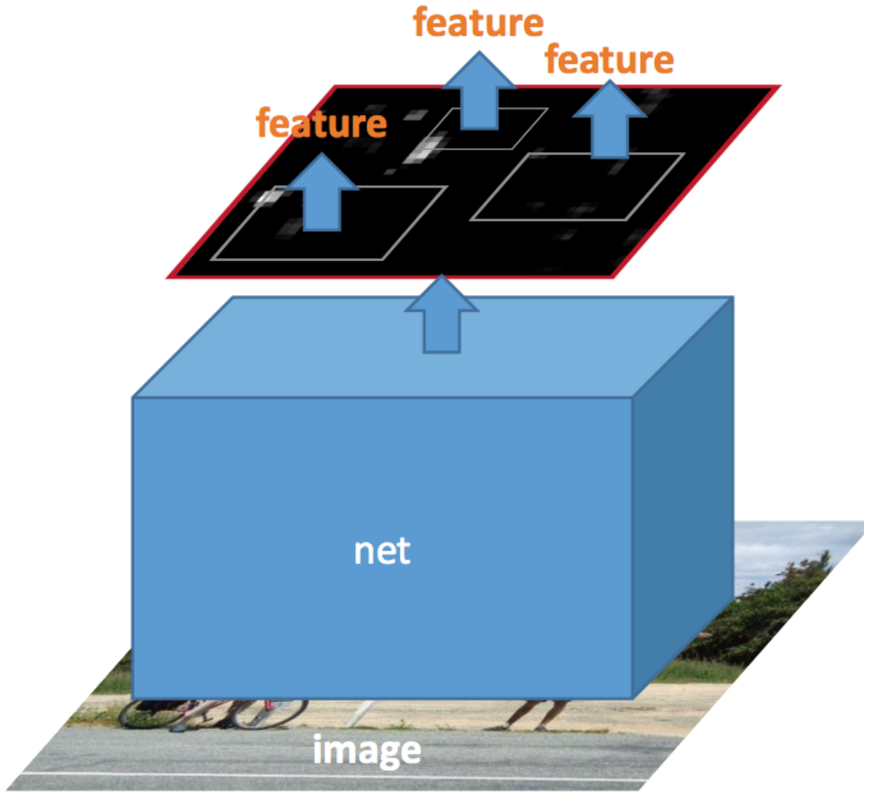
R-CNN vs. SPP

[He et al, ECCV 2014]



R-CNN

2000 nets on image regions



SPP-net

1 net on full image

Fast **R-CNN**

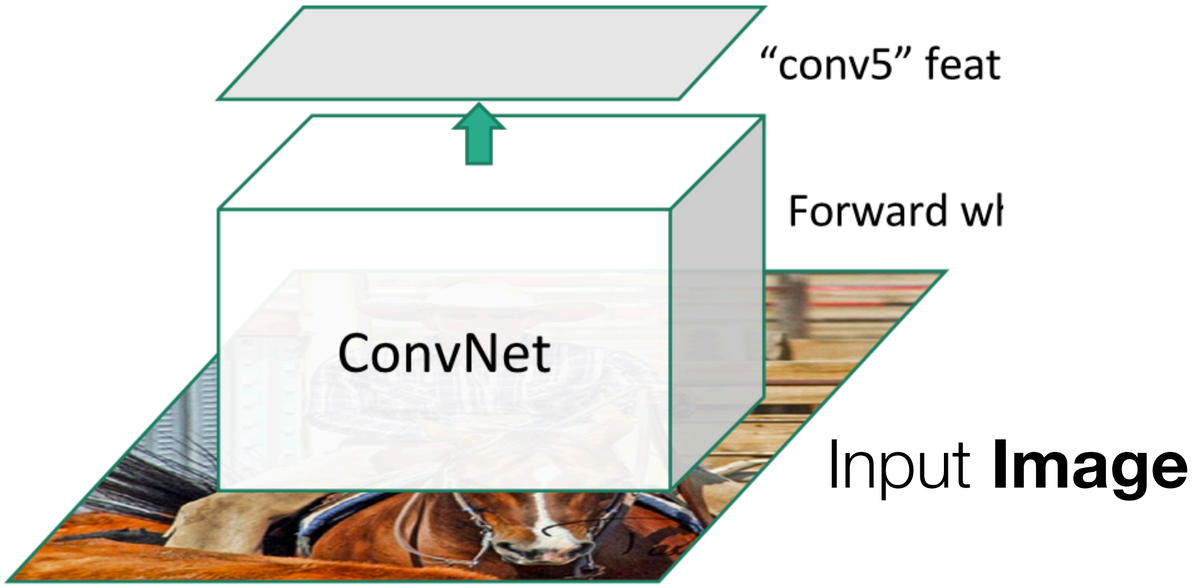
[Girshick et al, ICCV 2015]

Input **Image**

* image from Ross Girshick

Fast **R-CNN**

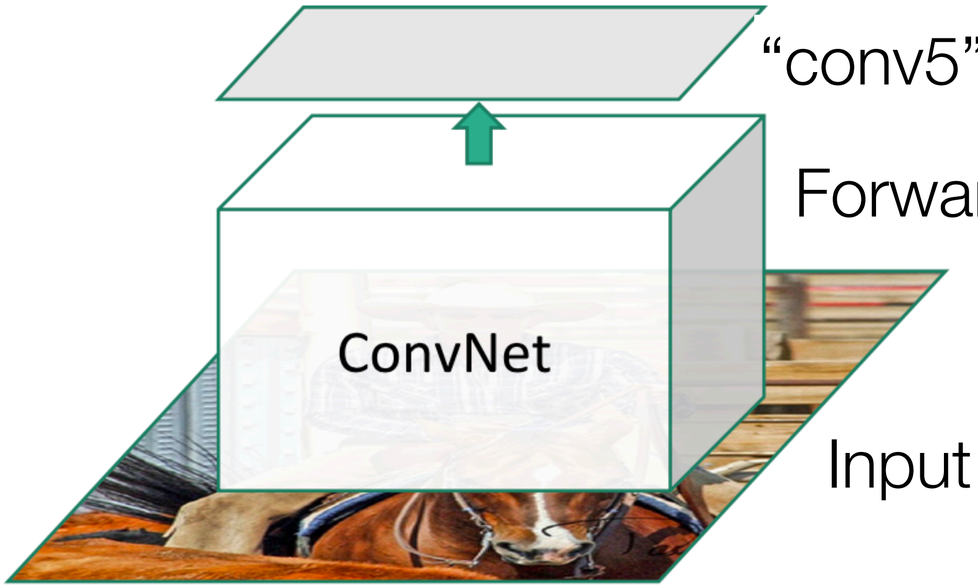
[Girshick et al, ICCV 2015]



* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]



"conv5" feature map

Forward prop the **whole image** through CNN

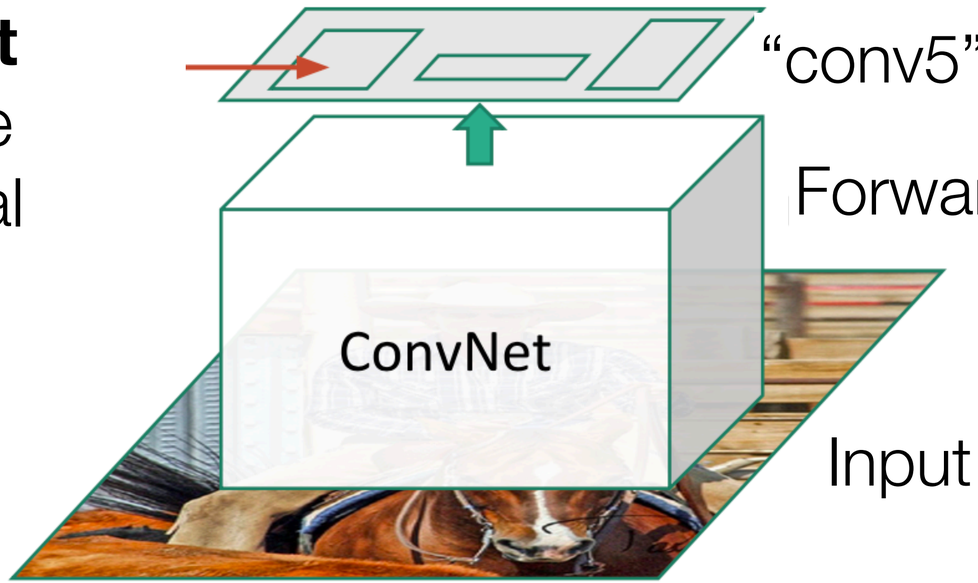
Input **Image**

* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]

Regions of Interest from the proposal method



"conv5" feature map

Forward prop the **whole image** through CNN

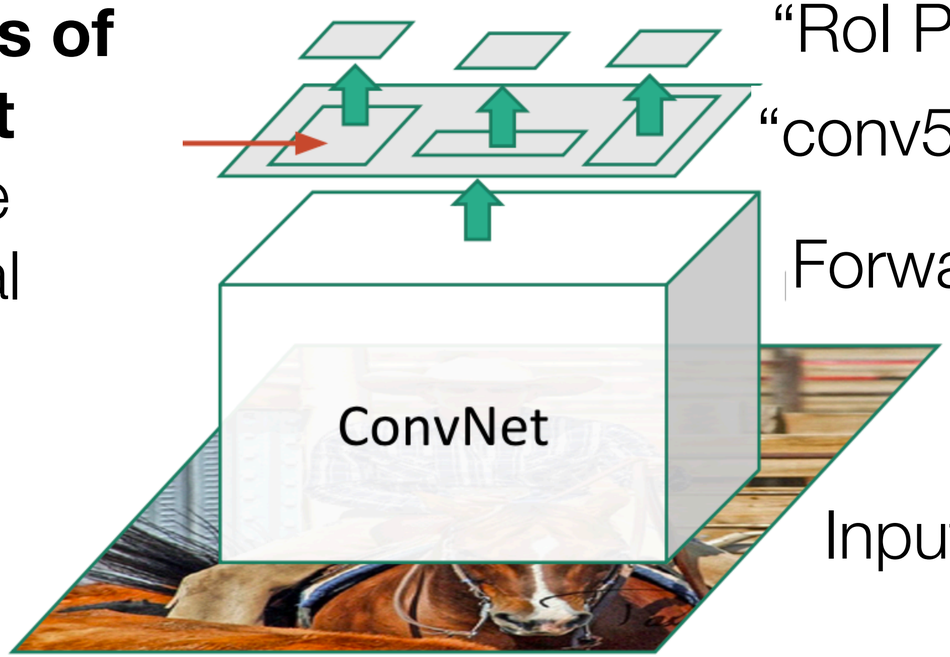
Input **Image**

* image from Ross Girshick

Fast **R-CNN**

[Girshick et al, ICCV 2015]

Regions of Interest from the proposal method



"RoI Pooling" layer

"conv5" feature map

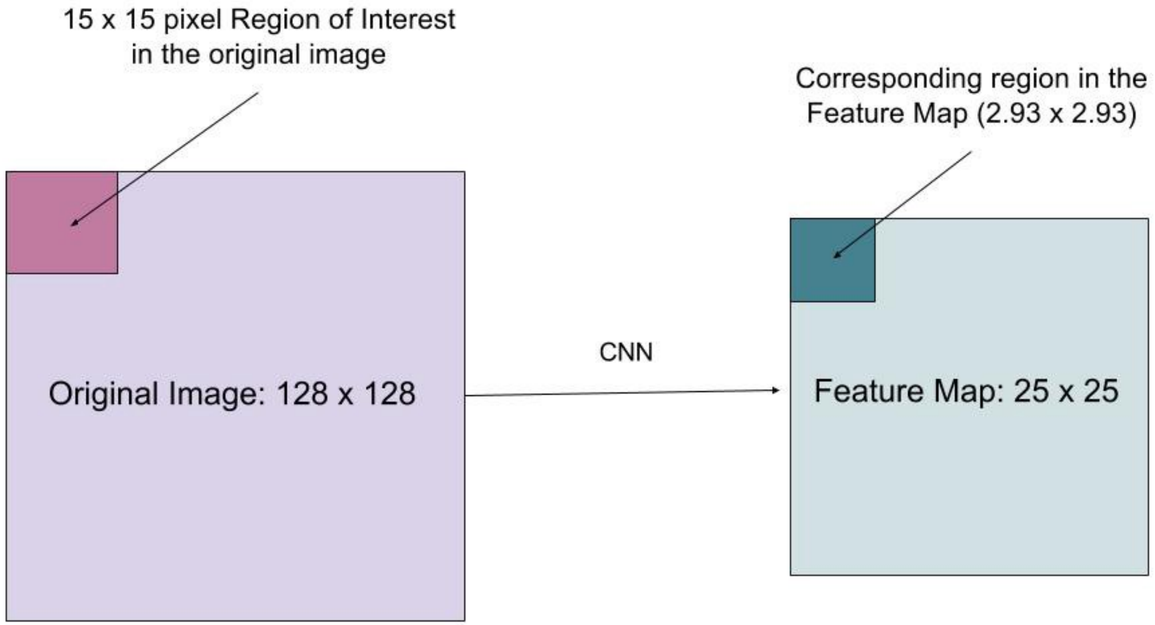
Forward prop the **whole image** through CNN

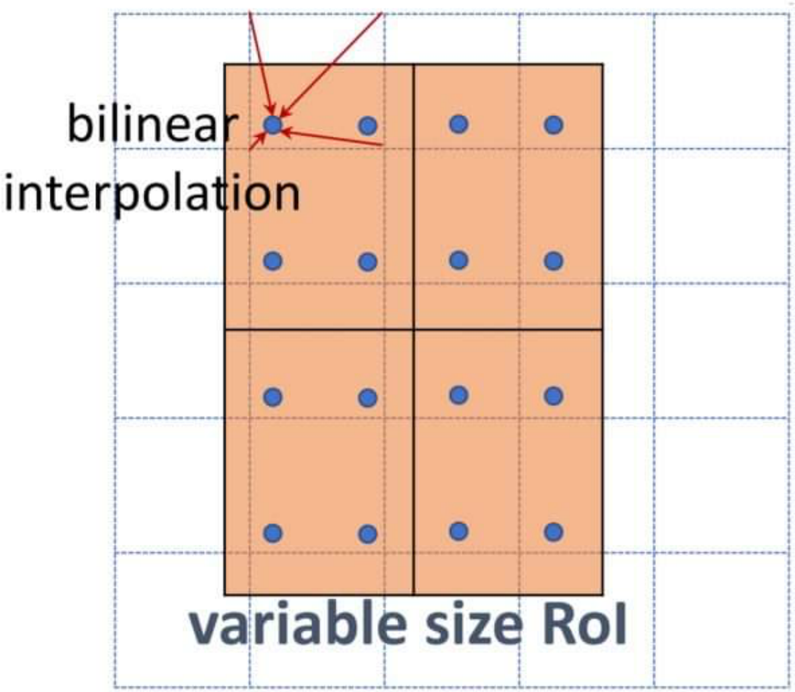
Input **Image**

Girshick, "Fast R-C
Figure copyright R

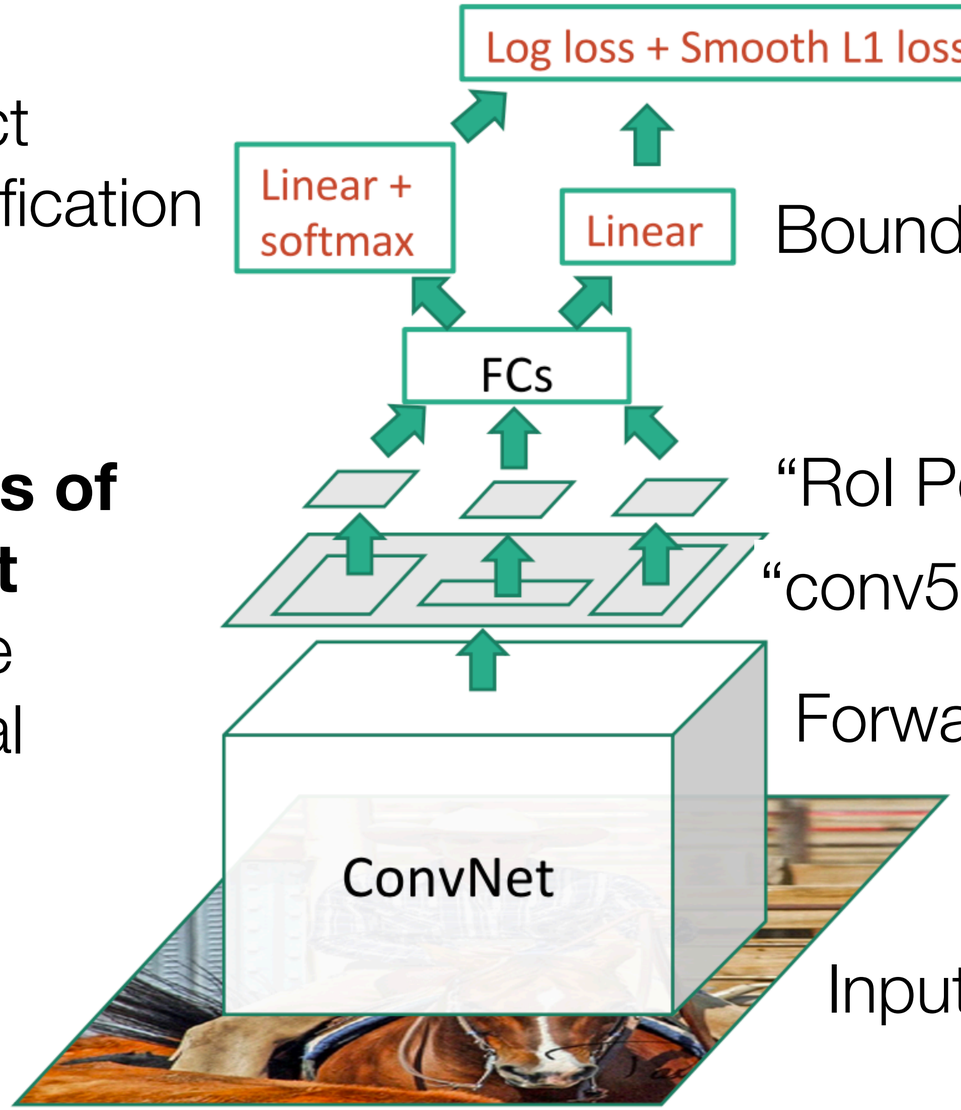
* image from Ross Girshick

RoI Align


15 x 15 pixel Region of Interest
in the original image
Corresponding region in the
Feature Map (2.93 x 2.93)
Original Image: 128 x 128
CNN
Feature Map: 25 x 25


bilinear
interpolation
variable size RoI

Fast **R-CNN**



Multi-task loss

Object classification

Bounding box regression

[Girshick et al, ICCV 2015]

Regions of Interest from the proposal method

"RoI Pooling" layer

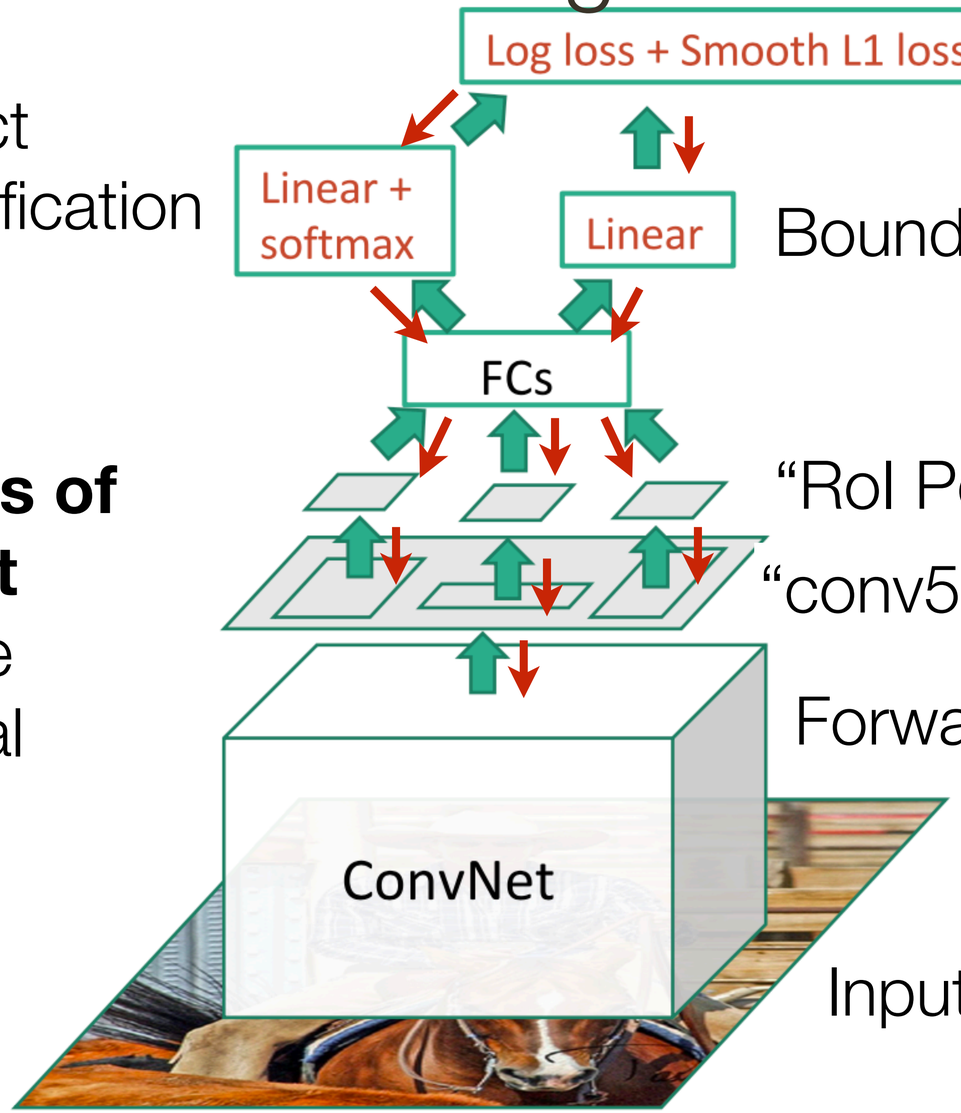
"conv5" feature map

Forward prop the **whole image** through CNN

Input **Image**

* image from Ross Girshick

Fast **R-CNN:** Training



Multi-task loss

[Girshick et al, ICCV 2015]

Object classification

Bounding box regression

Regions of Interest from the proposal method

"RoI Pooling" layer

"conv5" feature map

Forward prop the **whole image** through CNN

Input **Image**

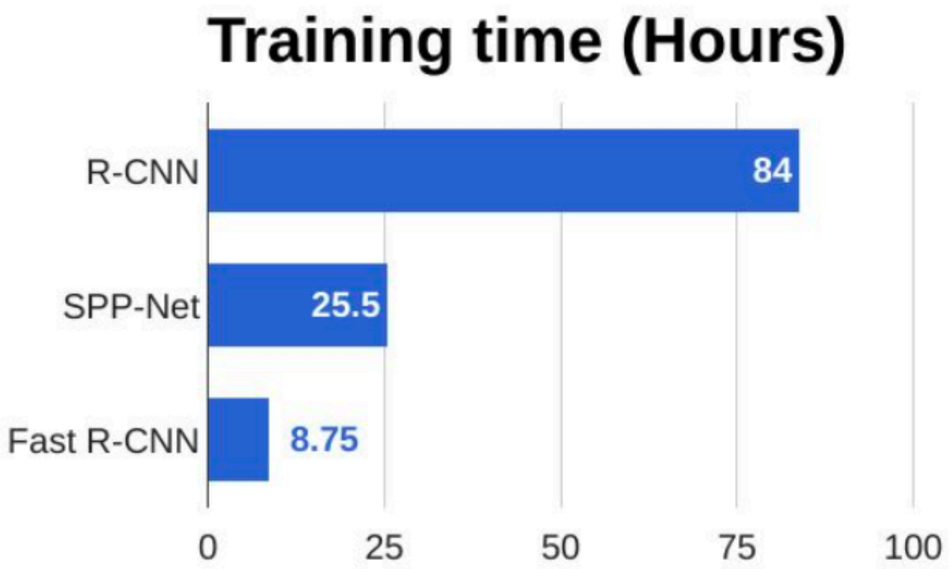
* image from Ross Girshick

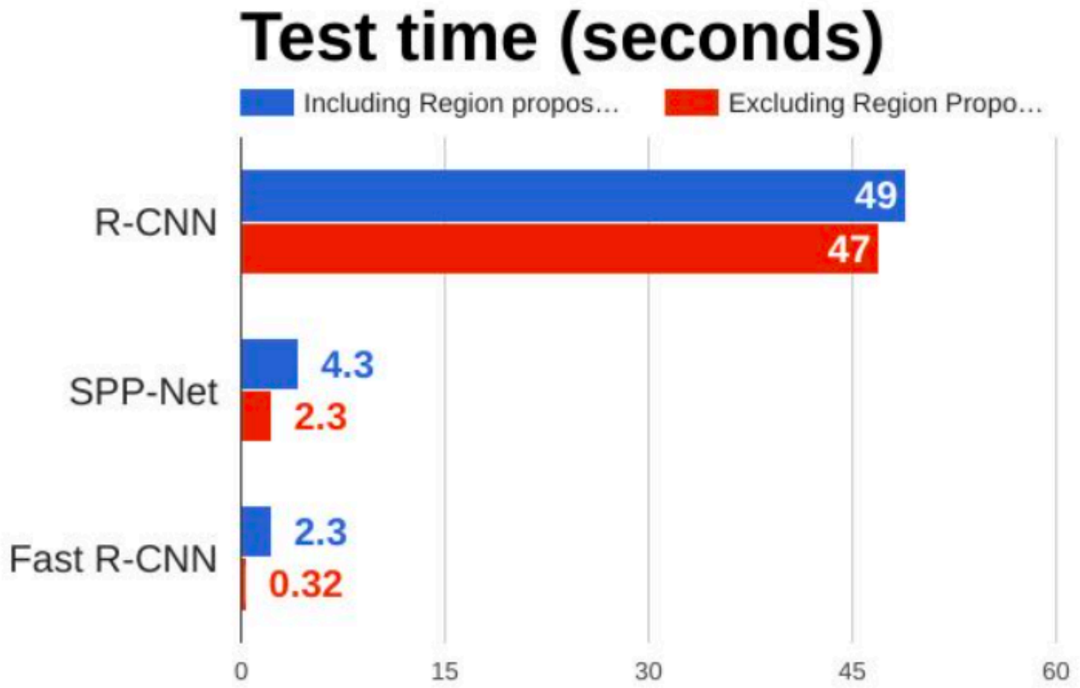
R-CNN vs. SPP vs. Fast R-CNN

[Girshick et al, CVPR 2014]

[Girshick et al, ICCV 2015]

[He et al, ECCV 2014]

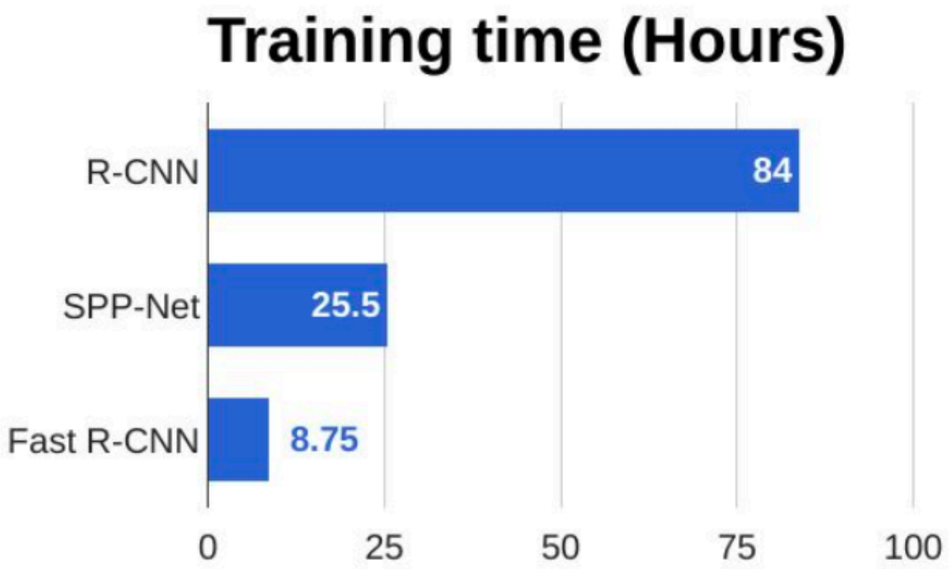


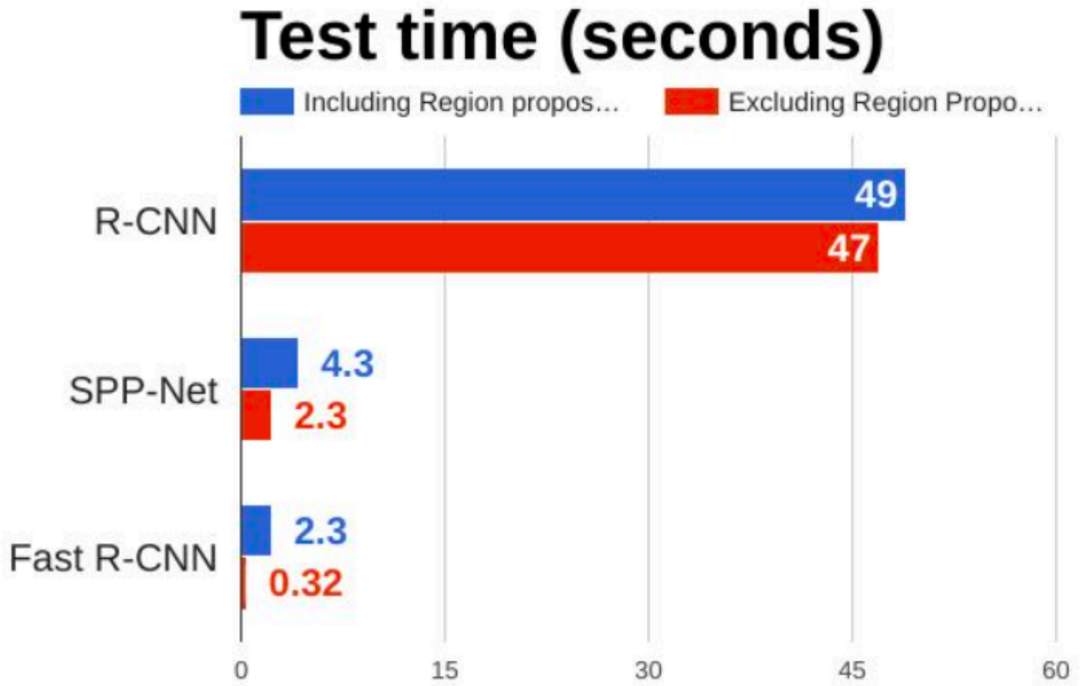


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

R-CNN vs. SPP vs. Fast R-CNN

[Girshick et al, CVPR 2014]
[Girshick et al, ICCV 2015]
[He et al, ECCV 2014]





Observation: Performance dominated by the region proposals at this point!

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

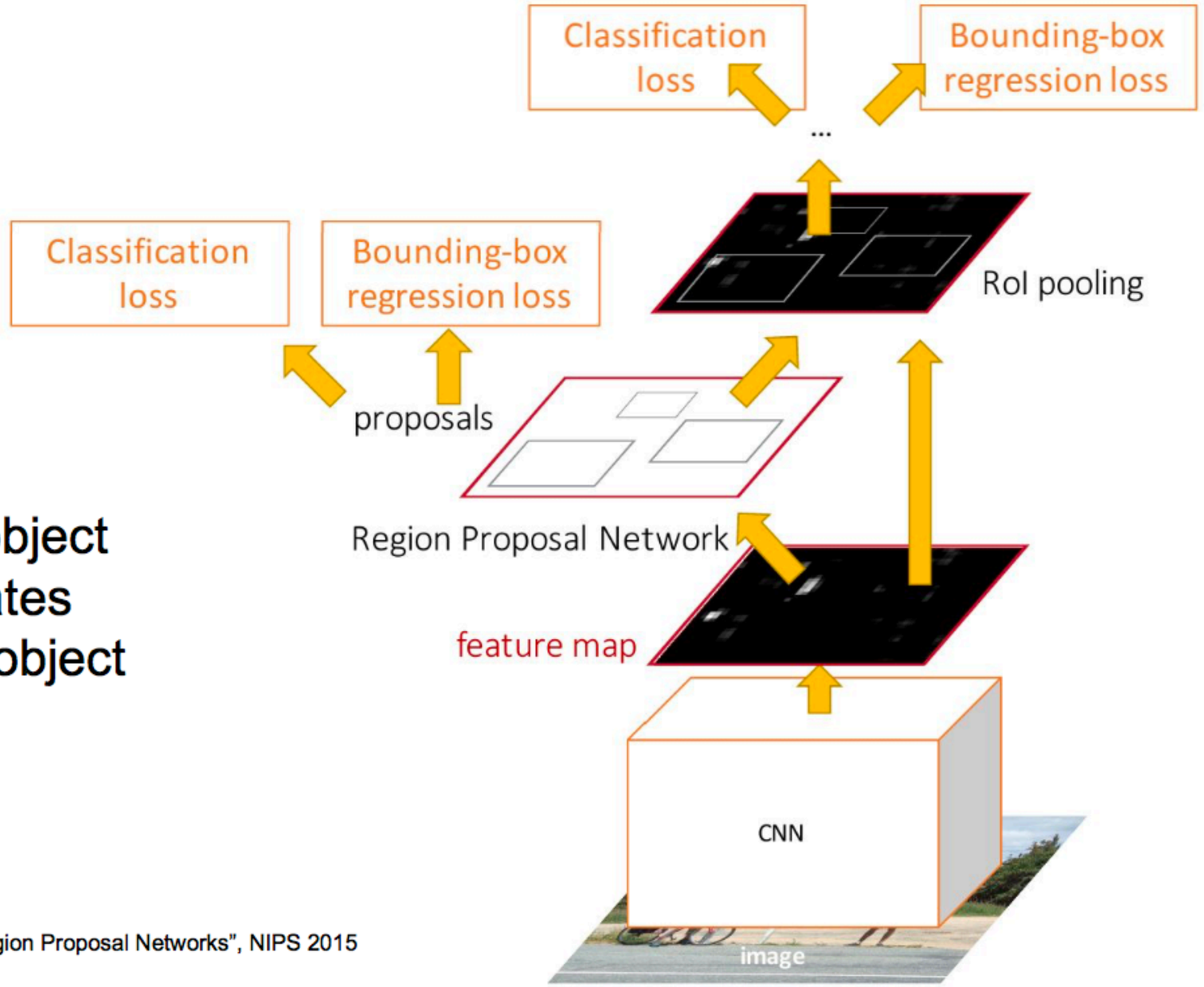
Faster **R-CNN**

Make CNN do proposals!

Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



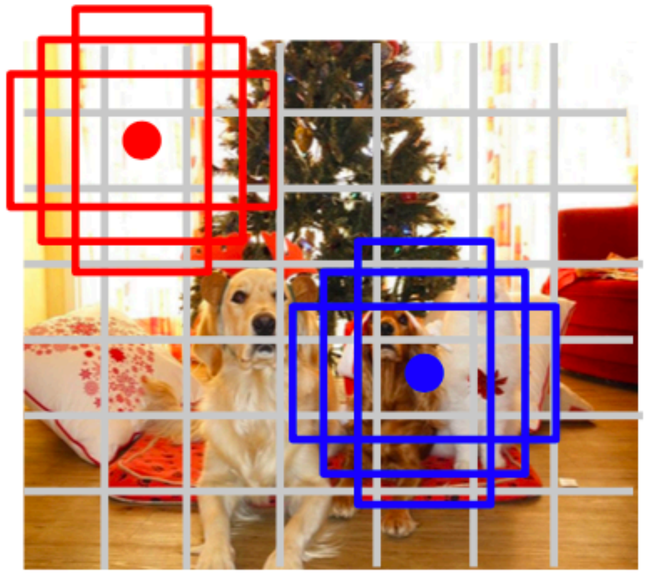
Ren et al, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", NIPS 2015
Figure copyright 2015, Ross Girshick; reproduced with permission

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

YOLO: You Only Look Once

[Redmon et al, CVPR 2016]

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes**
centered at each grid cell
Here B = 3

Within each grid cell:

- Regress from each of the B base boxes to a final box with 5 numbers:
  (dx, dy, dh, dw, confidence)
- Predict scores for each of C classes (including background as a class)

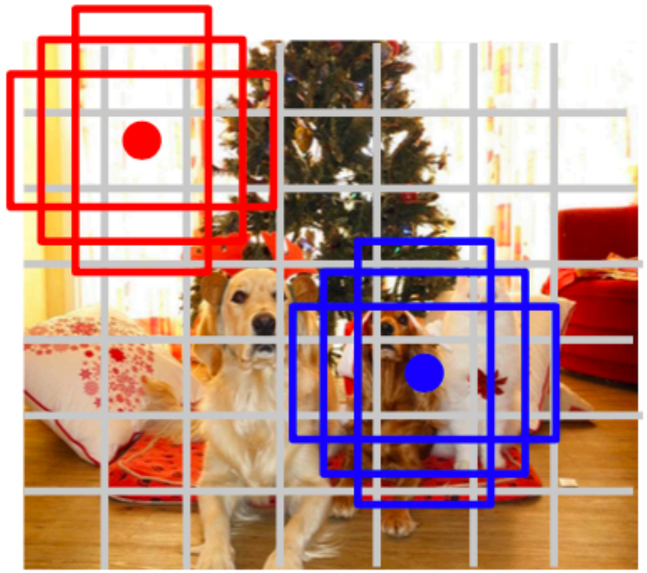
Output:
7 x 7 x (5 * B + C)

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

YOLO: You Only Look Once

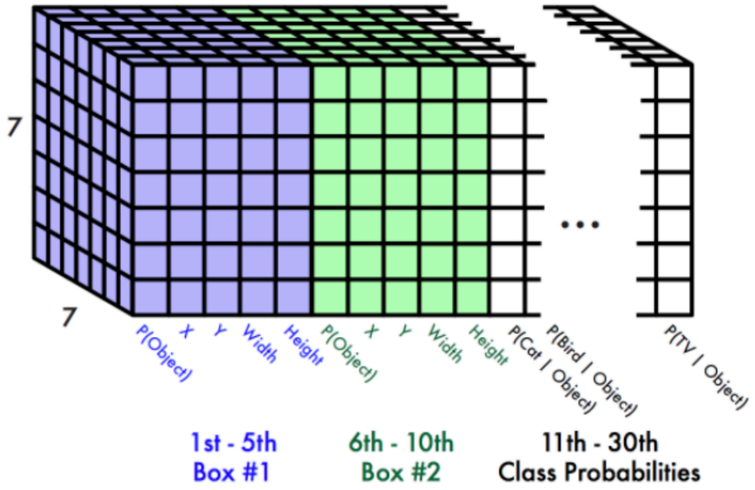
[Redmon et al, CVPR 2016]

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes**
centered at each grid cell
Here B = 3



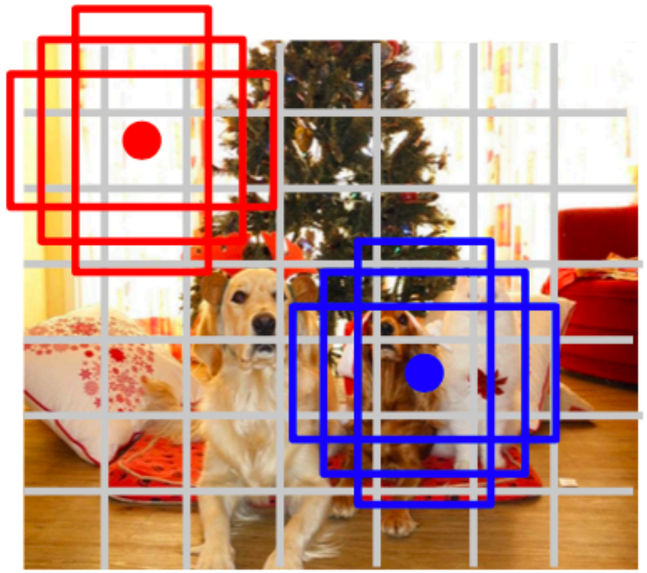

YOLO v2
http://pjreddie.com/yolo


YOLO v2
http://pjreddie.com/yolo

YOLO: You Only Look Once

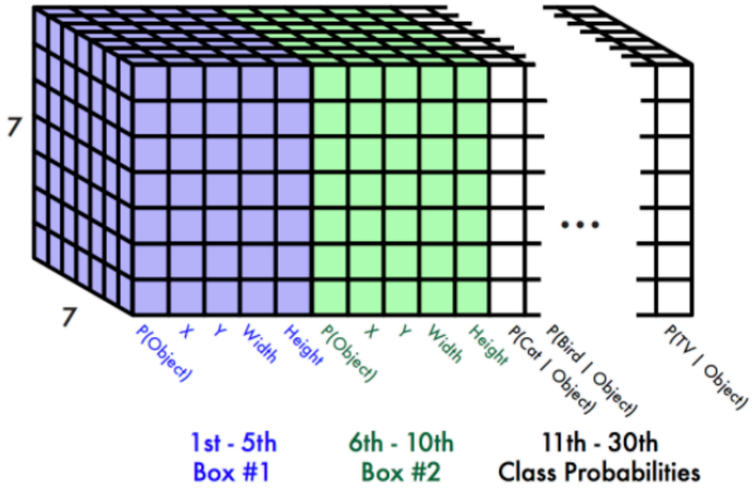
[Redmon et al, CVPR 2016]

Input image
3 x H x W



Divide image into grid
7 x 7

Image a set of **base boxes**
centered at each grid cell
Here B = 3



Computer **Vision Problems** (no language for now)

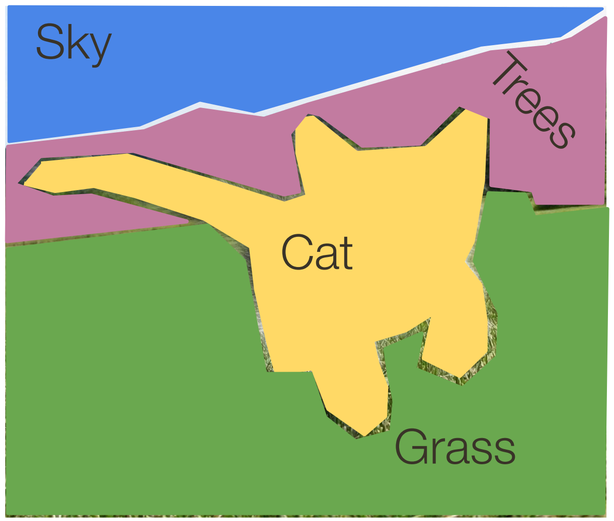
Segmentation

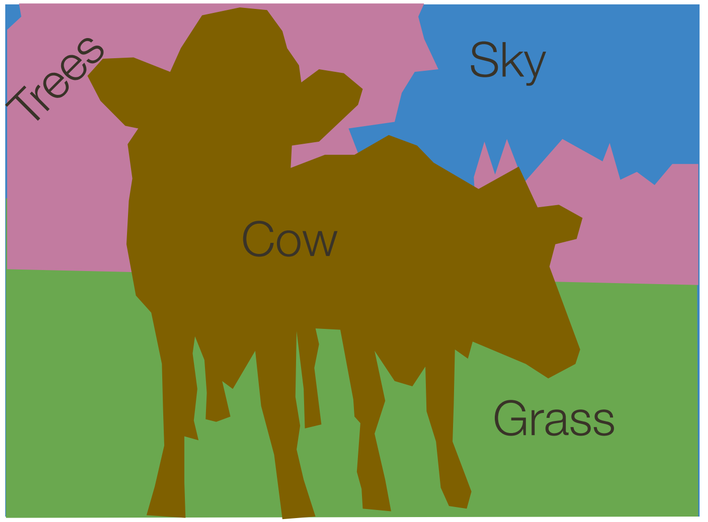
Horse
Person

COCO
Common Objects in Context

Semantic **Segmentation**

Label **every pixel** with a category label (without differentiating instances)



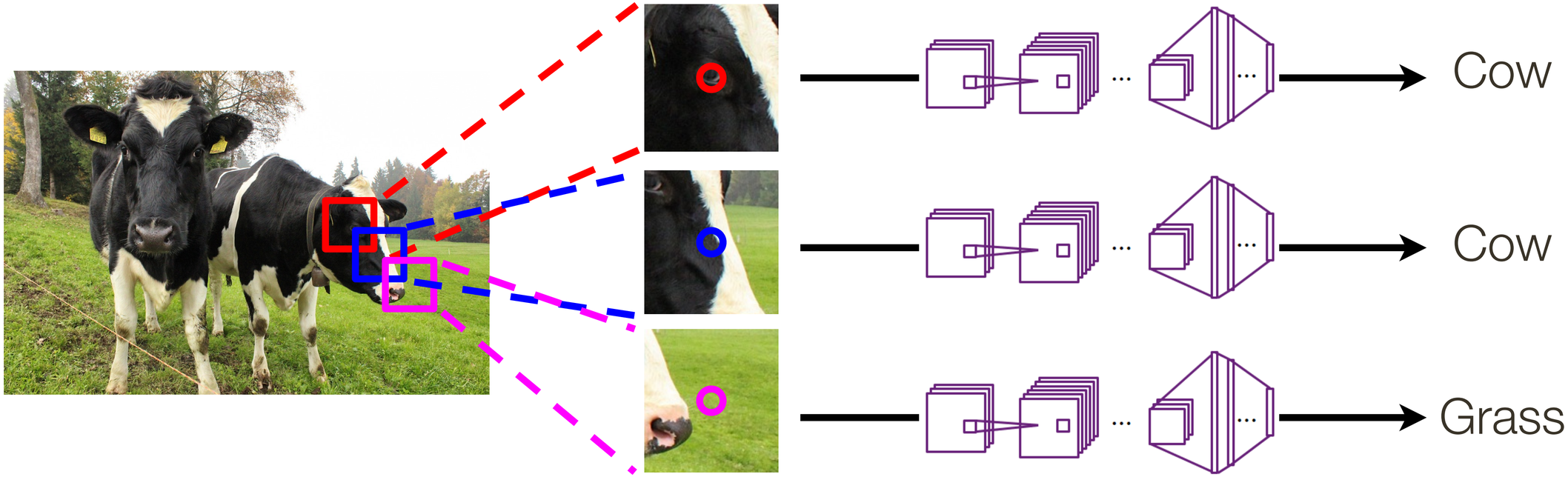


* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Sliding Window

[Farabet et al, TPAMI 2013]
[Pinheiro et al, ICML 2014]

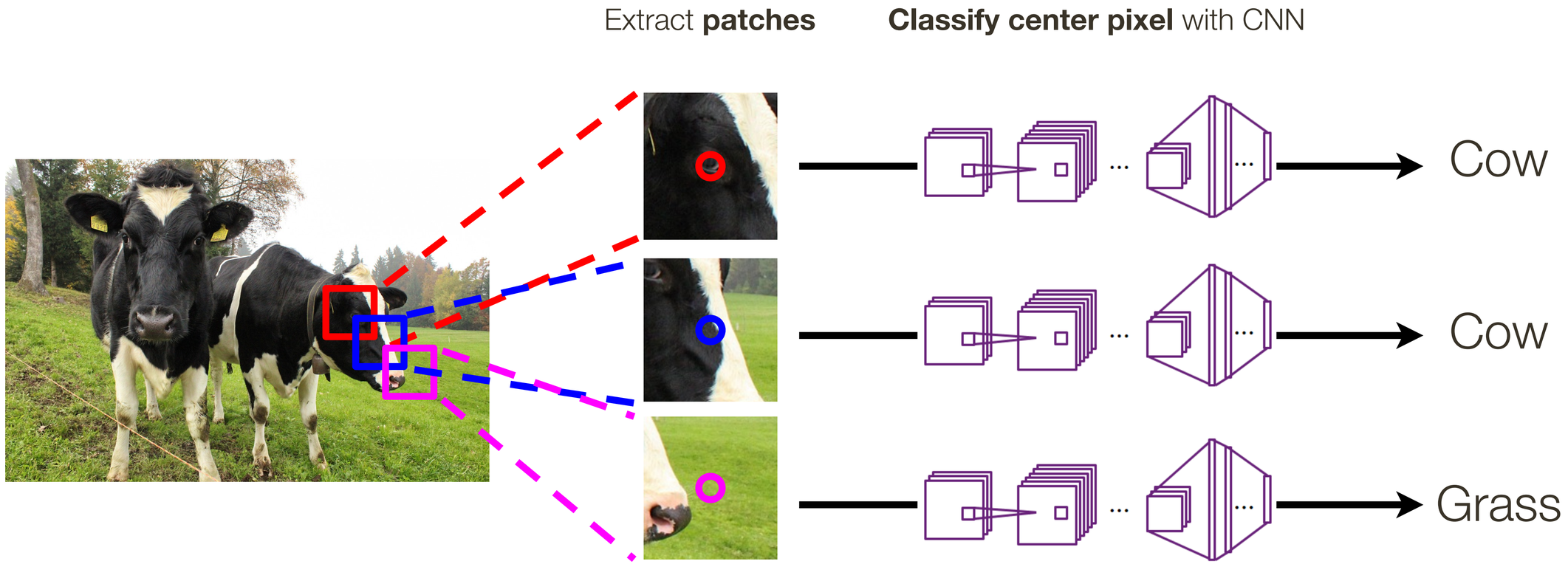
Extract **patches** **Classify center pixel** with CNN



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Sliding Window

[Farabet et al, TPAMI 2013]
[Pinheiro et al, ICML 2014]

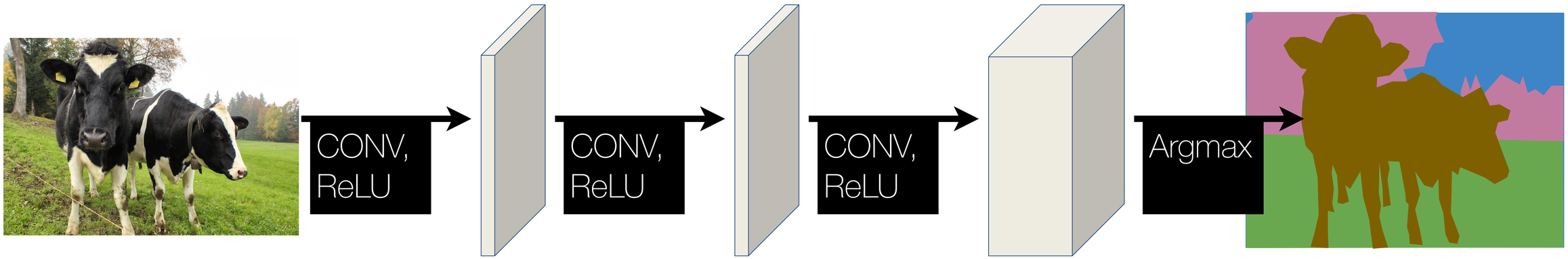


Problem: VERY inefficient, no reuse of computations for overlapping patches

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

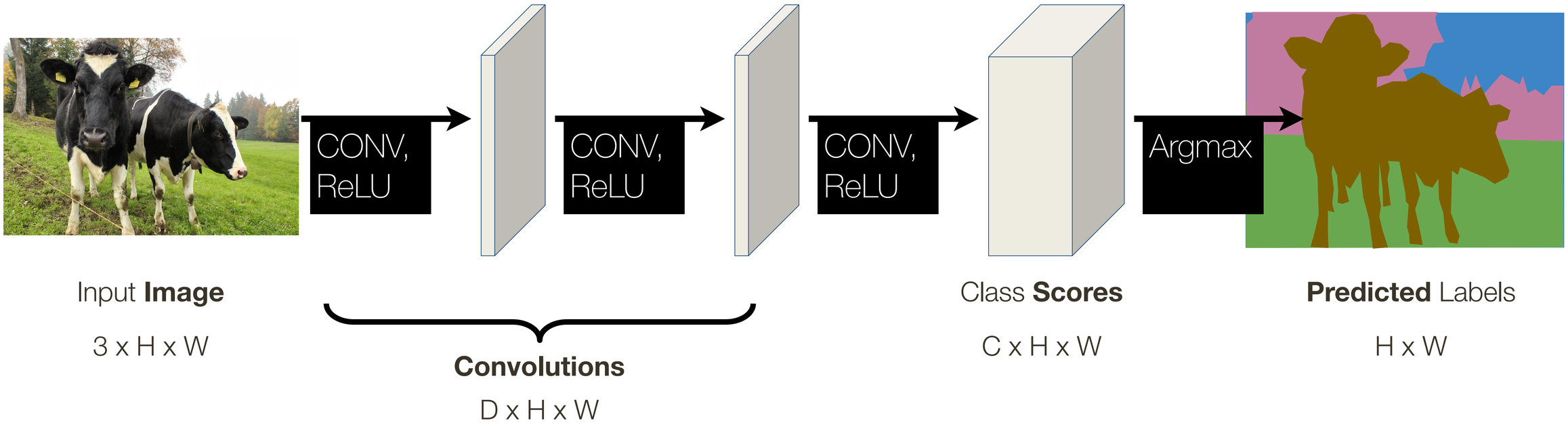
Design a network as a number of convolutional layers to make predictions for all pixels at once!



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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



Problem: Convolutions at the original image scale will be very expensive (in terms of storage)

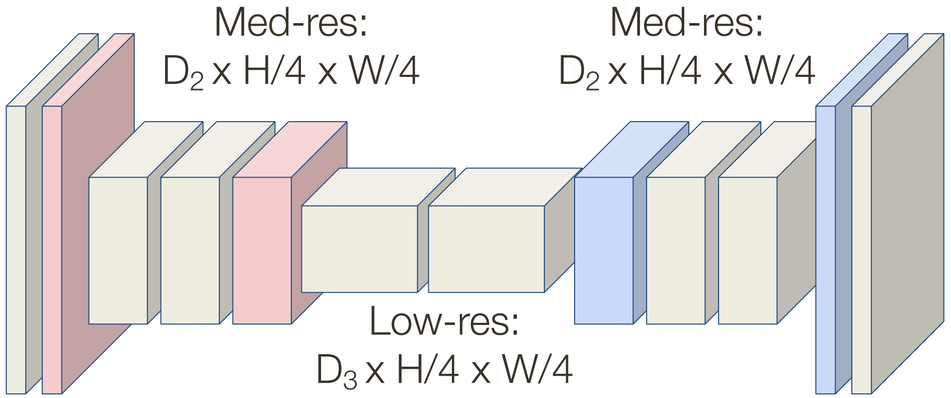
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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

Input **Image**

3 x H x W



Predicted Labels

H x W

[Long et al, CVPR 2015]
[Noh et al, ICCV 2015]

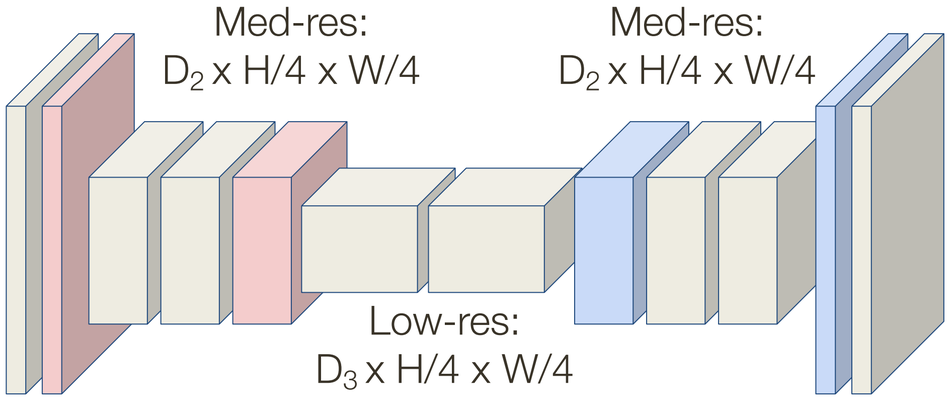
* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Semantic **Segmentation:** Fully Convolutional CNNs

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

Input **Image**

3 x H x W



Predicted Labels

H x W

Downsampling = Pooling

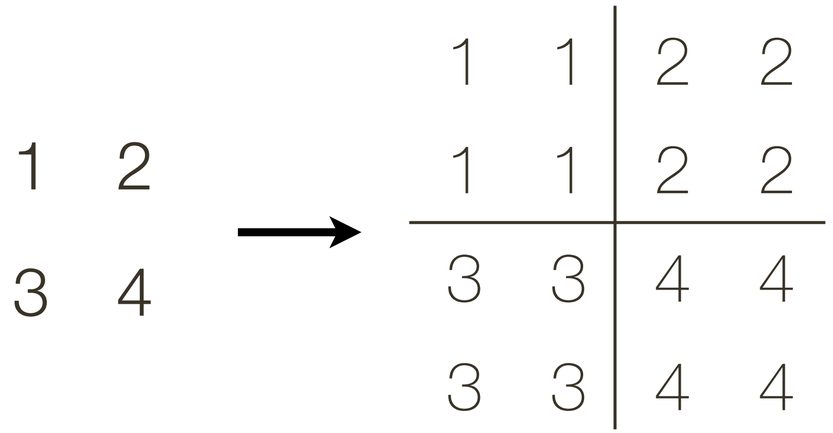
Upsampling = ???

[Long et al, CVPR 2015]
[Noh et al, ICCV 2015]

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling** (a.k.a “Unpooling”)

Nearest Neighbor



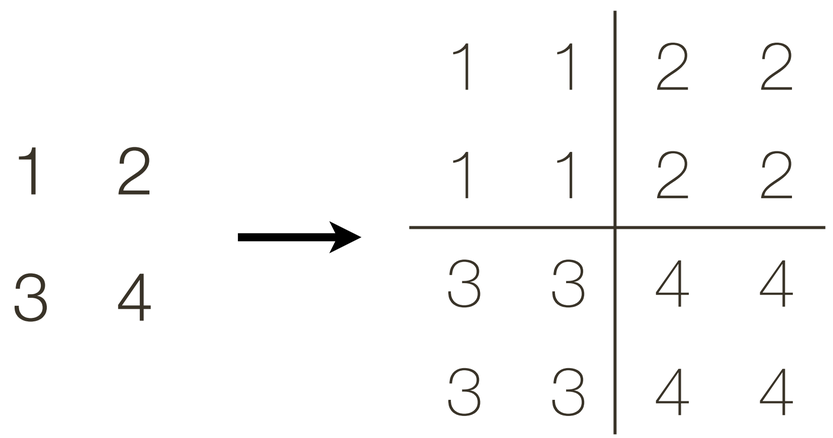
Input: 2 x 2

Output: 4 x 4

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling** (a.k.a “Unpooling”)

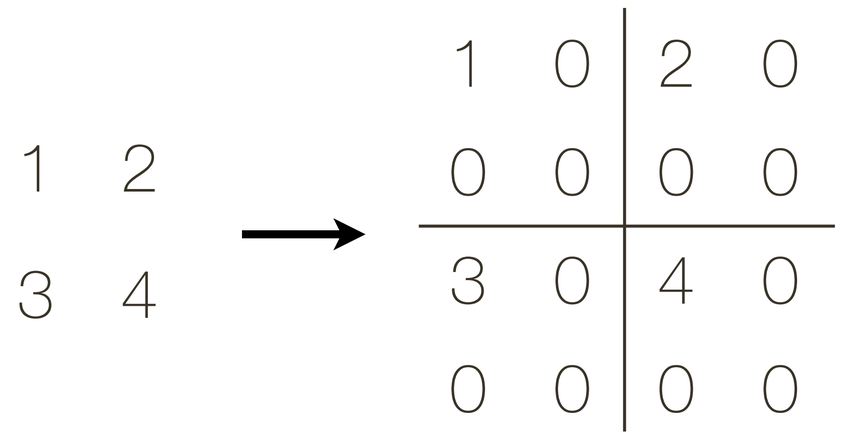
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”



Input: 2 x 2

Output: 4 x 4

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** 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

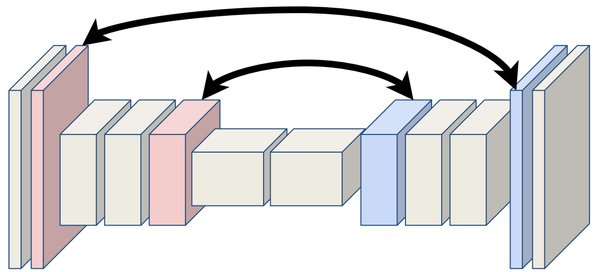
Max Unpooling
Use positions from pooling layer

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

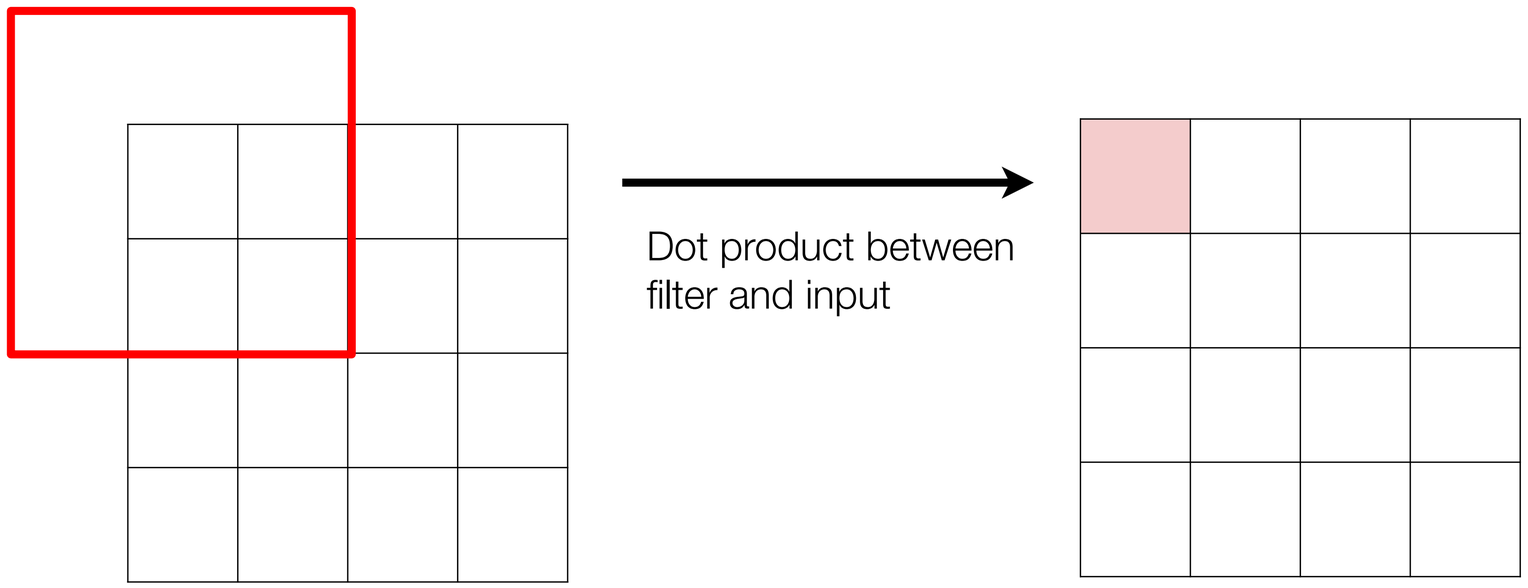


Corresponding pairs of downsampling and upsampling layers

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



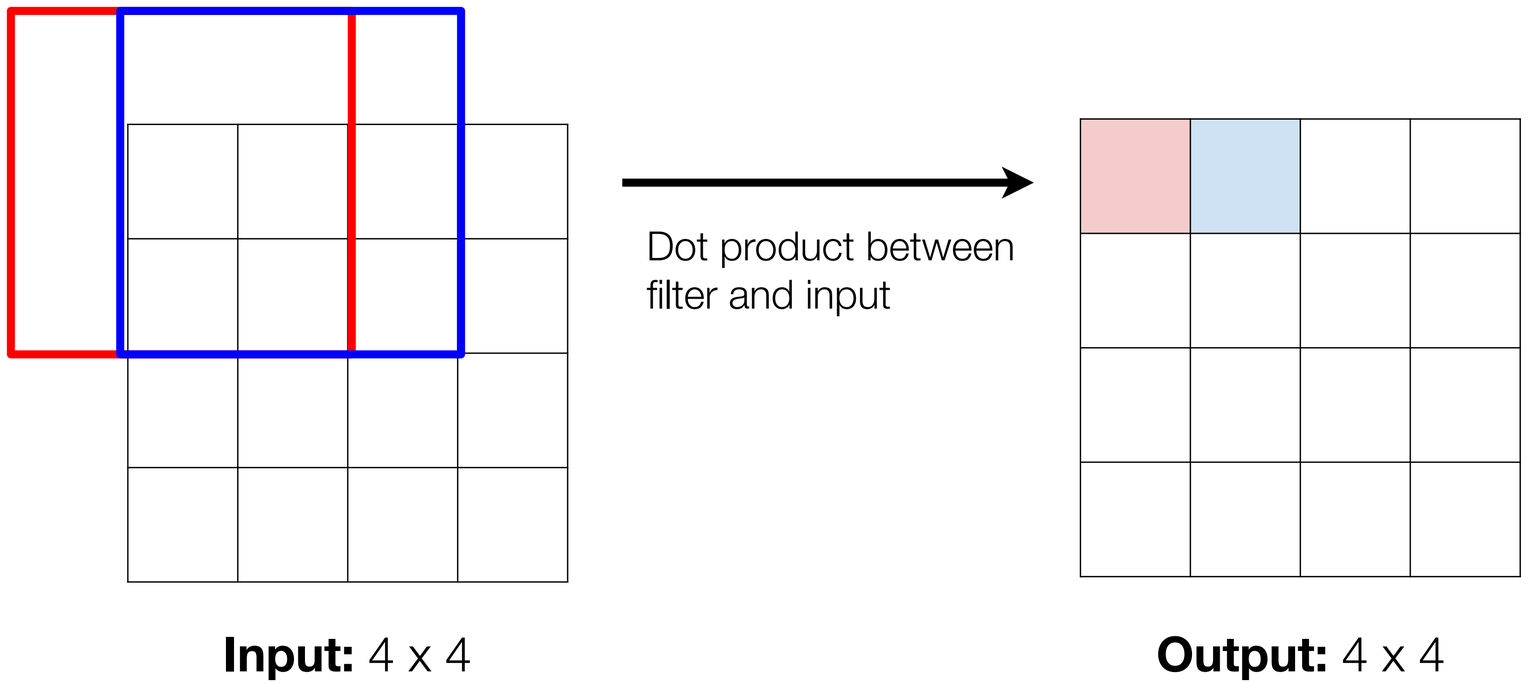
Input: 4 x 4

Output: 4 x 4

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



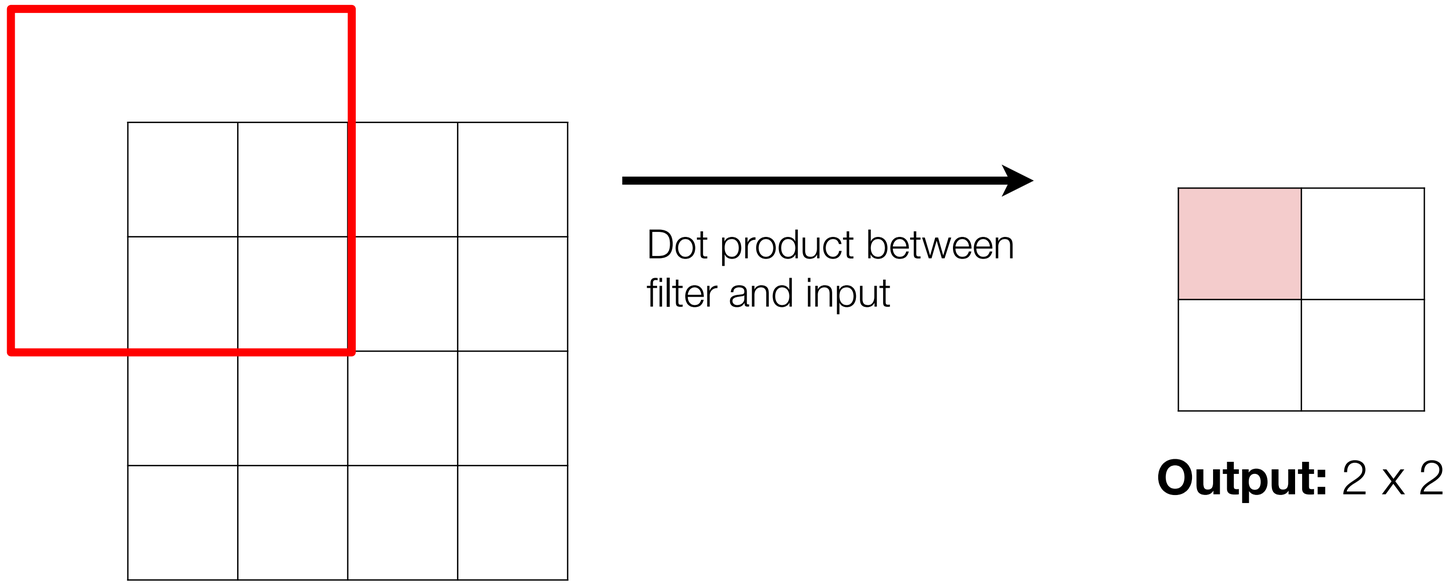
Input: 4 x 4

Output: 4 x 4

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



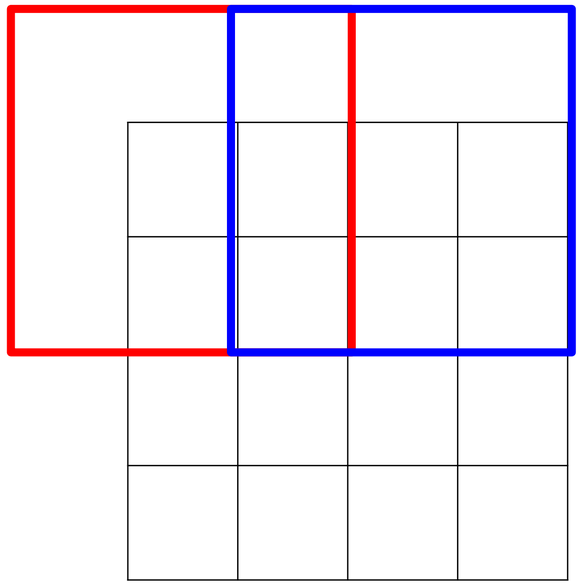
Input: 4 x 4

Output: 2 x 2

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

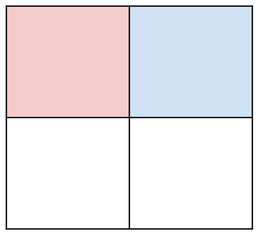
In-network **Up Sampling:** Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input



Output: 2 x 2

Filter moves 2 pixels in the **input** for every one pixel in the **output**

Stride gives ratio in movement in input vs output

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** Transpose Convolution

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

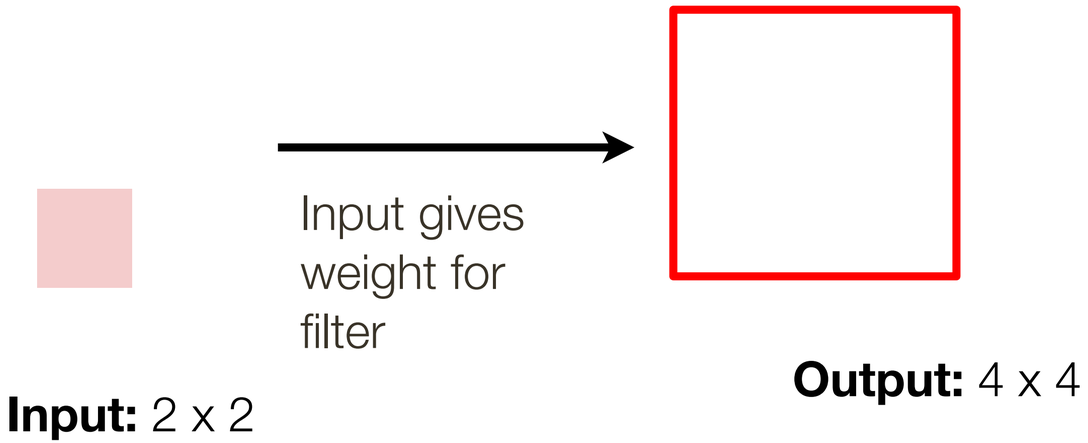
Input: 2 x 2

Output: 4 x 4

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** Transpose Convolution

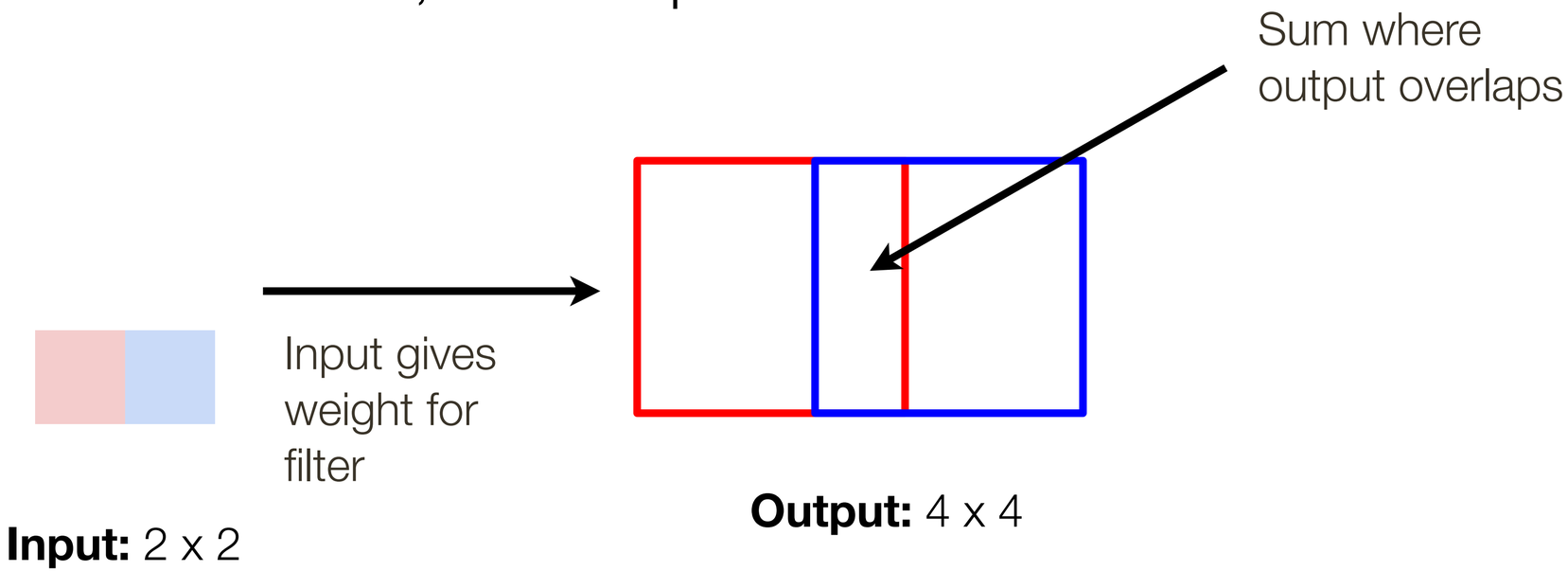
3 x 3 **transpose** convolution, stride 2 pad 1



* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

In-network **Up Sampling:** Transpose Convolution

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

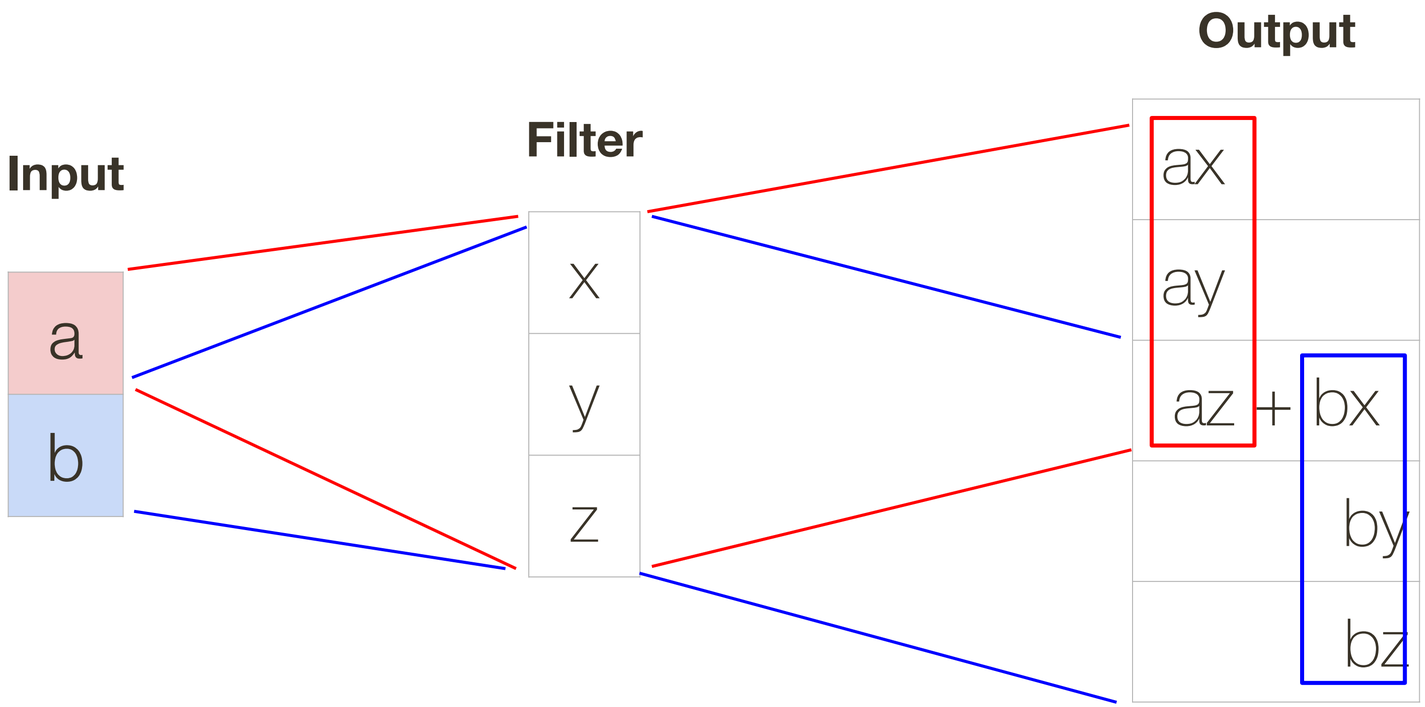


Filter moves 2 pixels in the **output** for every one pixel in the **input**

Stride gives ratio in movement in output vs input

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

Transpose Convolution: 1-D Example

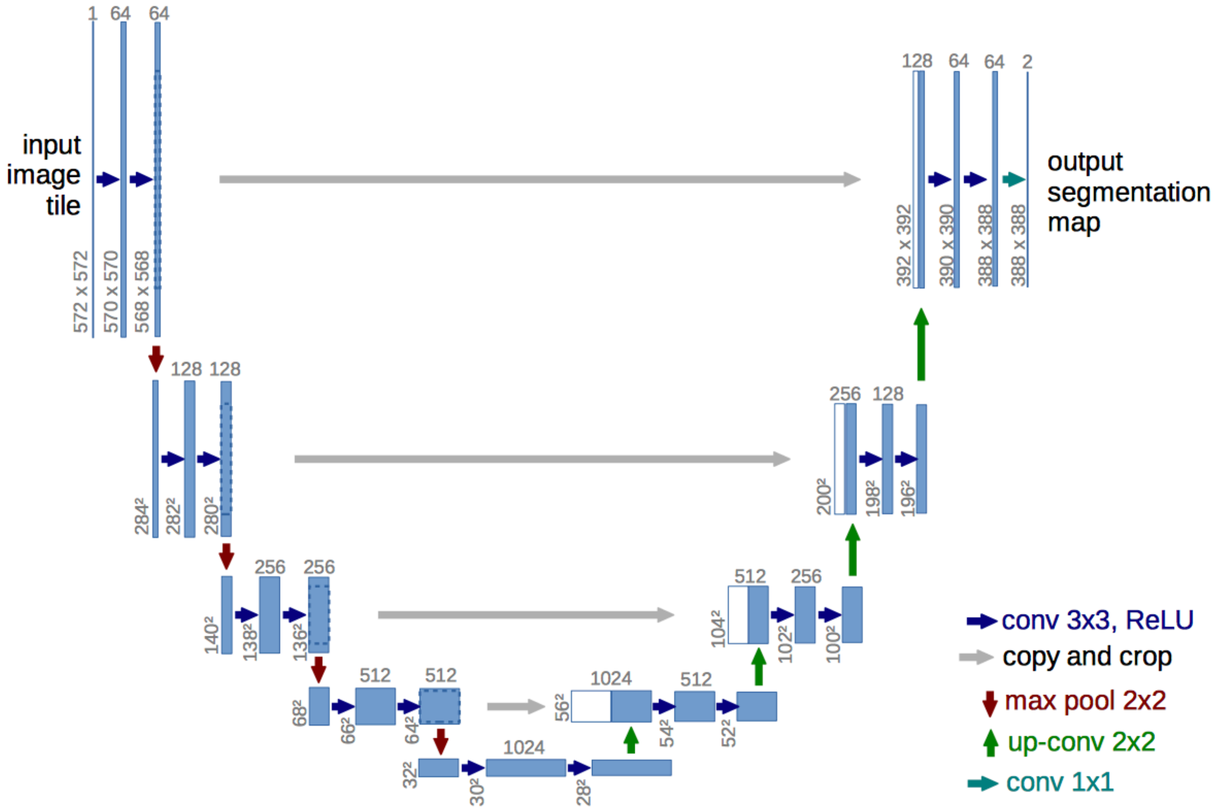


Output contains copies of the filter weighted multiplied by the input, summing at overlaps in the output

* slide from Fei-Dei Li, Justin Johnson, Serena Yeung, **cs231n Stanford**

U-Net Architecture

ResNet-like Fully convolutional CNN

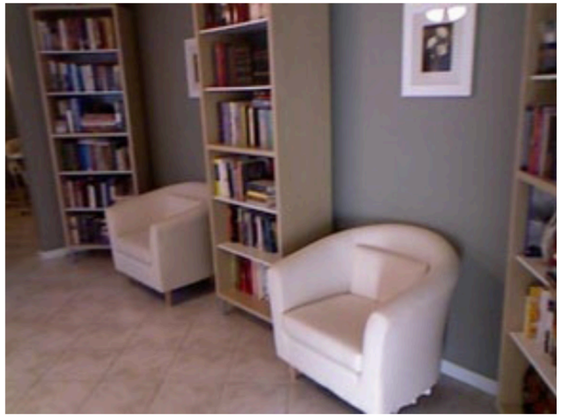


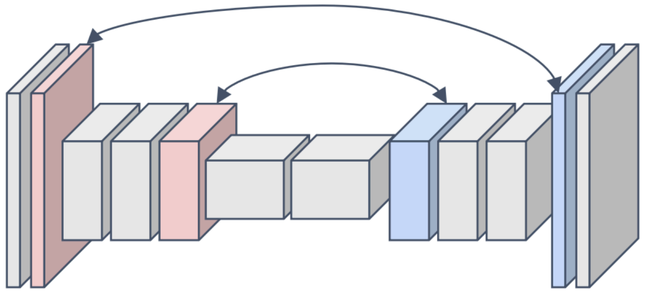
[Ronneberger et al, CVPR 2015]

Image **Colorization**

[Zhang et al. 2016]

Depth Estimation

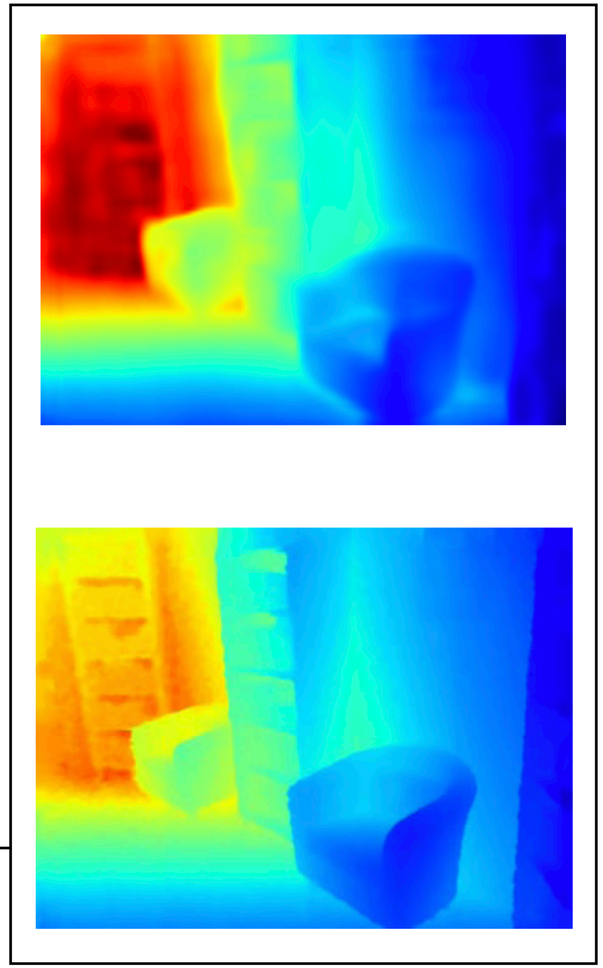




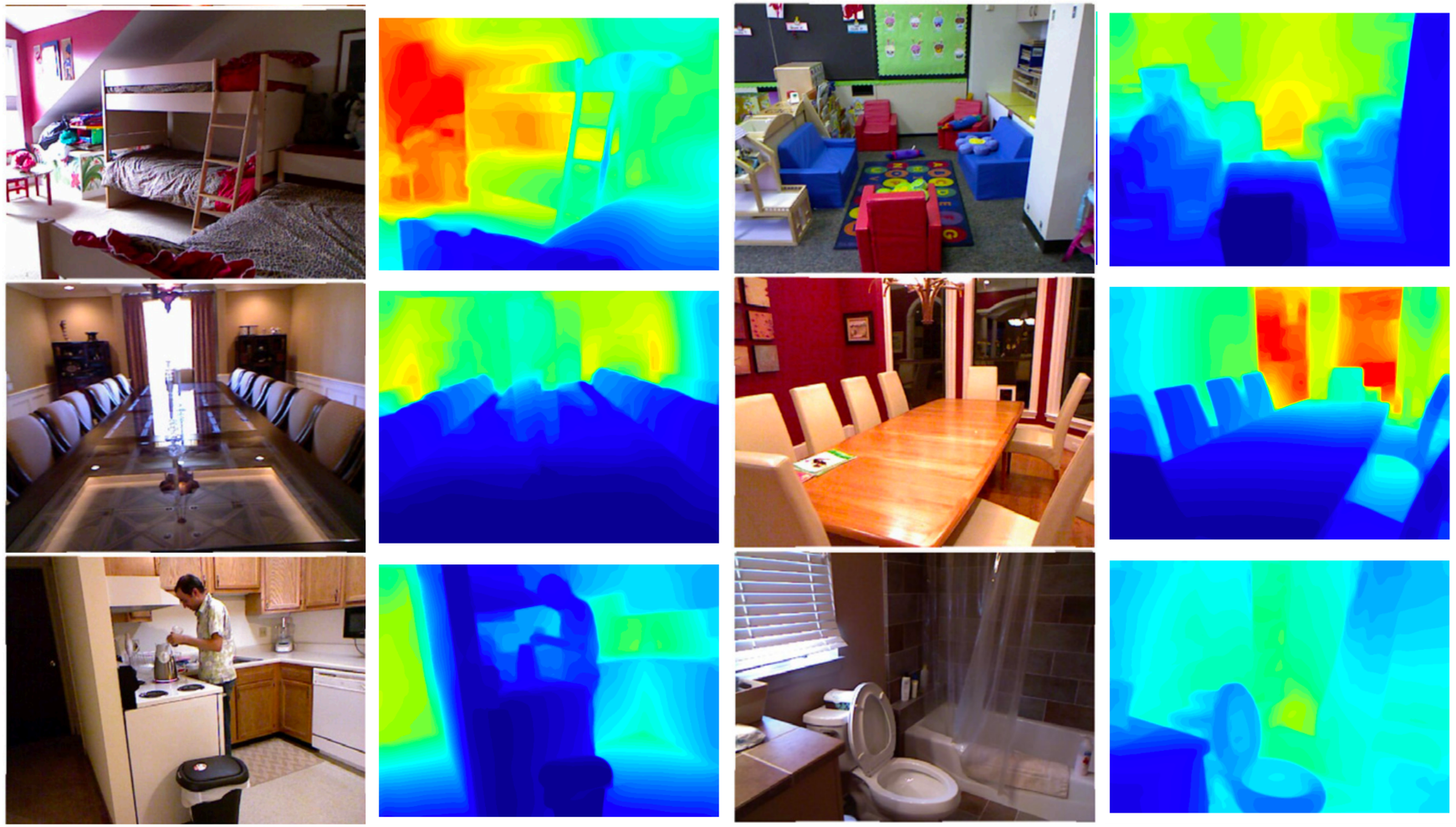
U-Net with skip connections

Direct supervision via Kinect RGB+D

Loss, e.g., L2



Depth Estimation



[DenseNet Huang et al 2018] [Alhashim Wonka 2019]

Super Resolution

bicubic
(21.59dB/0.6423)

SRResNet
(23.53dB/0.7832)

original

A state of the art super-res network trained with L2 loss is good at sharpening edges, but results lack realistic texture

[Ledig et al 2017]

Image-to-Image Translation

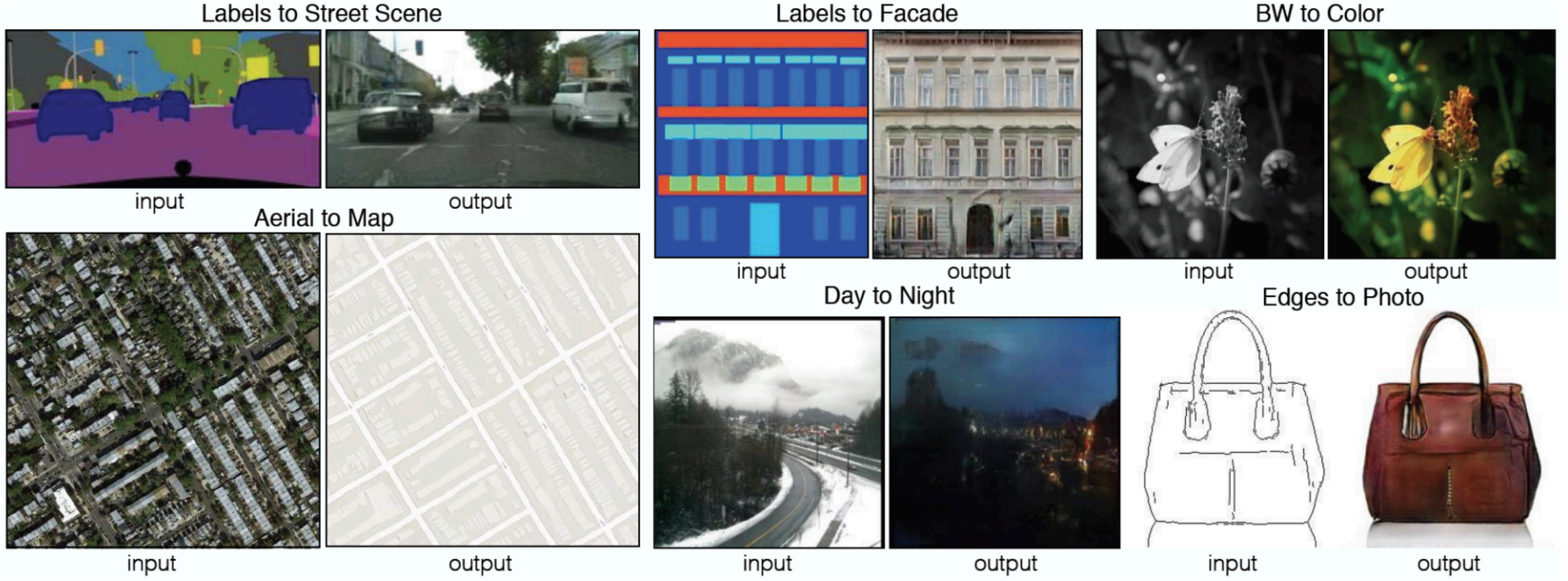


Figure 1 in the original paper.

[Isola et al., 2016]

Generative Adversarial Networks

Architecture: DCGAN-based

Training is conditioned on the **images from the source domain**

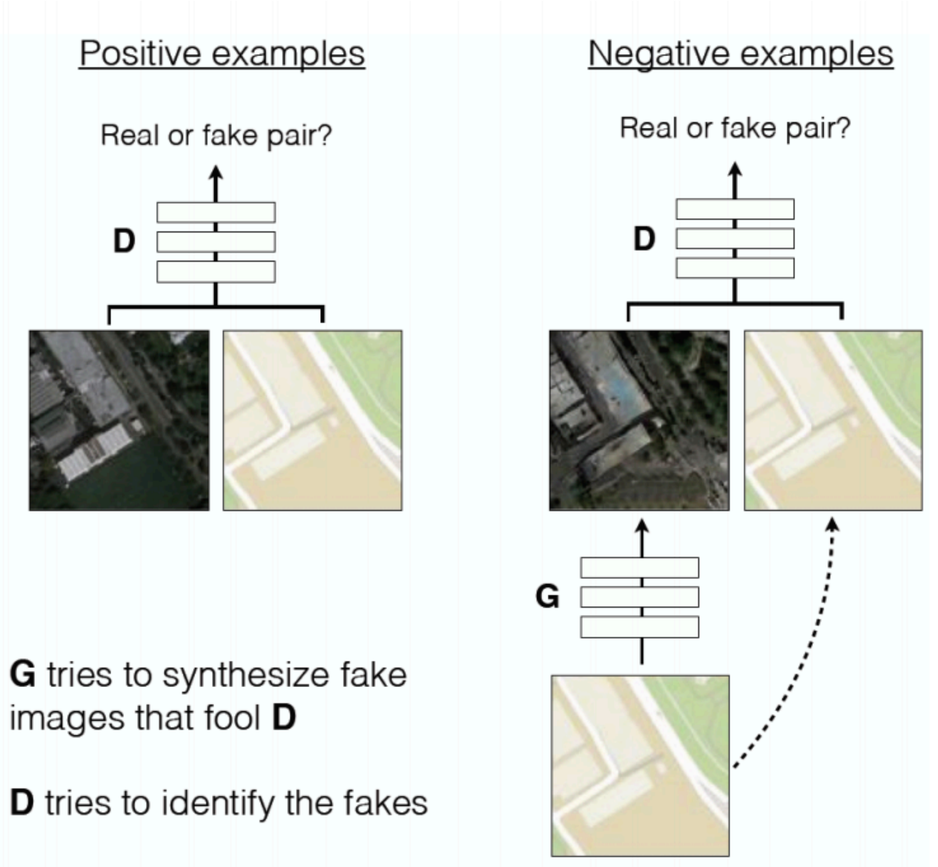
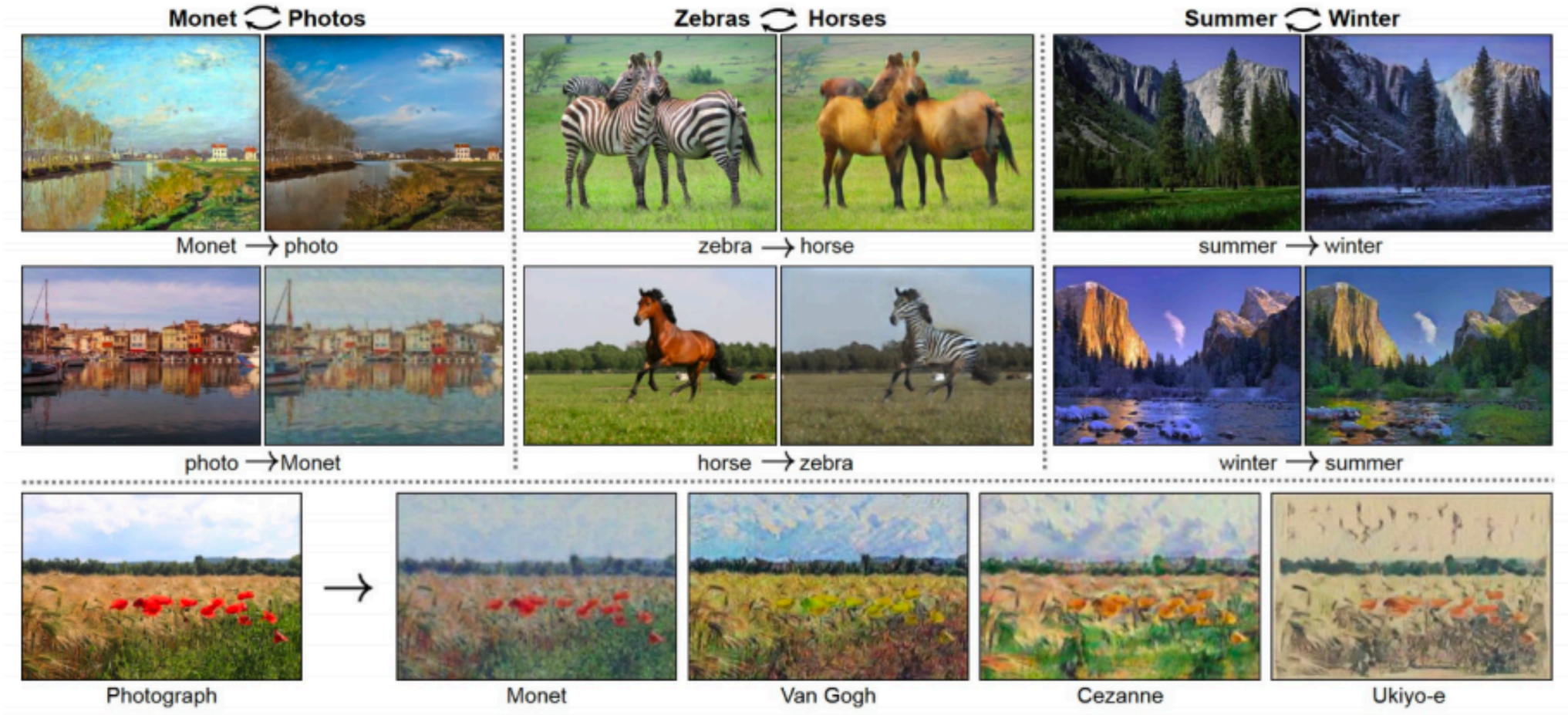


Figure 2 in the original paper.

[Isola et al., 2016]

Image-to-Image Translation

Style transfer: change the style of an image while preserving the content



Data: two unrelated collections of image, one for each style

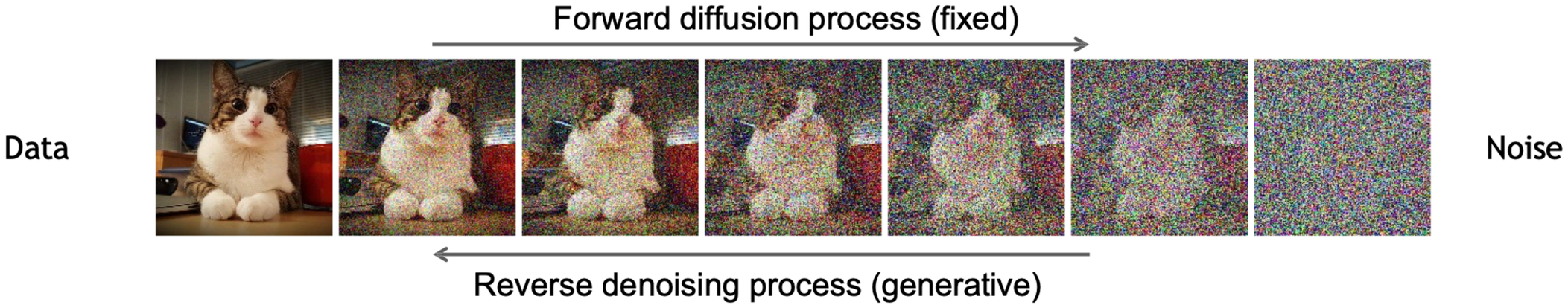
[Zhu et al., 2017]

Denoising **Diffusion** Models

Key Idea: Learning to generate by denoising

Denoising diffusion models consist of two processes:

- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising

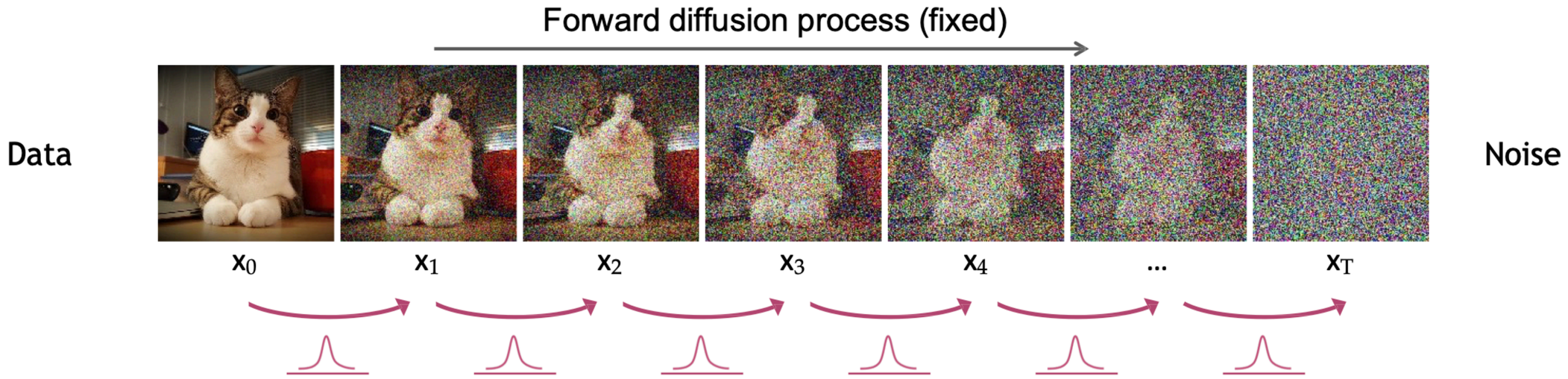


Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Slide from: https://www.cs.unc.edu/~ronisen/teaching/fall_2022/pdf_lectures/lecture7-8_diffusion_model.pdf

Forward Diffusion Process

The formal definition of the forward process in T steps:



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x_t}; \sqrt{1-\beta_t}\mathbf{x_{t-1}}, \beta_t\mathbf{I})$$

mean: $\sqrt{1-\beta_t}\mathbf{x_{t-1}}$ variance: $\beta_t\mathbf{I}$

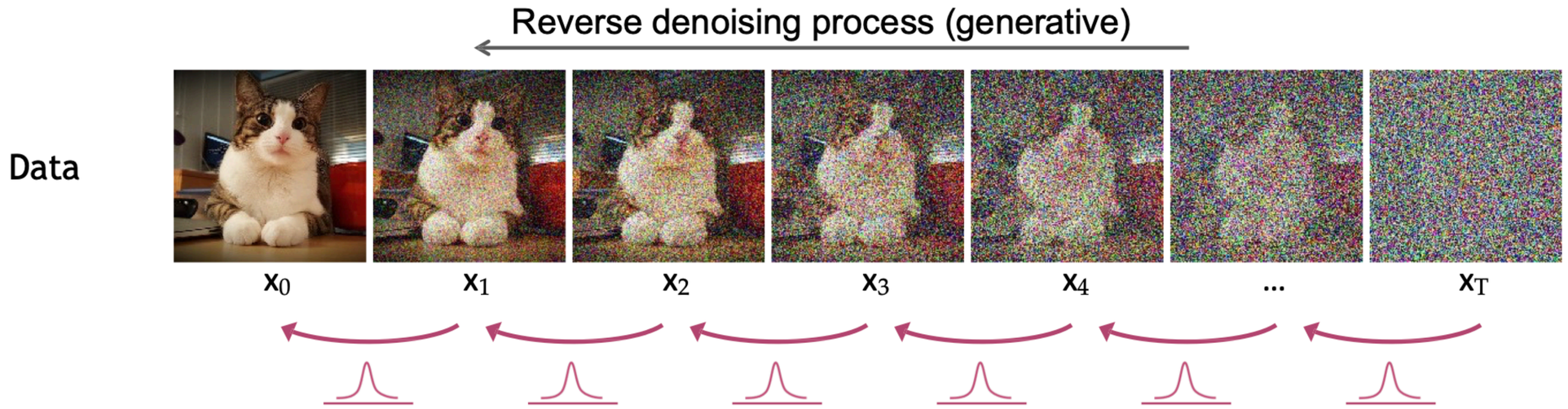
➡ Sample: $x_t = \sqrt{1-\beta_t}x_{t-1} + \sqrt{\beta_t}\epsilon_{t-1}$

where, $\epsilon_{t-1} \sim \mathcal{N}(0, \mathbf{I})$

Slide from: https://www.cs.unc.edu/~ronisen/teaching/fall_2022/pdf_lectures/lecture7-8_diffusion_model.pdf

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

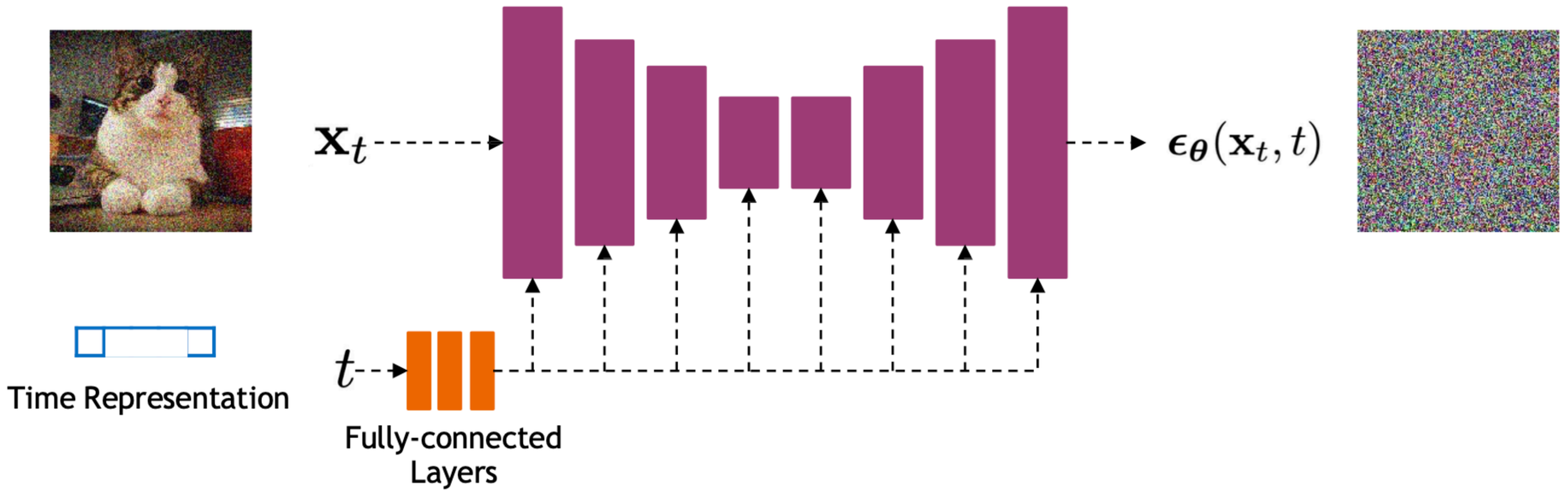
Trainable network
(U-net, Denoising Autoencoder)

Autoencoder predicts the mean of the denoised image x(t-1) given x(t).

Slide from: https://www.cs.unc.edu/~ronisen/teaching/fall_2022/pdf_lectures/lecture7-8_diffusion_model.pdf

Implementation: Network Architecture

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$

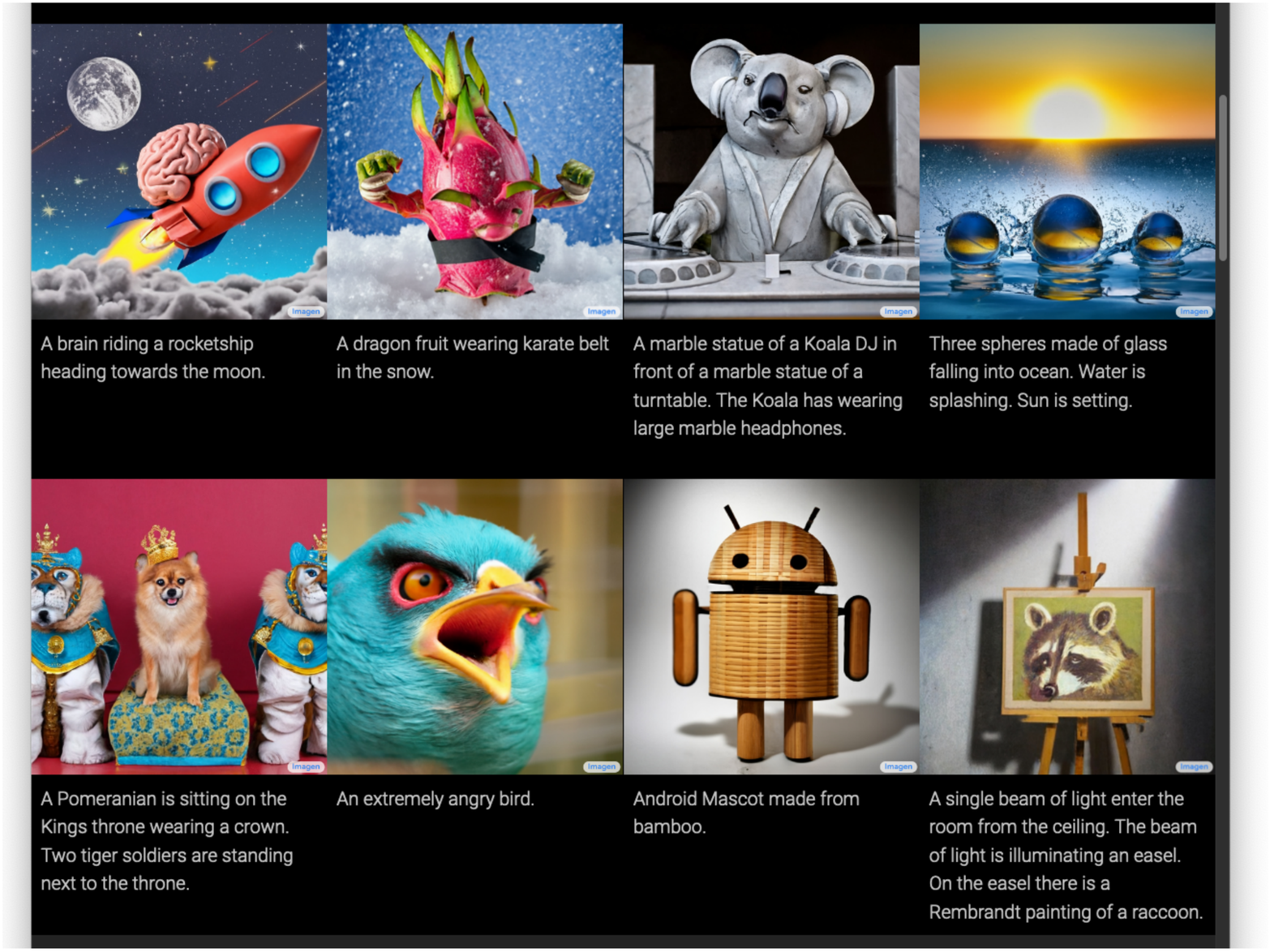


Time representation: sinusoidal positional embeddings or random Fourier features.

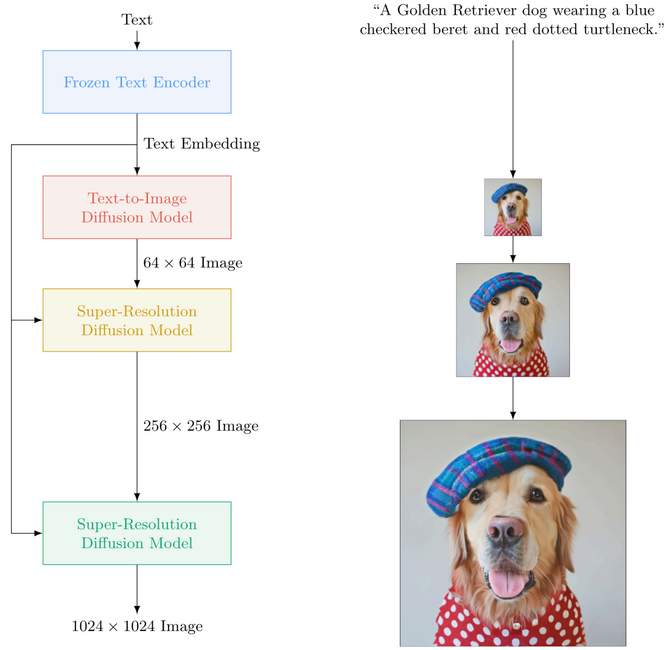
Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

Slide from: https://www.cs.unc.edu/~ronisen/teaching/fall_2022/pdf_lectures/lecture7-8_diffusion_model.pdf

4. Visual **Imagination**



- imagen.research.google
- Text to image generation
- Uses diffusion process, training using large dataset of text (web scale) and image-text (400M) pairs

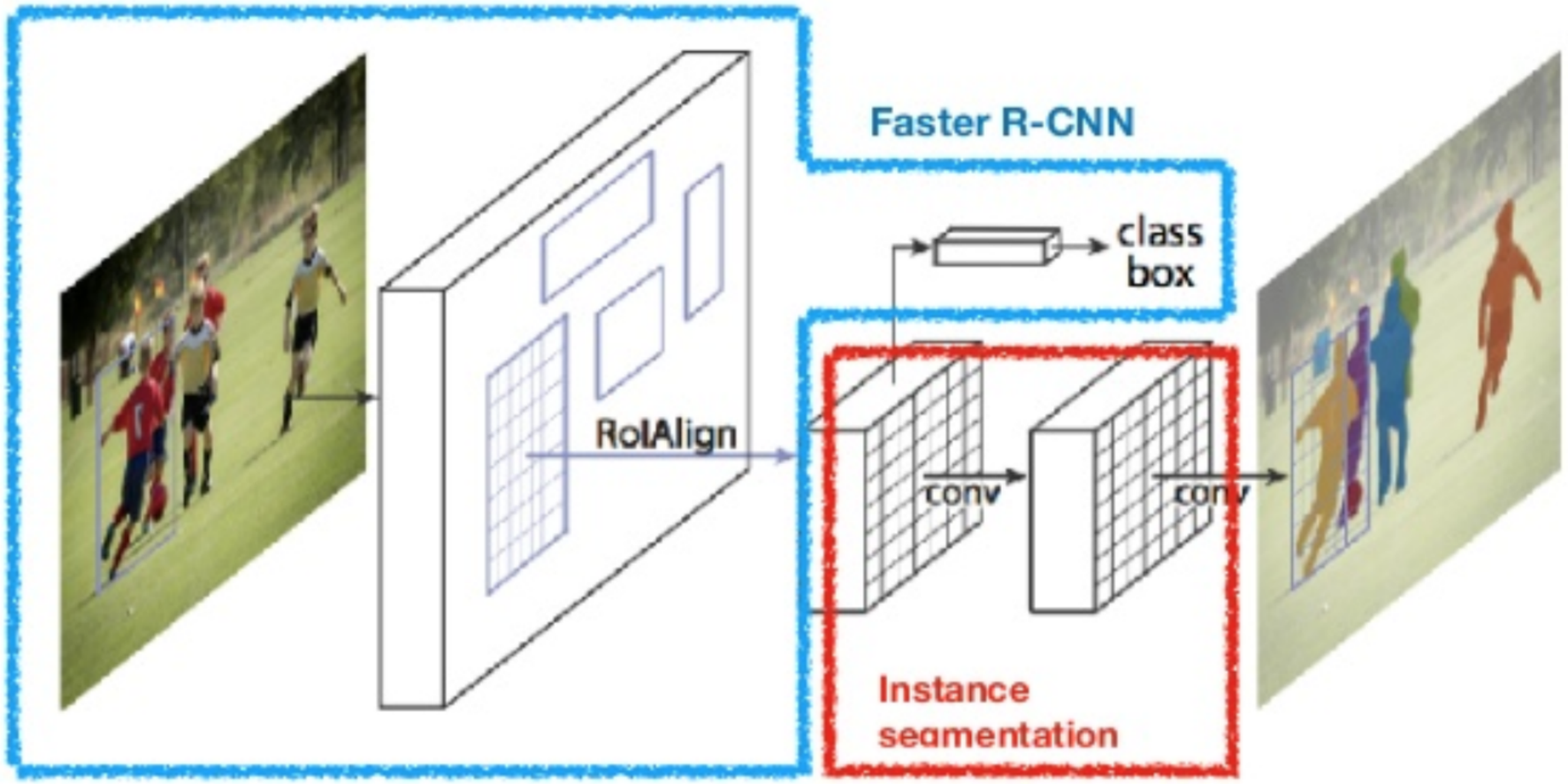


Computer **Vision Problems** (no language for now)

Instance Segmentation

Horse1
Horse2
Person1
Person2

Mask R-CNN



[He et al, 2017]

Summary

Common types of layers:

1. **Convolutional** Layer
   — Parameters define a set of learnable filters

2. **Pooling** Layer
   — Performs a downsampling along the spatial dimensions

3. **Fully-Connected** Layer
   — As in a regular neural network

Each layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function

Summary

The parameters of a neural network are learned using **backpropagation**, which computes gradients via recursive application of the chain rule

A **convolutional neural network** assumes inputs are images, and constrains the network architecture to reduce the number of parameters

A **convolutional layer** applies a set of learnable filters

A **pooling layer** performs spatial downsampling

A **fully-connected** layer is the same as in a regular neural network

Convolutional neural networks can be seen as learning a hierarchy of filters