Lecture 6: From SOTA Models to Transfer Learning across Tasks

AC295

Advanced Practical Data Science

Pavlos Protopapas

Outline

1: Communications and Recap

2: SOTA Deep Models

3: Transfer Learning across Tasks

Segmentation

Communications

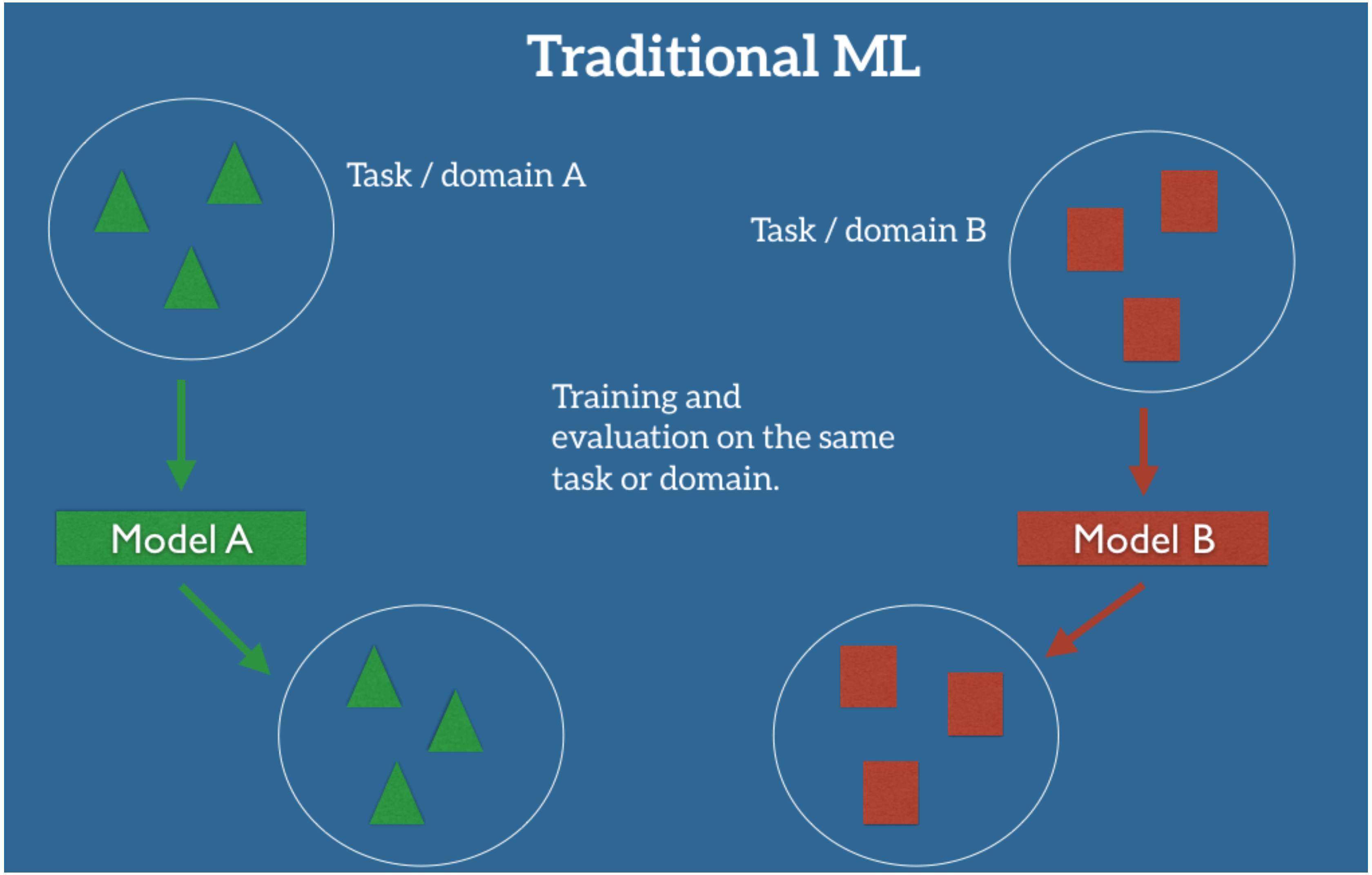
Practicum 2:

Recap

A. TL

Transfer Learning To The Rescue

- Train on a big "**source**" data set, with a big model, on one particular downstream tasks (say classification). Do it once and save the parameters. This is called a **pre-trained model**.
- Use these parameters for other smaller "**target** " datasets, say, for classification on new images (possibly different **domain**, or training distribution), or for image segmentation on old images(new **task**), or new images (new task and new domain).
- Less helpful if you have a large target dataset with many labels.
- Will fail if source domain (where you trained big model) has nothing in common with target domain (that you want to train on smaller data set).


Traditional ML
Task / domain A
Task / domain B
Training and evaluation on the same task or domain.
Model A
Model B

Outline

1: Communications and Recap

2: SOTA Deep Models

3: Transfer Learning across Tasks

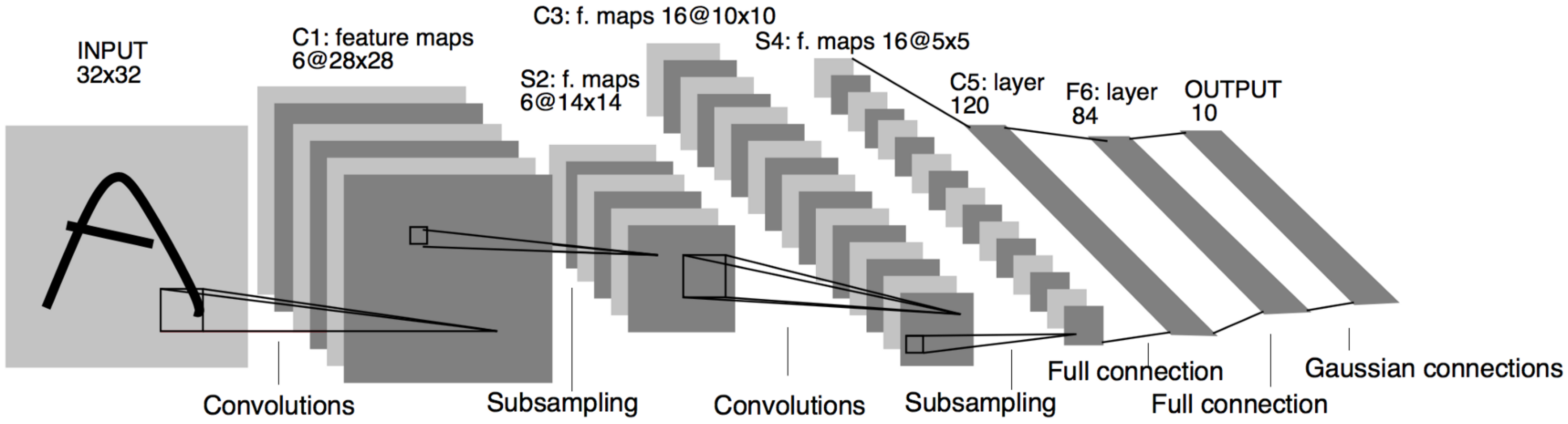
SOTA Deep Models: Initial Ideas

- The first piece of research proposing something similar to a Convolutional Neural Network was authored by Kunihiko Fukushima in 1980 and was called the **NeoCognitron**[1].
- Inspired by discoveries on visual cortex of mammals.
- End of the 80's: several papers advanced the field
  - **Backpropagation** published Yann LeCun in 1985 (independently by other researchers as well)
  - TDNN by Waiber et al., 1989 - **Convolutional-like** network trained with backprop.
  - Backpropagation applied to handwritten zip code recognition by LeCun et al., 1989

[1] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological Cybernetics, 36(4): 93-202, 1980.

SOTA Deep Models: Le Net

- In 1998 **LeCun** publishes one of his most recognized papers describing a "modern" CNN architecture for document recognition, called LeNet[2]. **Not his first iteration**, this was indeed LeNet-5.

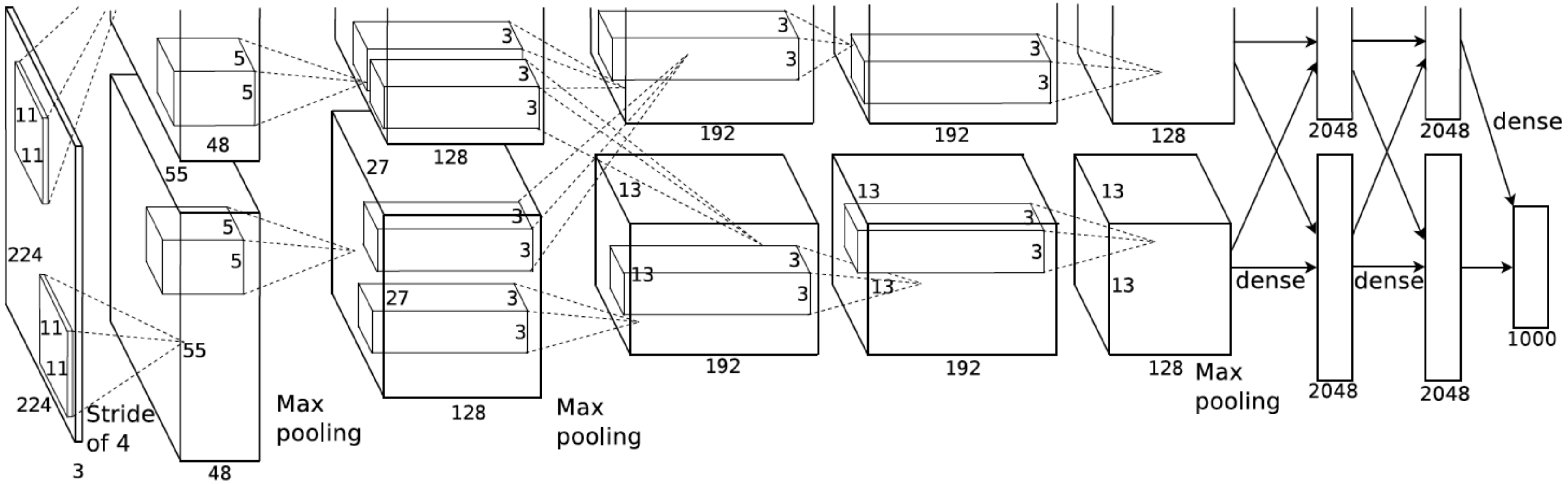


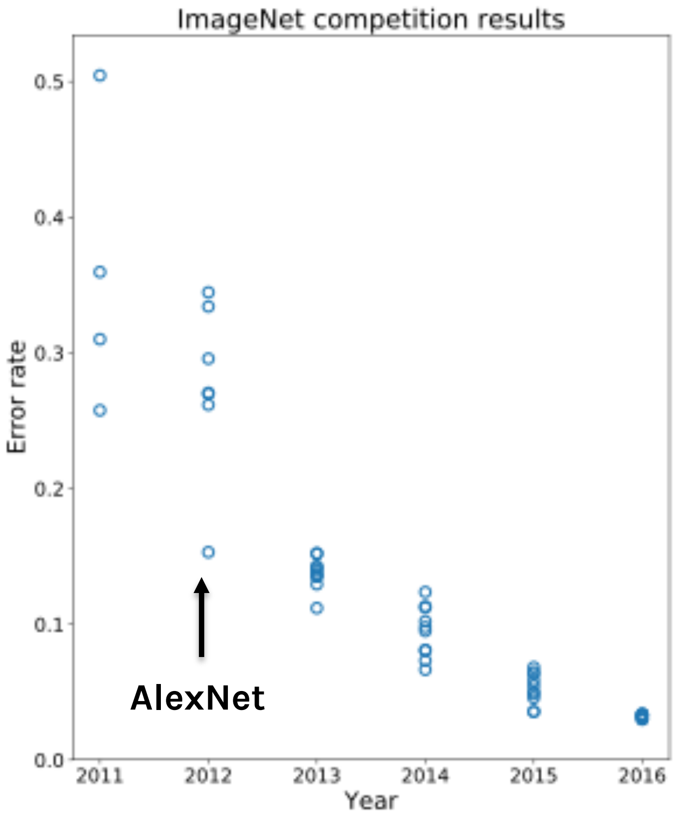
[2] LeCun, Yann, et al. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86.11 (1998): 2278-2324.

SOTA Deep Models: AlexNet

- Developed by **Alex Krizhevsky, Ilya Sutskever and Geoffrey Hinton** at Utoronto in 2012. More than 25000 citations.
- Destroyed the competition in the 2012 **ImageNet Large Scale Visual Recognition Challenge**. Showed benefits of CNNs and kickstarted AI revolution.
- Main contributions:
    - Trained on ImageNet with data augmentation.
    - Increased depth of model, GPU training (5/6 days).
    - Smart optimizer and Dropout layers.
    - ReLU activation!

SOTA Deep Models: AlexNet <cont>

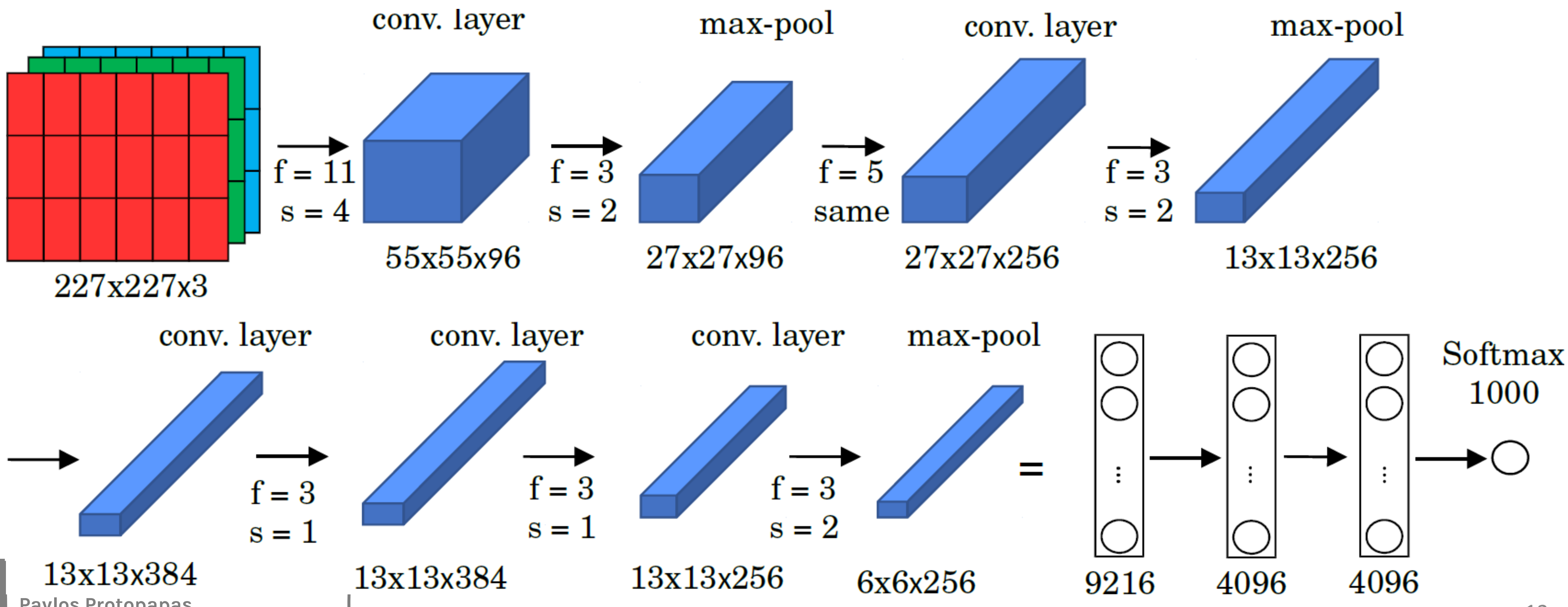




https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf

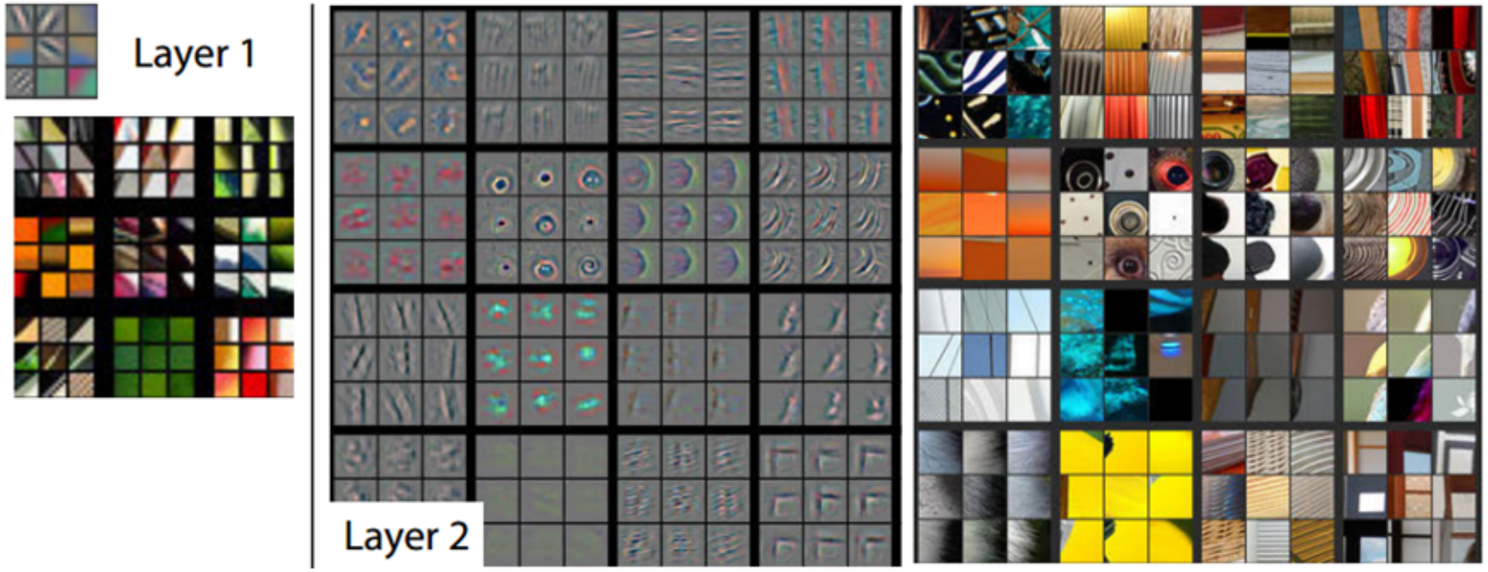
SOTA Deep Models: AlexNet <cont>

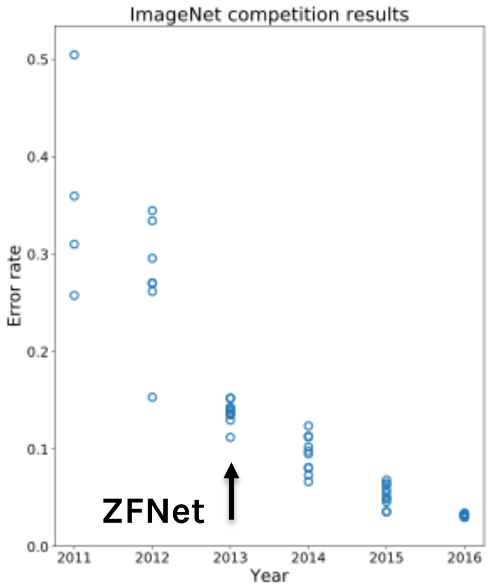
- Trained on 1.2 million high-resolution (227x227x3) images in the ImageNet 2012 contest;
- 1000 different classes, NN with 60 million parameters to optimize (~ 255 MB).



SOTA Deep Models: ZFNet

- Introduced by **Matthew Zeiler and Rob Fergus** from NYU, won ILSVRC 2013 with 11.2% error rate. **Decreased sizes of filters**.
- Paper presented a visualization technique named Deconvolutional Network, which helps to **examine different feature activations and their relation to the input space**. Trained for 12 days.

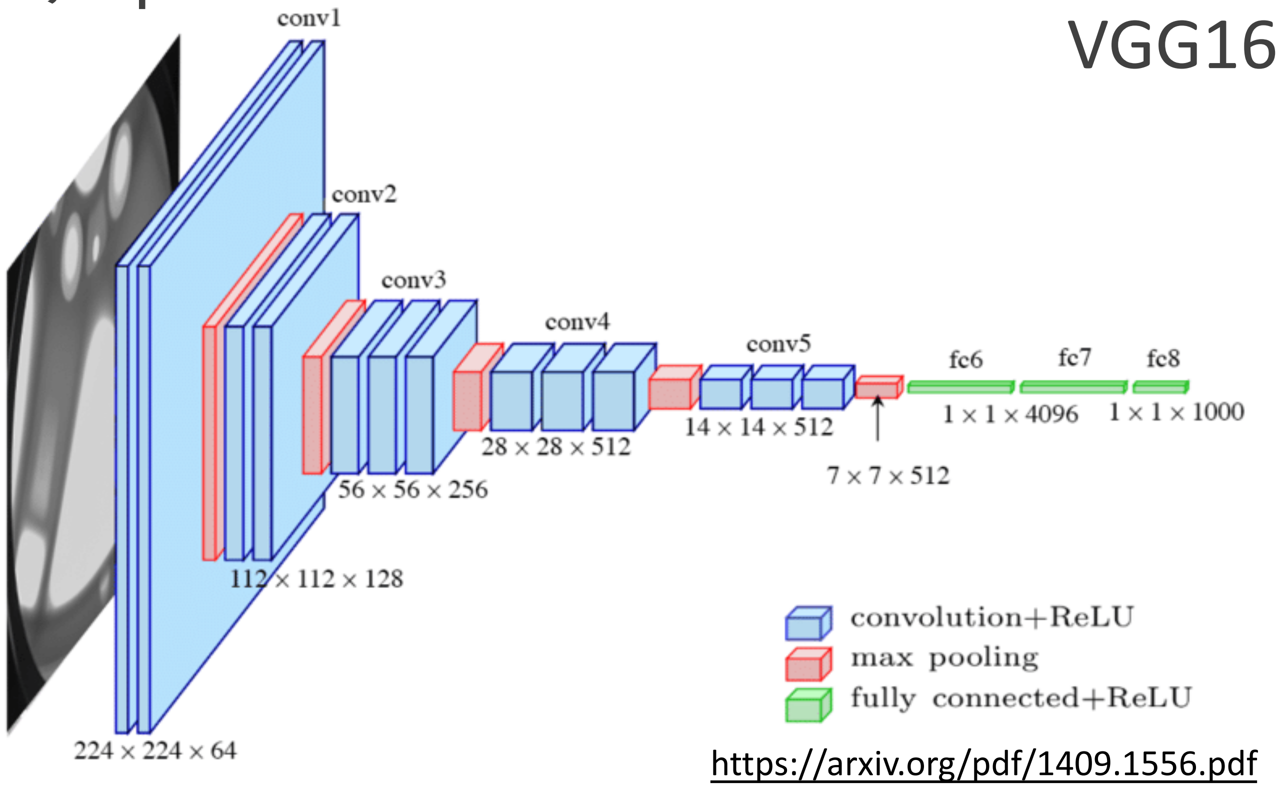




SOTA Deep Models: VGG16

- Introduced by **Simonyan and Zisserman** in 2014
- **Simplicity and depth** as main points
- **Uses 3x3 filters exclusively and 2x2 MaxPool** layers with stride 2
- Showed that two 3x3 filters have an effective receptive field of 5x5
- As **spatial size decreases, depth increases**
- Trained for 2/3 weeks
- Still used as of today

VGG16

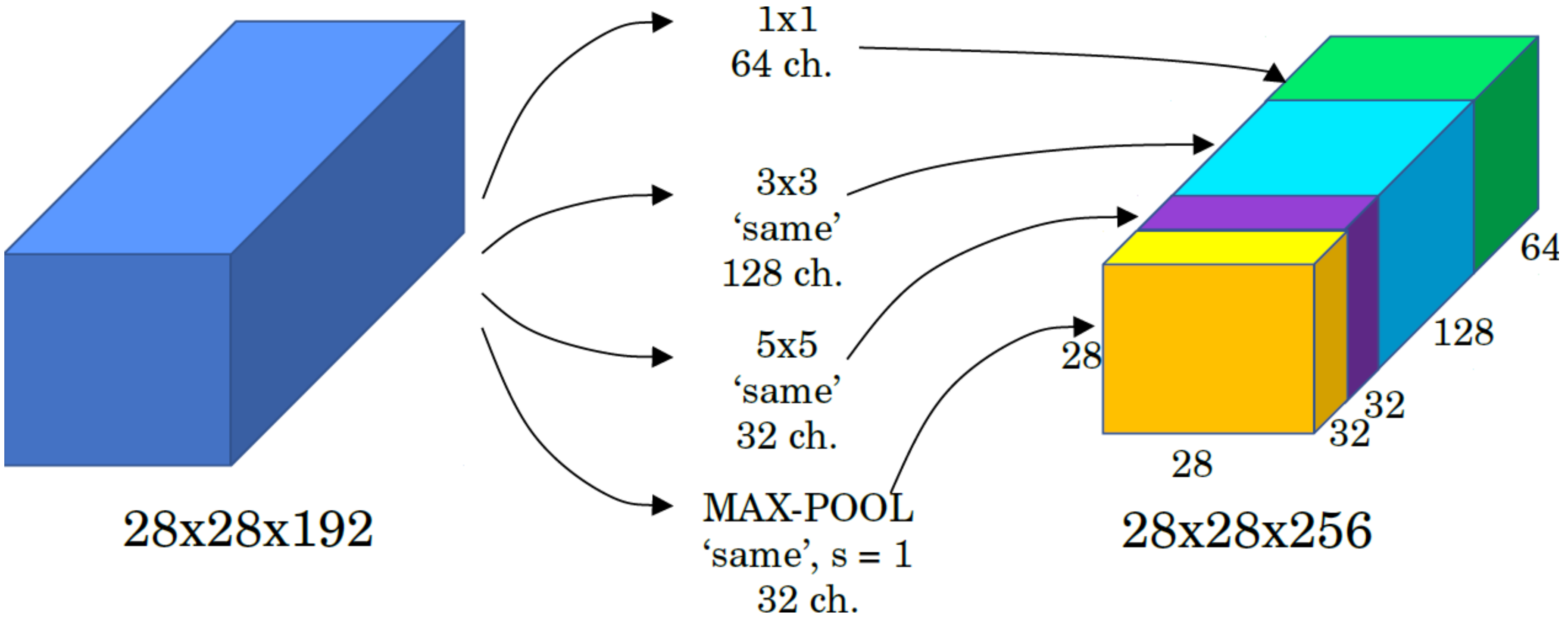


https://arxiv.org/pdf/1409.1556.pdf


WE NEED TO GO
DEEPER

SOTA Deep Models: Inception (GoogLeNet)

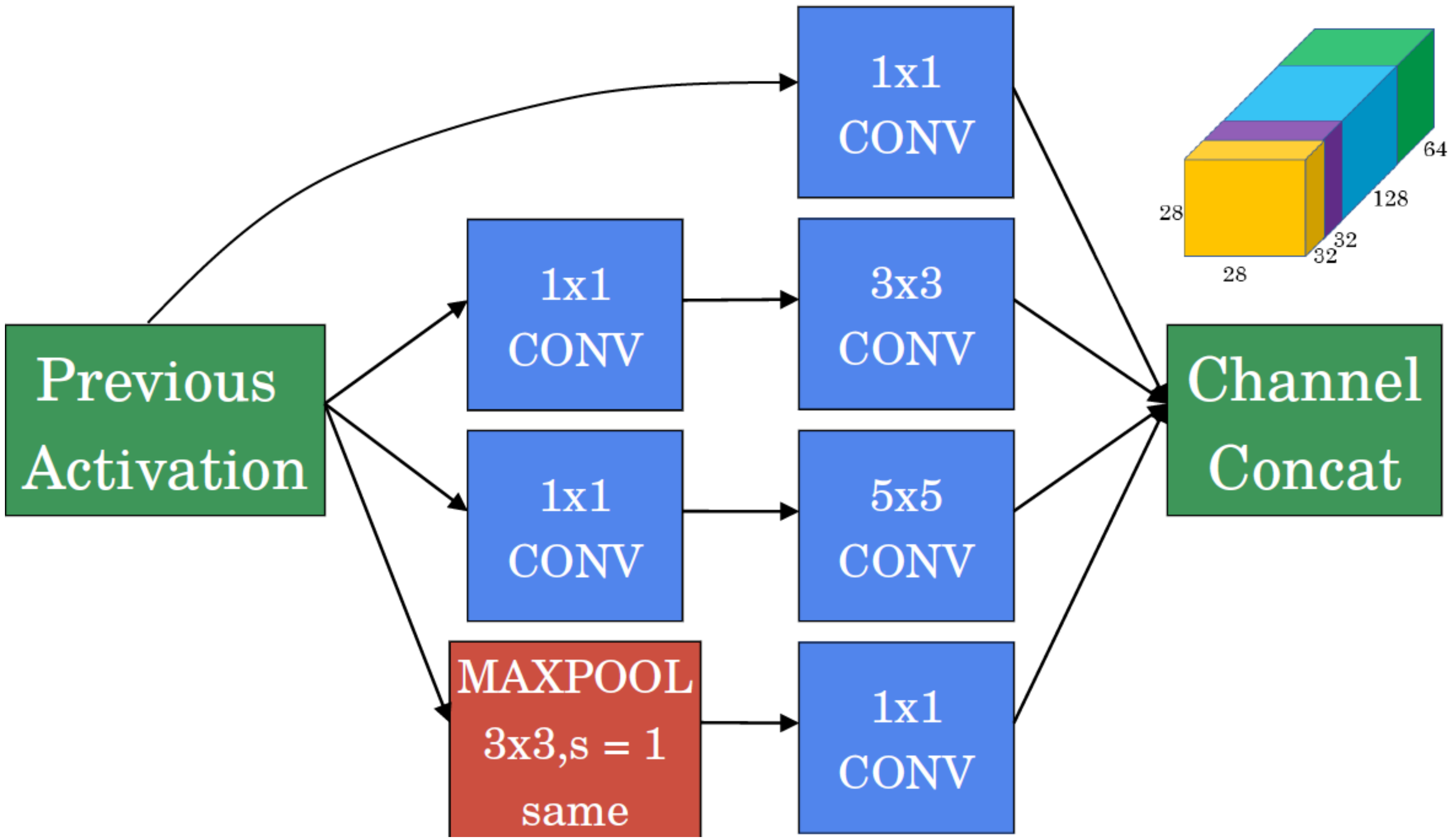
- The motivation behind inception networks is to use **more than a single type of convolution layer at each layer**.
- Use 1 x 1, 3 x 3, 5 x5 convolutional layers, and max-pooling layers **in parallel**.
- **All modules use same convolution**.



https://arxiv.org/pdf/1409.4842.pdf

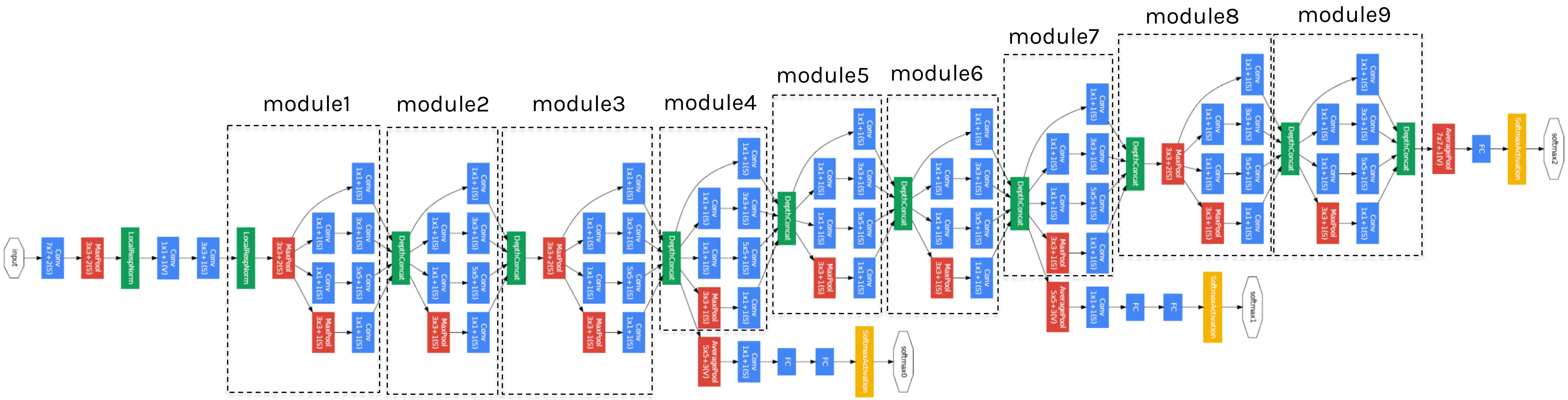
SOTA Deep Models: Inception (GoogLeNet)

- Use 1 x 1 convolutions that **reduce the size of the channel dimension**.
- The **number of channels can vary** from the input to the output.



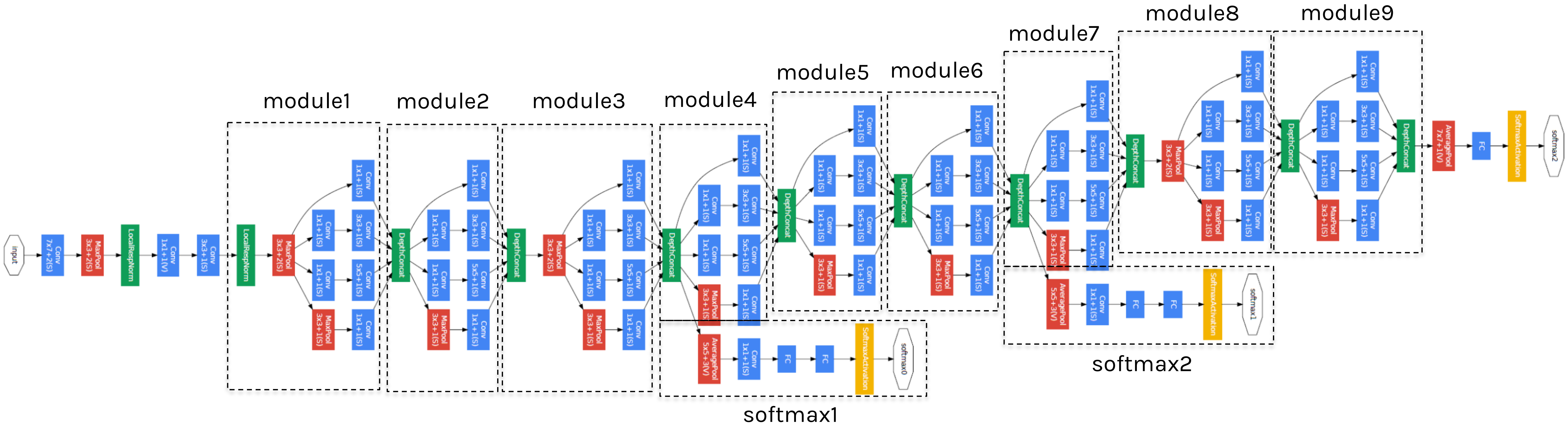
SOTA Deep Models: Inception (GoogLeNet)

- The inception network is formed by **concatenating other inception modules**.



SOTA Deep Models: Inception (GoogLeNet)

- It includes several auxiliary softmax output units to t to combat the vanishing gradient It includes several auxiliary softmax output units to t to combat the vanishing gradient problem while providing regularization.

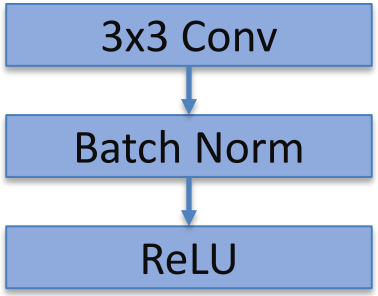


- Auxiliary networks are not used during prediction.

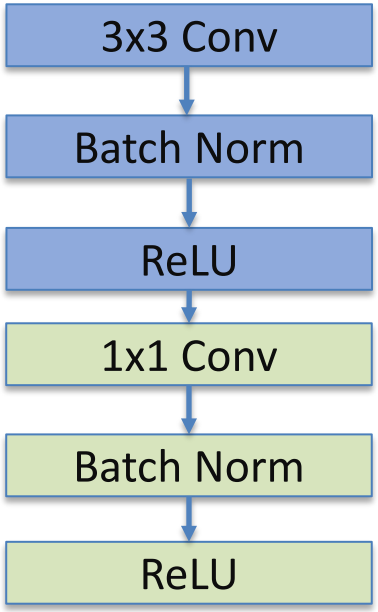
SOTA Deep Models: MobileNet

- Presented by **Howard et al.** (Google), 2017;
- The MobileNet model is based on **depthwise separable convolutions (DW)**
- DW factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution.

Standard Convolution



Depth-Wise Separable Convolution (DW)



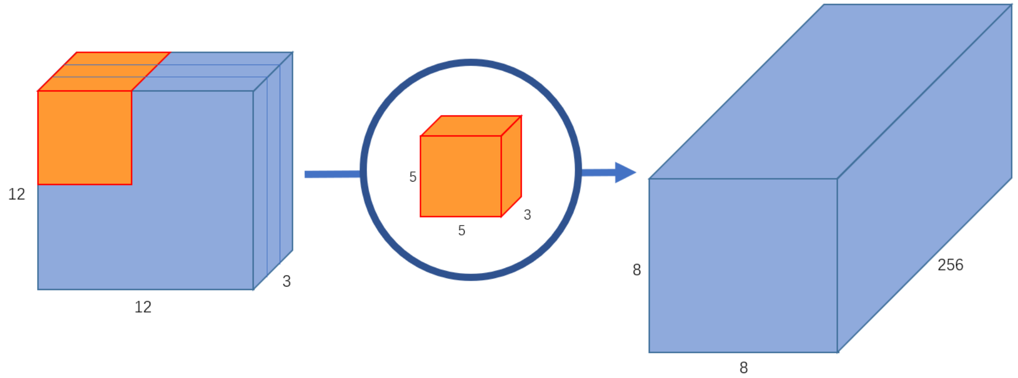
https://arxiv.org/pdf/1704.04861.pdf

Fig. Left: Standard convolutional layer with batchnorm and ReLU. Right Depthwise Separable convolutions with Depthwise and pointwise layers followed by batchbnorm and ReLU

MobileNet <cont>

Standard Convolution

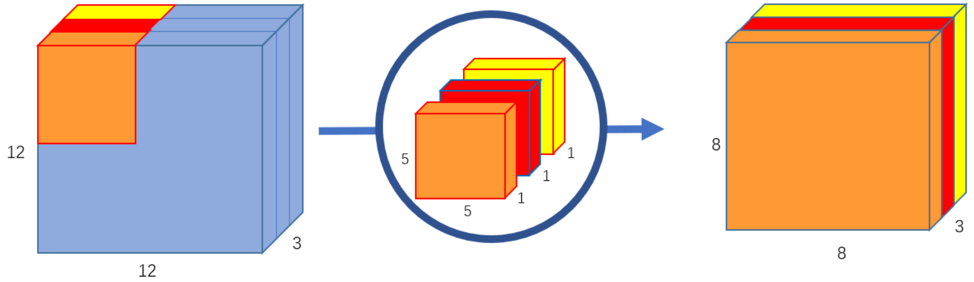
Filters and combines inputs into a new set of outputs in one step



Input: 12x12x3
Filter: 5x5x3x256

Output: 8x8x256
(no padding)

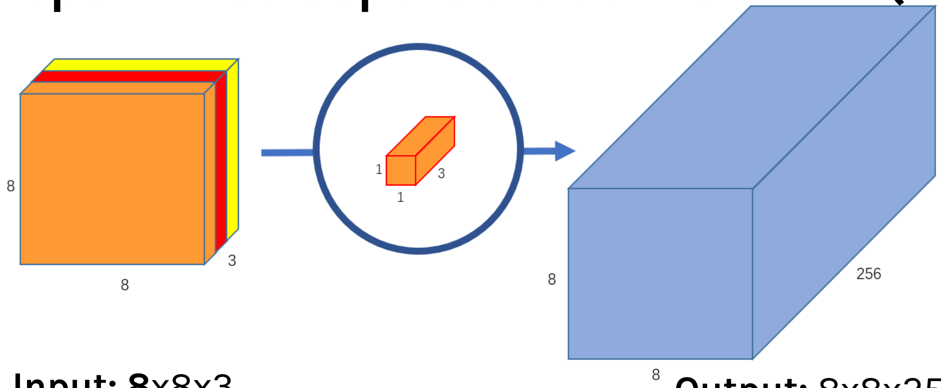
Depth-Wise Separable Convolution (DW)



Input: 12x12x3
Filter: 5x5x3

Output: 8x8x3
(no padding)

Depth-Wise Separable Convolution (DW)



Input: 8x8x3
Filter: 1x1x3x256

Output: 8x8x256
(no padding)

MACs: (5x5)x3x256x(12x12) ~ **2.8M**
Parameters: (5x5x3)x256 + 256 ~ **20K**

MACs: (5x5)x3x(12x12) + 3x256x(8x8) ~ **60K**
Parameters: (5x5x3 + 3) + (1x1x3x256+256) ~ **1K**

SOTA Deep Models: MobileNet <cont>

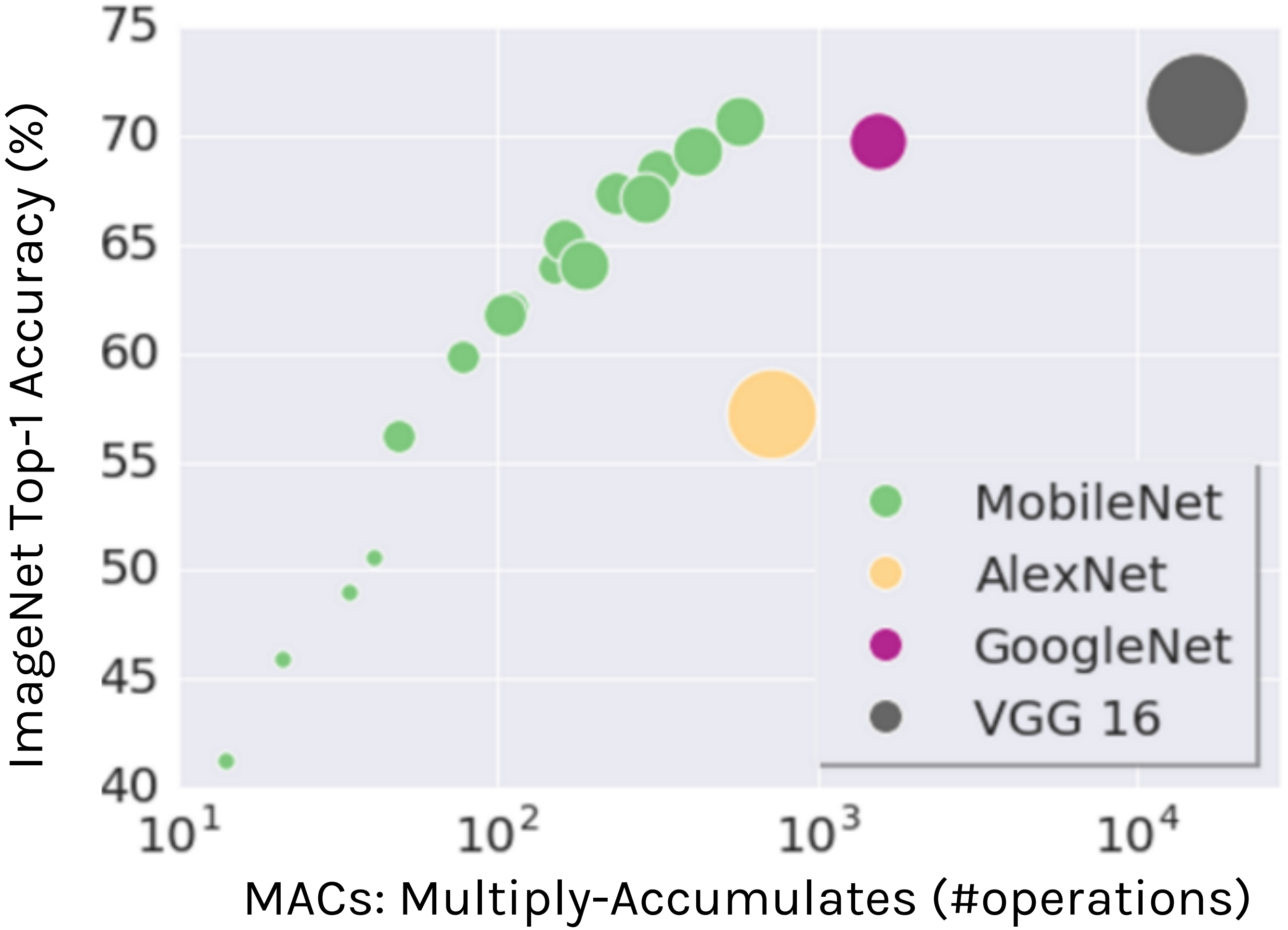


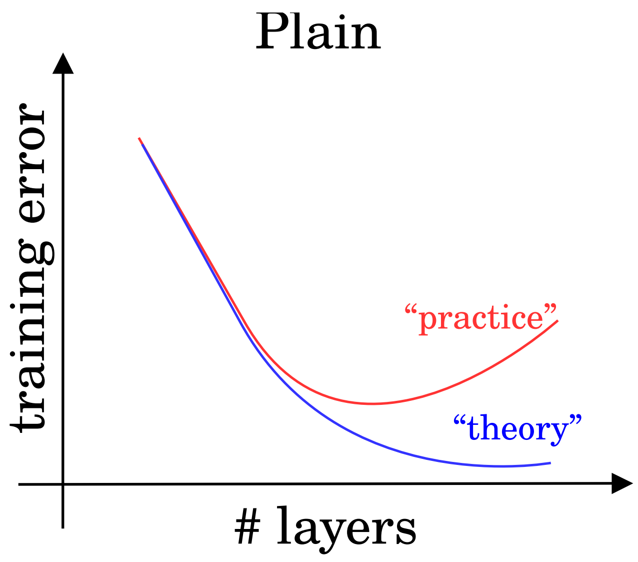
Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
5× Conv dw / s1 Conv / s1	$3 \times 3 \times 512$ dw $1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$ $14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

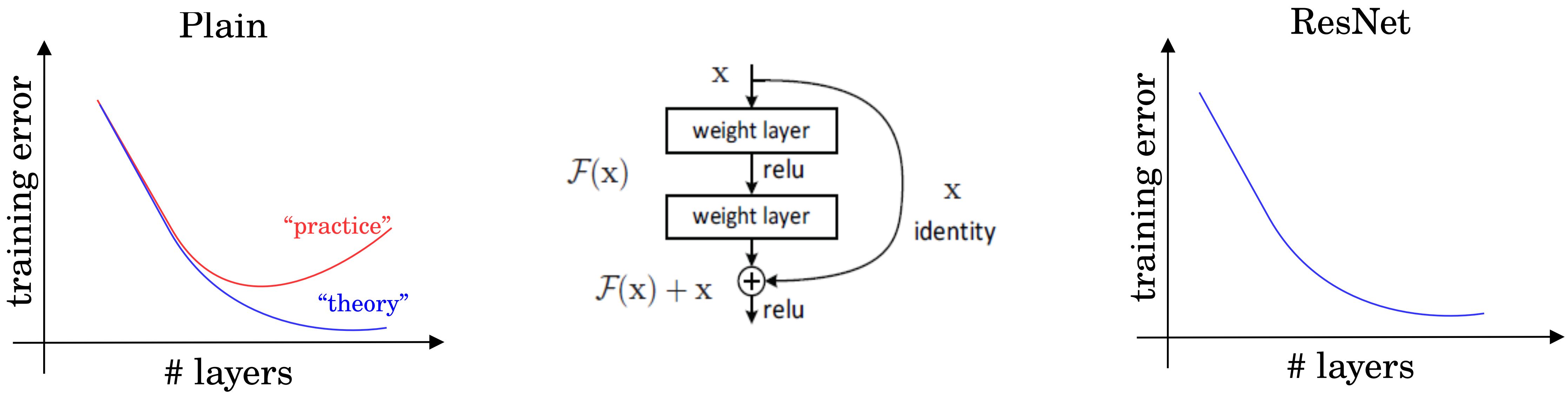
MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications (arXiv.1704.04861)


WE NEED TO GO DEEPER
memegenerator.net

SOTA Deep Models: ResNet



SOTA Deep Models: ResNet

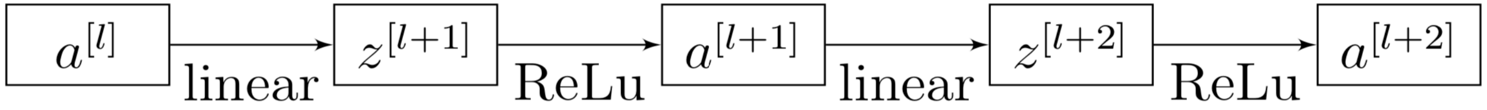


- Presented by **He et al.** (Microsoft), 2015. Won ILSVRC 2015 in multiple categories.
- Main idea: **residual block** that allows for extremely deep networks.
- It is easier to **optimize the residual mapping** than the original one. Furthermore, **residual block can decide to "shut itself down**" if needed.

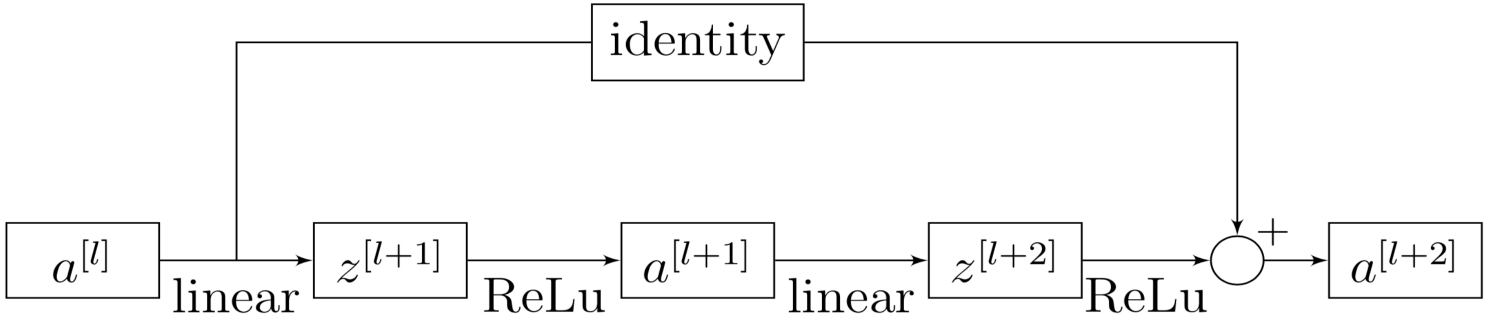
SOTA Deep Models: ResNet <cont>

- Network structure a) without and b) with *"Residual Block"*

a)



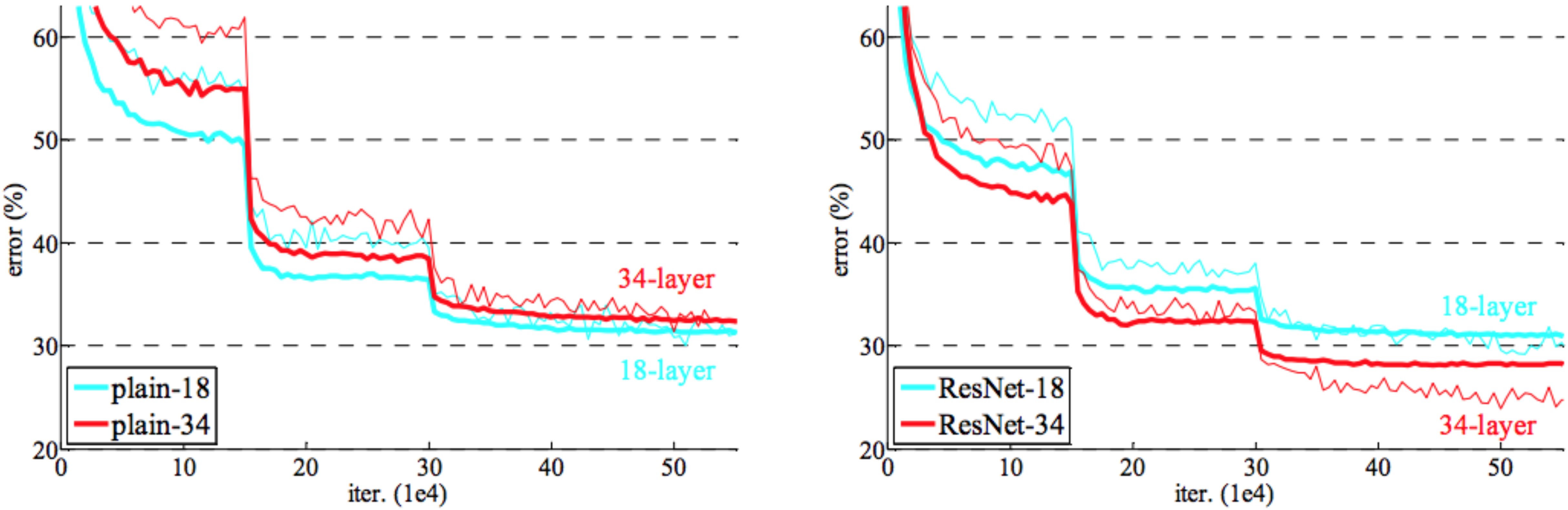
b)



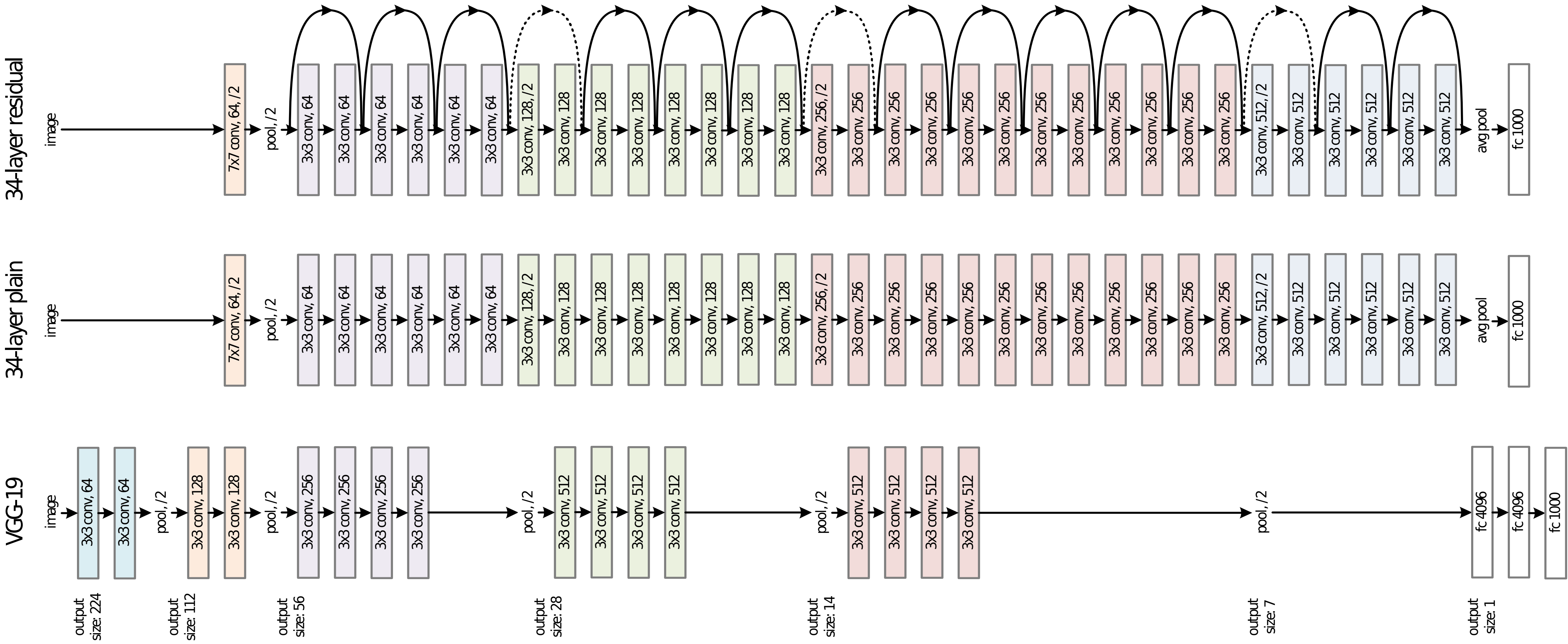
Original idea in Highway Networks: https://arxiv.org/pdf/1505.00387.pdf

SOTA Deep Models: ResNet <cont>

- The residual network stacks blocks sequentially (fig. a);
- The network become deeper without increasing the training time (fig. b).



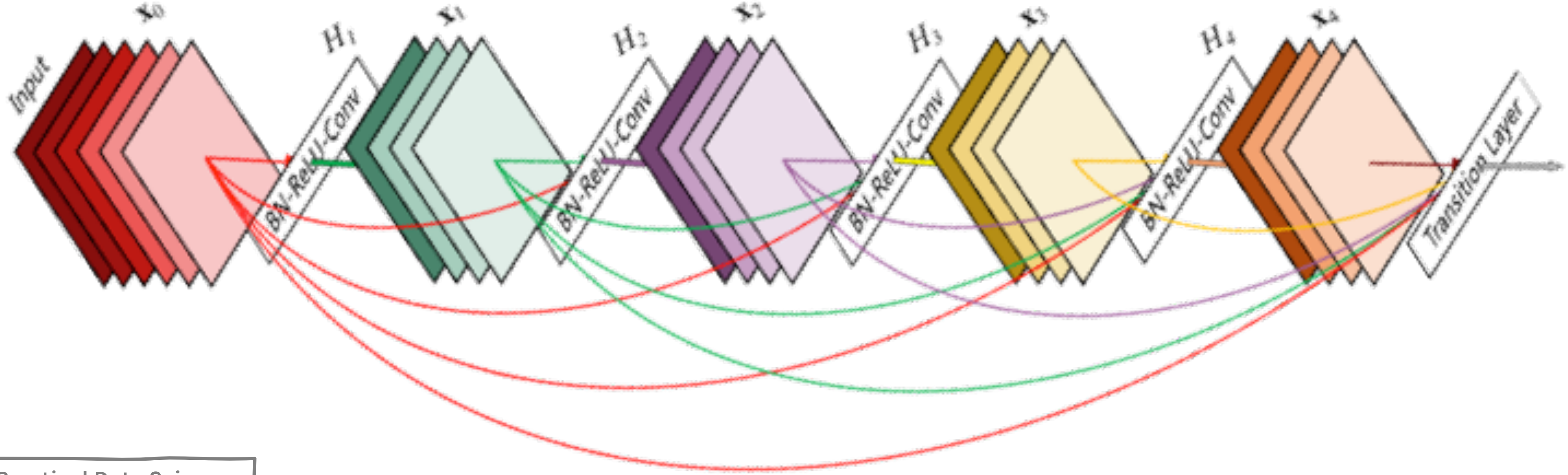
SOTA Deep Models: ResNet <cont>



Networks comparison: *ResNet34* (top), *ResNet34 "Without Residual Blocks"* (middle), VGG-19 (bottom).

SOTA Deep Models: DenseNets

- Motivation: to allow **maximum information (and gradient) flow** to connect every layer directly with each other.
- Connect every layer directly with each other.
- DenseNets **concatenates** outputs from the previous layers instead of using the summation.
- DenseNets **layers are very narrow** (e.g. 12 filters), and they just add a small set of new feature-maps.



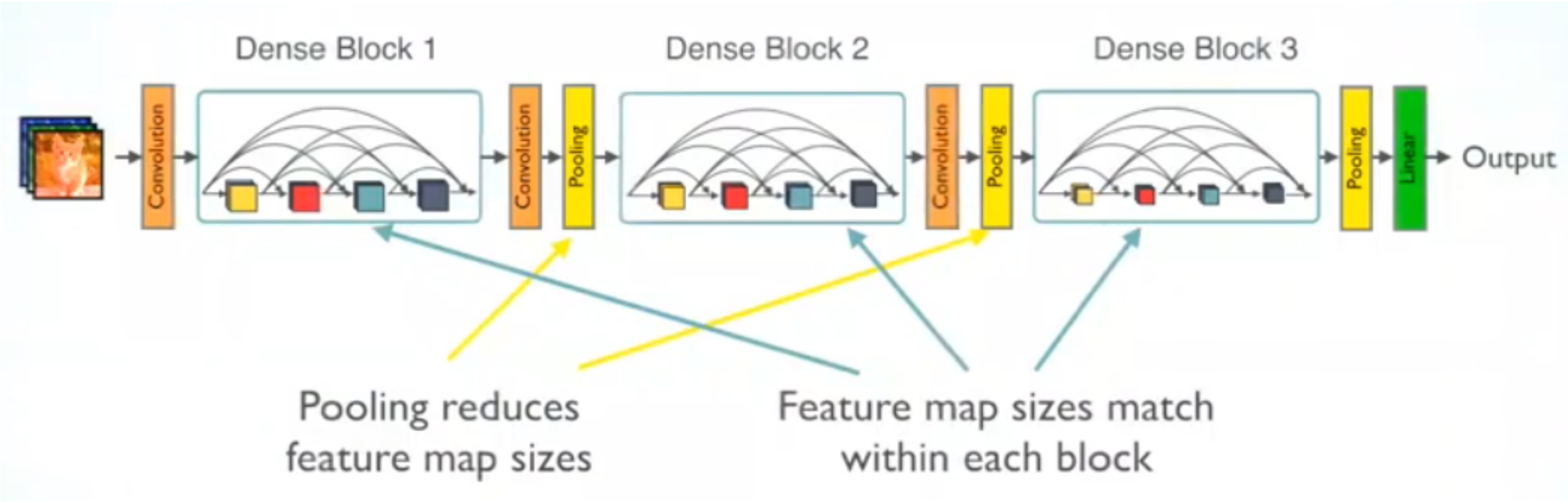
https://arxiv.org/pdf/1608.06993v3.pdf

- DenseNets do not sum the output feature maps of the layer with the incoming feature maps but **concatenate** them:

- $a^{[l]} = g\left(\left[a^{[0]}, a^{[1]}, \dots, a^{[l-1]}\right]\right)$

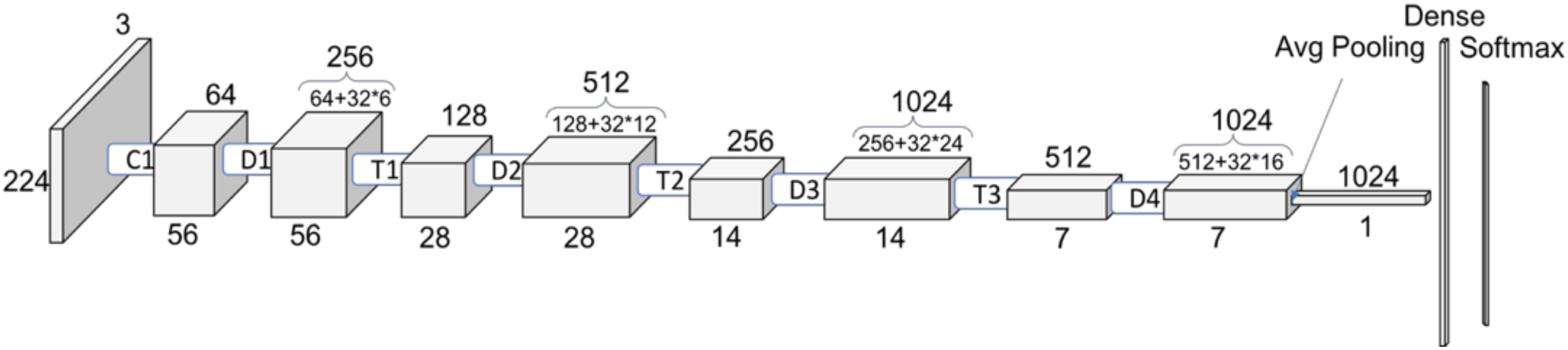
- Dimensions of the feature maps remains constant within a block, but the number of filters changes between them → **growth rate**:

- $k^{[l]} = k^{[0]} + k(l-1)$



https://towardsdatascience.com/review-densenet-image-classification-b6631a8ef803

SOTA Deep Models: DenseNets



SOTA Deep Models and Beyond

- MobileNetV2 and V3 (https://arxiv.org/abs/1801.04381, https://arxiv.org/abs/1905.02244)
- Inception-Resnet, v1 and v2 (https://arxiv.org/abs/1602.07261)
- Wide-Resnet (https://arxiv.org/abs/1605.07146)
- Xception (https://arxiv.org/pdf/1610.02357v2.pdf)
- ResNeXt (https://arxiv.org/pdf/1611.05431)
- ShuffleNet, v1 and v2 (https://arxiv.org/abs/1707.01083)
- Squeeze and Excitation Nets (https://arxiv.org/abs/1709.01507)
- Deep Pyramidal Residual Networks (https://arxiv.org/pdf/1610.02915.pdf)
- FractalNet (https://arxiv.org/pdf/1605.07648.pdf)

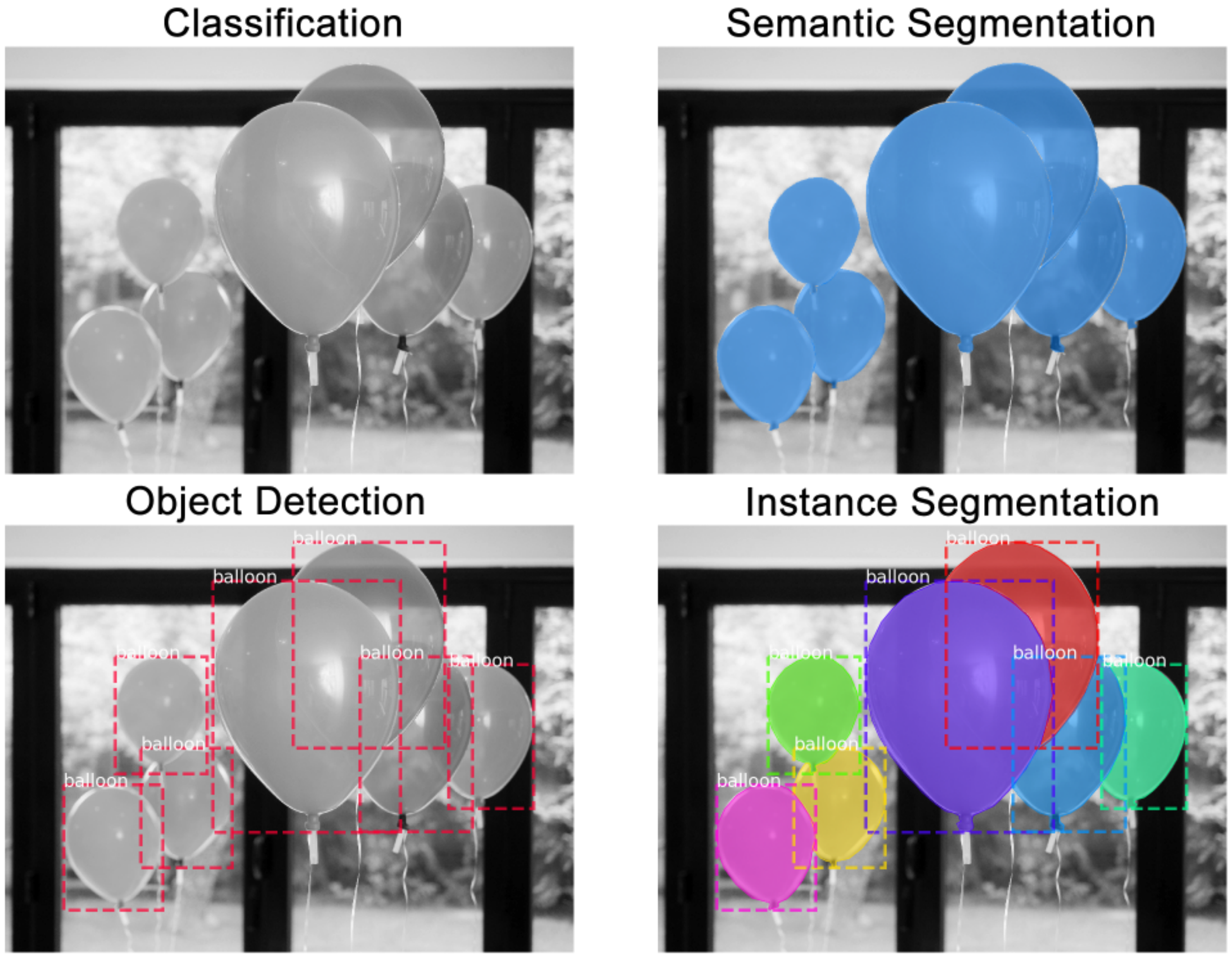
Outline

1: Communications and Recap

2: SOTA Deep Models

3: Transfer Learning across Tasks

Tasks


Classification
Semantic Segmentation
Object Detection
Instance Segmentation
balloon

Outline: Semantic Segmentation and Object Detection

Object Detection: let’s classify and locate

- Sliding Window versus Region Proposals
- Two stage detectors: the evolution of R-CNN , Fast R-CNN, Faster R-CNN
- Single stage detectors: detection without Region Proposals: YOLO / SSD

Semantic segmentation: classify every pixel

- Fully-Convolutional Networks
- SegNet & U-NET
- Faster R-CNN linked to Semantic Segmentation: Mask R-CNN

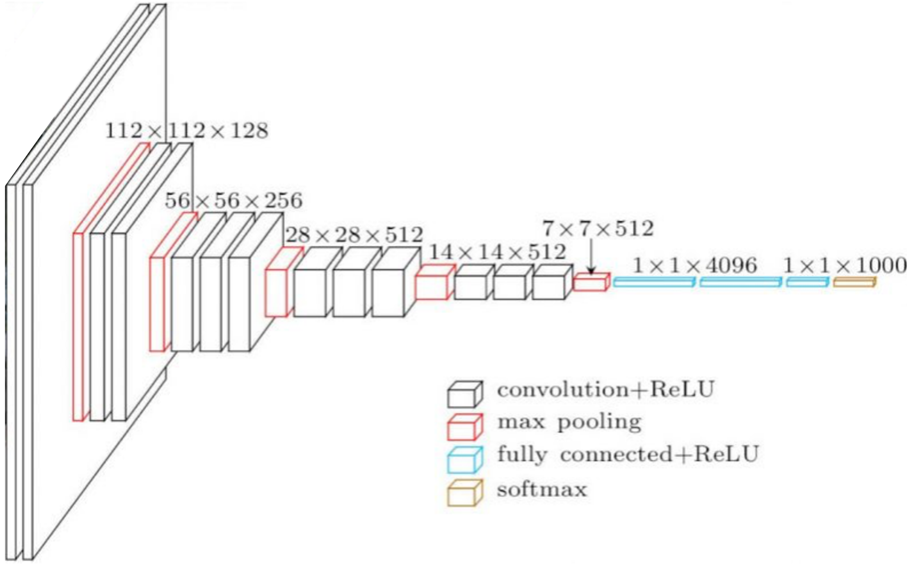
Demo: Using Transfer-Learning to train a U-NET

Tasks: Image Classification: Fully-Connected CNN

- Fundamental to computer vision given a set of labels {dog, cat, human, …};
- Predict the most likely class.

Input

VGG



Output

Classification (C = 1000):

- Dog: 0.95
- Cat: 0.02
- Human: 0.01
- …

Tasks: From Classification to Classification + Localization

- Localization demands to compute **where 1 object is present in an image;**
- Limitation: only 1 object (also non-overlapping);
- Typically implemented using a bounding box (x, y, w, h).

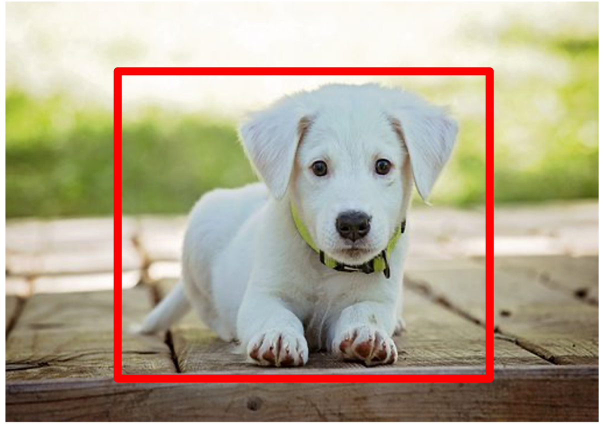
Predict

Classification Output:

- Dog: 0.95
- Cat: 0.02
- Human: 0.01
- ...

Output: Regular Image Classification

Predict



Classification output:

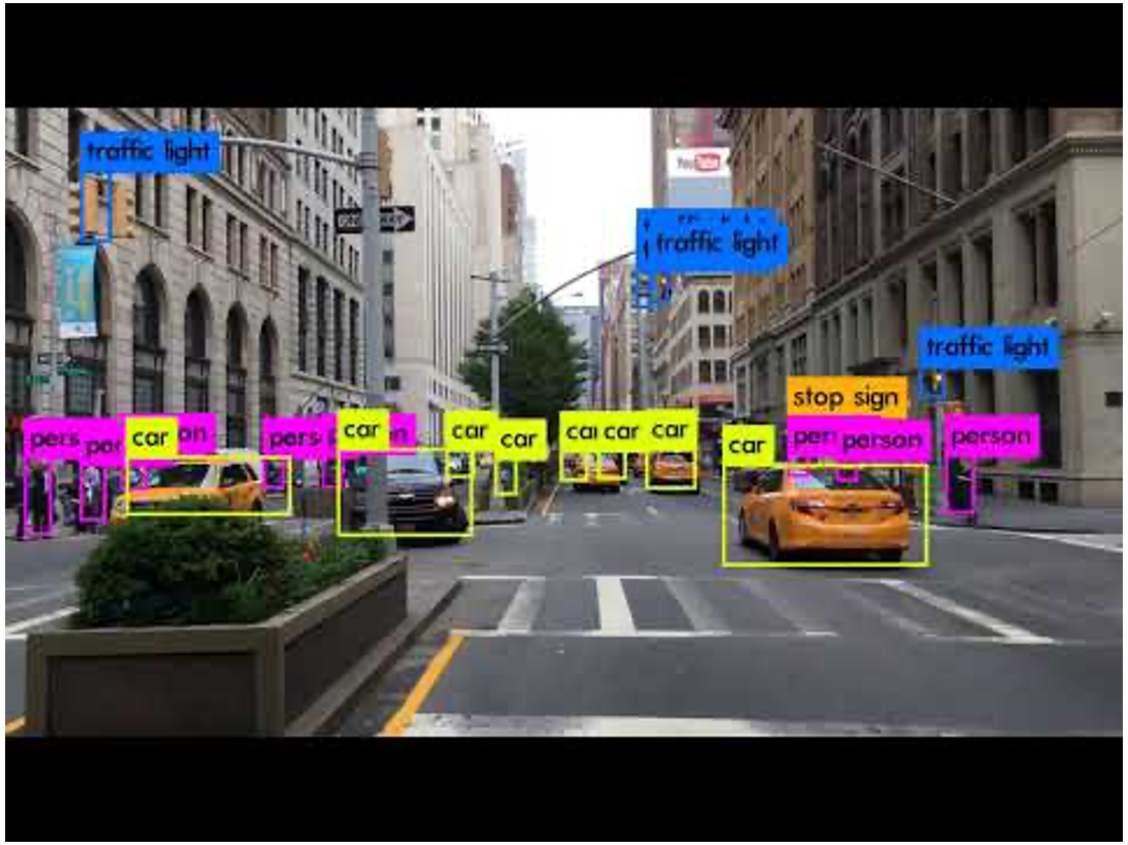
- Dog: 0.95
- Cat: 0.02
- Human: 0.01

Localization output:

- Bounding-Box: (x, y, w, h)

Tasks: From Classification + Localization to Object Detection

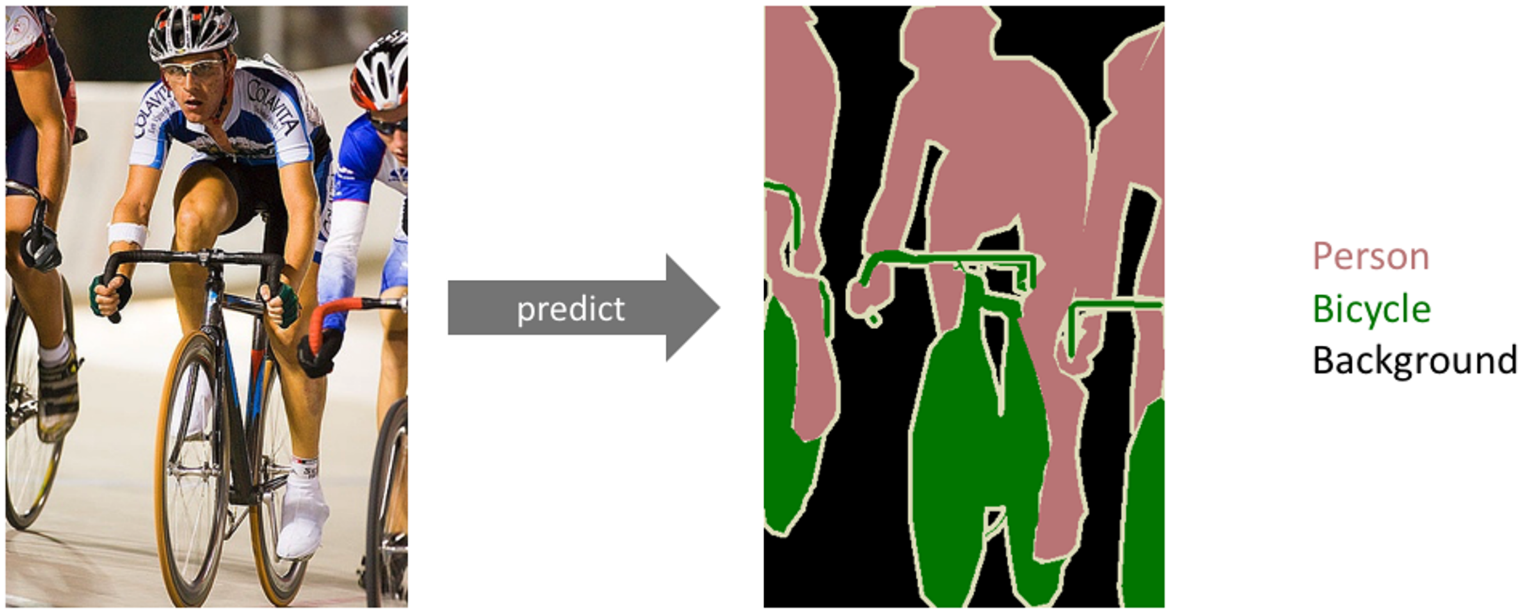
- Classification and Localization extended to multiple objects



Youtube 'YOLO in New York" by Joseph Redmon (creator of YOLO)

Tasks: From Classification to Semantic Segmentation

- **Image Classification:** assigning a single label to **the entire picture**
- **Semantic segmentation:** assigning a semantically meaningful label to **every pixel in the image**



Long, Shelhamer et al. "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015 : Cited by 14480

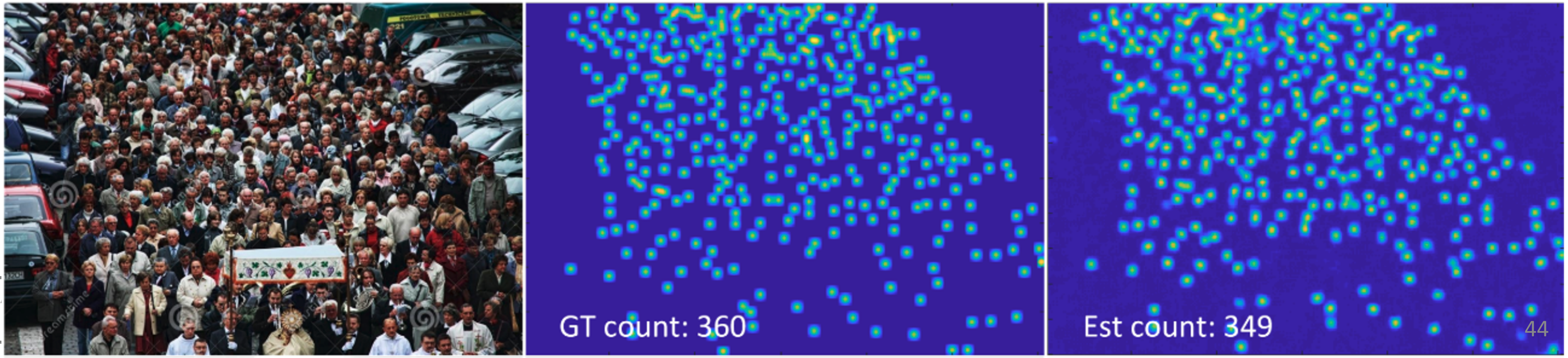
Why Object Detection and Semantic Segmentation

Computer vision:

- Autonomous vehicles
- Biomedical Imaging detecting cancer, diseases
- Video surveillance:
  - Counting people
  - Tracking people
- Aerial surveillance
- Geo Sensing: tracking wildfire, glaciers, via satellite

Note:

- Efficiency/inference-time is important!
- How many frames/sec. can we predict?
- Must for real-time segmentation & detection.



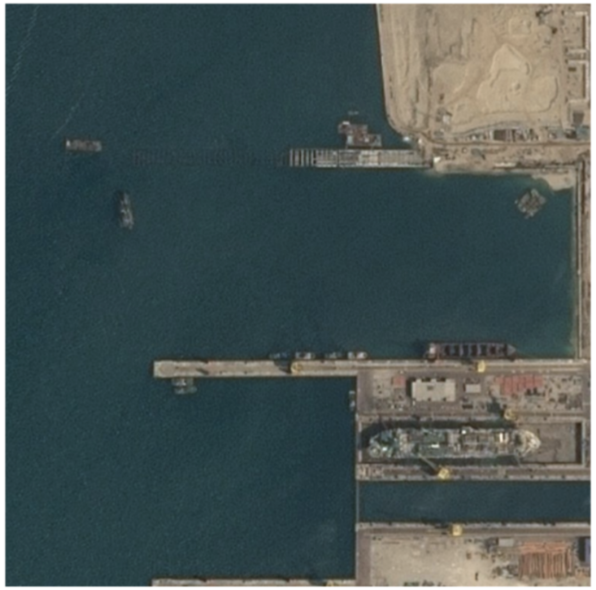
Why Object Detection and Semantic Segmentation

Youtube: "Tensorflow DeepLab v3 Xception Cityscapes"(link)

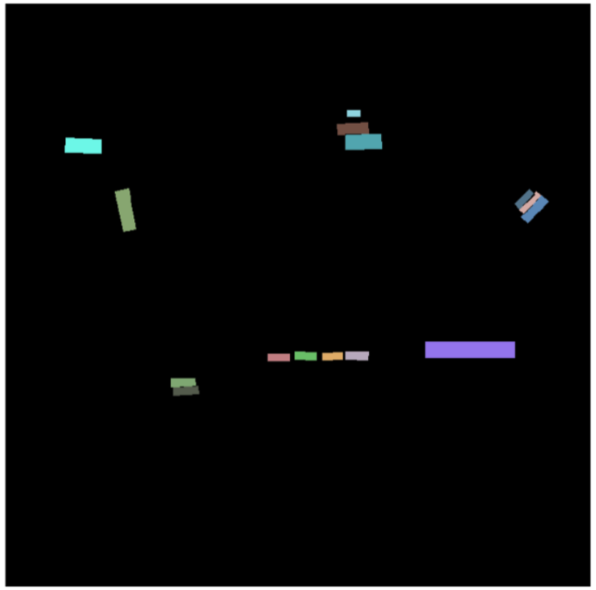
How to measure quality in detection and segmentation?

- **Pixel Accuracy:**
  - Percent of pixels in your image that are classified correctly
  - Our model has 95% accuracy! Great!

Input



Labels



Predict

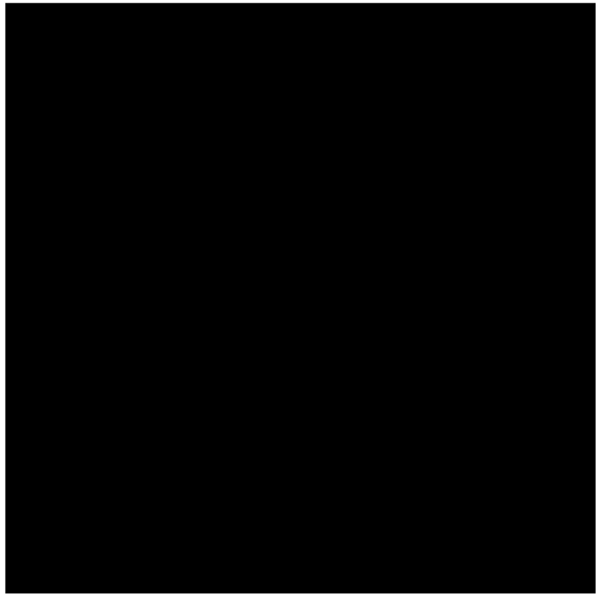


Image from Vlad Shmyhlo in article: Image Segmentation: Kaggle experience in TDS

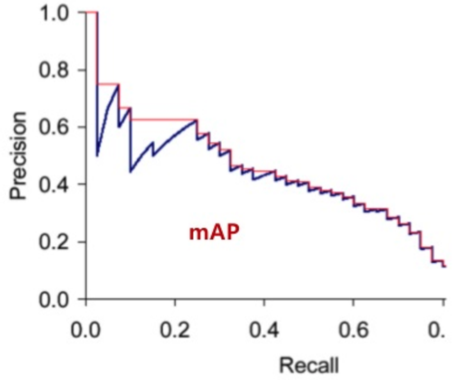
- Problem with accuracy: unbalanced data!

How do we measure accuracy?

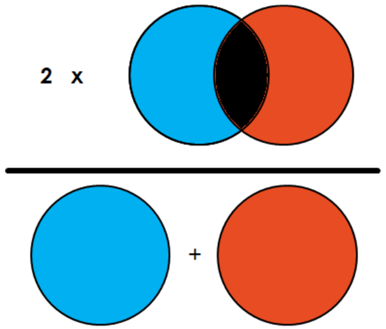
- **Pixel Accuracy**: Percent of pixels in your image that are classified correctly
- **IOU:** Intersection-Over-Union (Jaccard Index): Overlap / Union
- **DICE:** Coefficient (F1 Score): 2 x Overlap / Total number of pixels
- **mAP:** Mean Average Precision: AUC of Precision-Recall curve standard (0.5 is high)

IOU

mAP



DICE



Outline: Semantic Segmentation and Object Detection

Object Detection: let's classify and locate

- Sliding Window versus Region Proposals
- Two stage detectors: the evolution of R-CNN , Fast R-CNN, Faster R-CNN
- Single stage detectors: detection without Region Proposals: YOLO / SSD

Semantic segmentation: classify every pixel

- Fully-Convolutional Networks
- SegNet & U-NET
- Faster R-CNN linked to Semantic Segmentation: Mask R-CNN

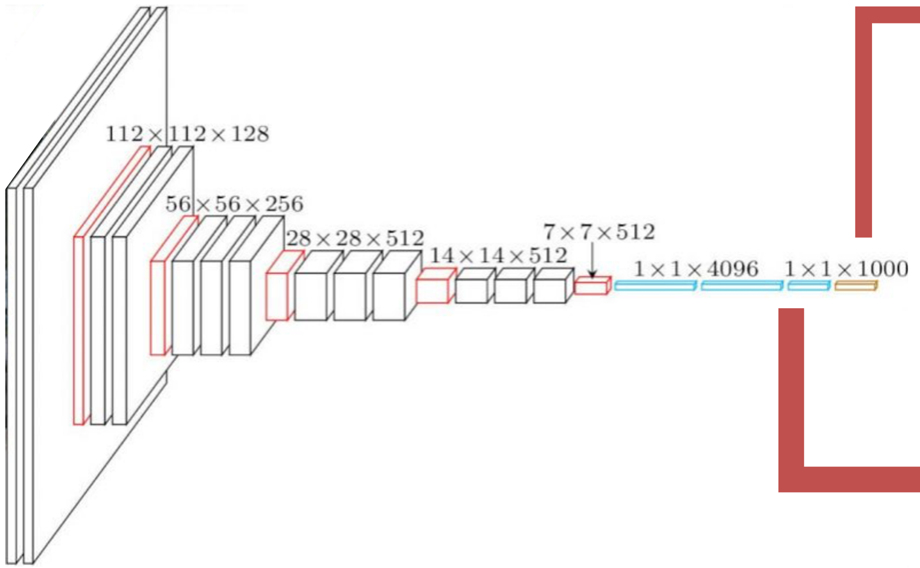
Demo: Using Transfer-Learning to train a U-NET

Object detection: let's classify and locate

- Object detection is just classification and localization combined:
    - Classification using standard CNN;
    - Localization using regression problem for predicting box coordinates
    - Combining loss from Classification (Softmax) and Regression (L2)

Input

VGG



Classification:
- Dog: 0.95
- Cat: 0.02
- Human: 0.01
- …

Bounding-Box :
- x
- y
- width (w)
- height (h)

Output

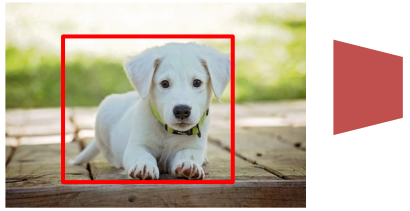
Softmax Loss

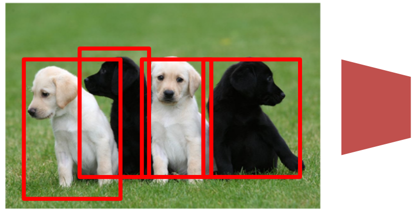
Multi-Task Learning

L2-norm Loss

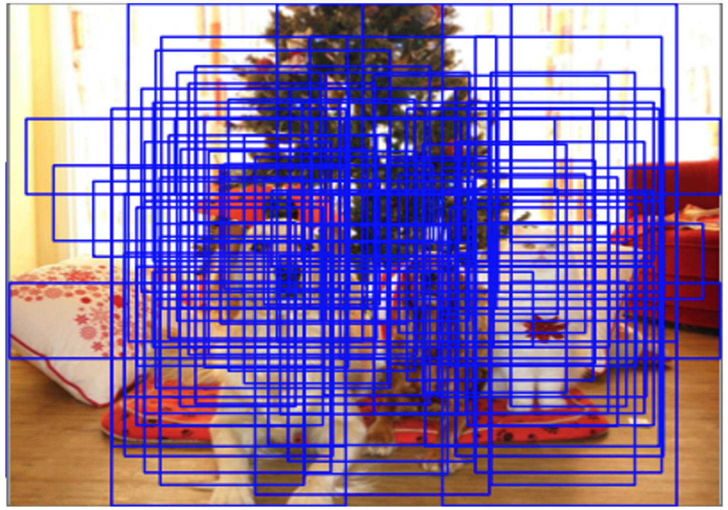
Sliding Windows, from single to multiple objects

- Might work for single object, but not for multiple objects
- Each image containing “x” objects: needs “x” number of classification and localization outputs
- Solution for multiple objects:
  - Crop the image “in a smart way”
  - Apply the CNN to each crop





- Can we just use sliding windows?
  - Problem: Need for applying CNN to huge number of locations, scales, bbox aspect ratios: very computationally expensive;
  - Solution: Region Proposals methods to find object-like regions.



Adapted from Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 “Convolutional Neural Networks for Visual Recognition” Lecture 12 Slide 37
Uijlings et al, Selective Search for Object Recognition” IJCV 2013 [link]

Object detection: Region Proposal Networks!

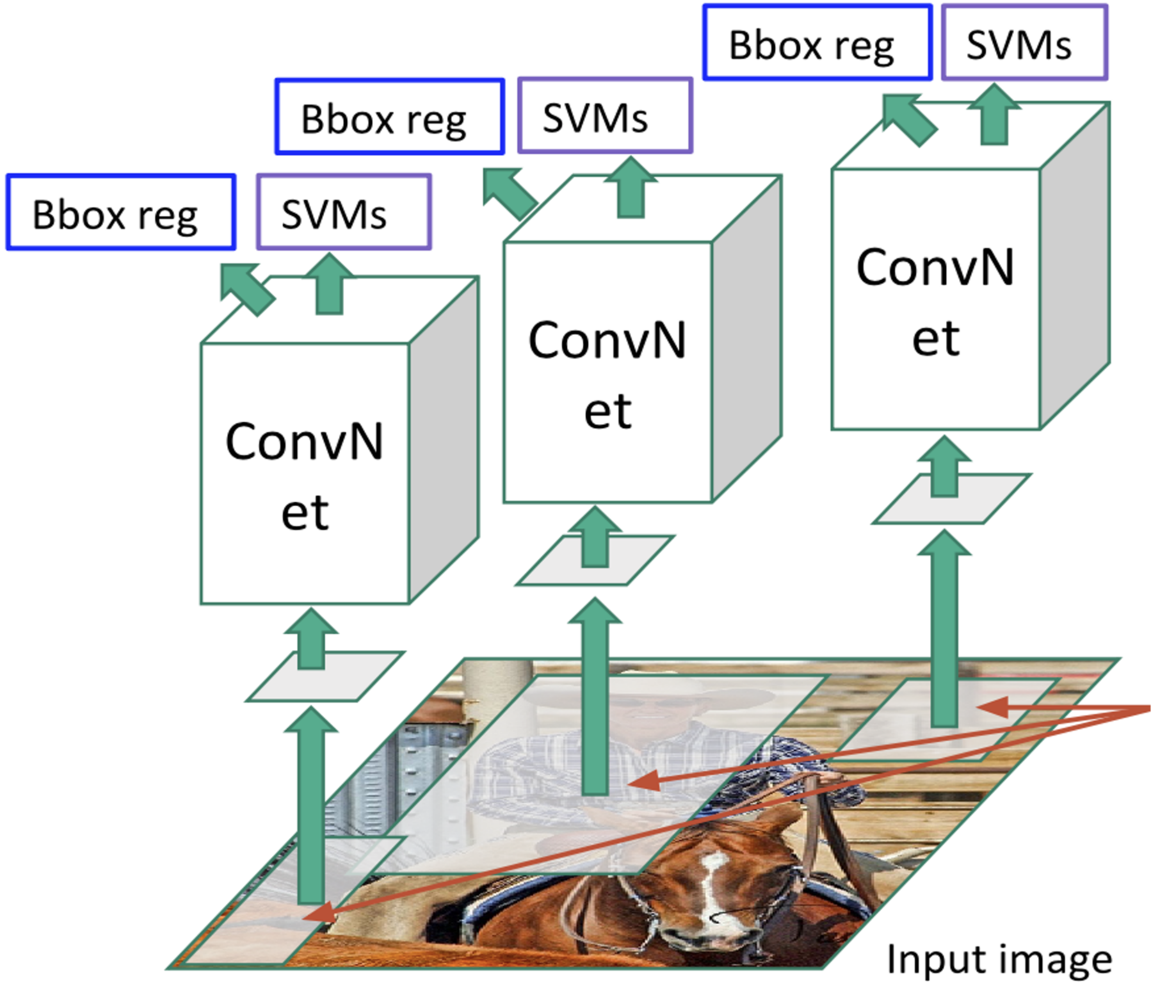
- **Problem:** Need for applying CNN to huge number of locations, scales, bbox aspect ratios, very computationally expensive!
- **Solution:** Region Proposals methods to find object-like regions:
- **Selective Search Algorithm:** returns boxes that are likely to contain objects:
  - Use hierarchical segmentation;
  - Start with small superpixels;
  - Merge based on similarity.
- **Output:** Where are object like regions?
  - No classification yet.

Uijlings et al, Selective Search for Object Recognition" IJCV 2013 link

The Evolution of R-CNN: R-CNN, Fast R-CNN, Faster R-CNN

- **R-CNN = Region-based CNN**
- Correct BBox by Bbox regressor (dx,dy,dw,dh)
- Forward each region through CNN
- Resize proposed RoI (224x224)

- Region of Interest (RoI) from selective search region proposal (approx 2k)
- **Problem:** need to do 2k independent forward passes for each image! **('slow' R-CNN)**



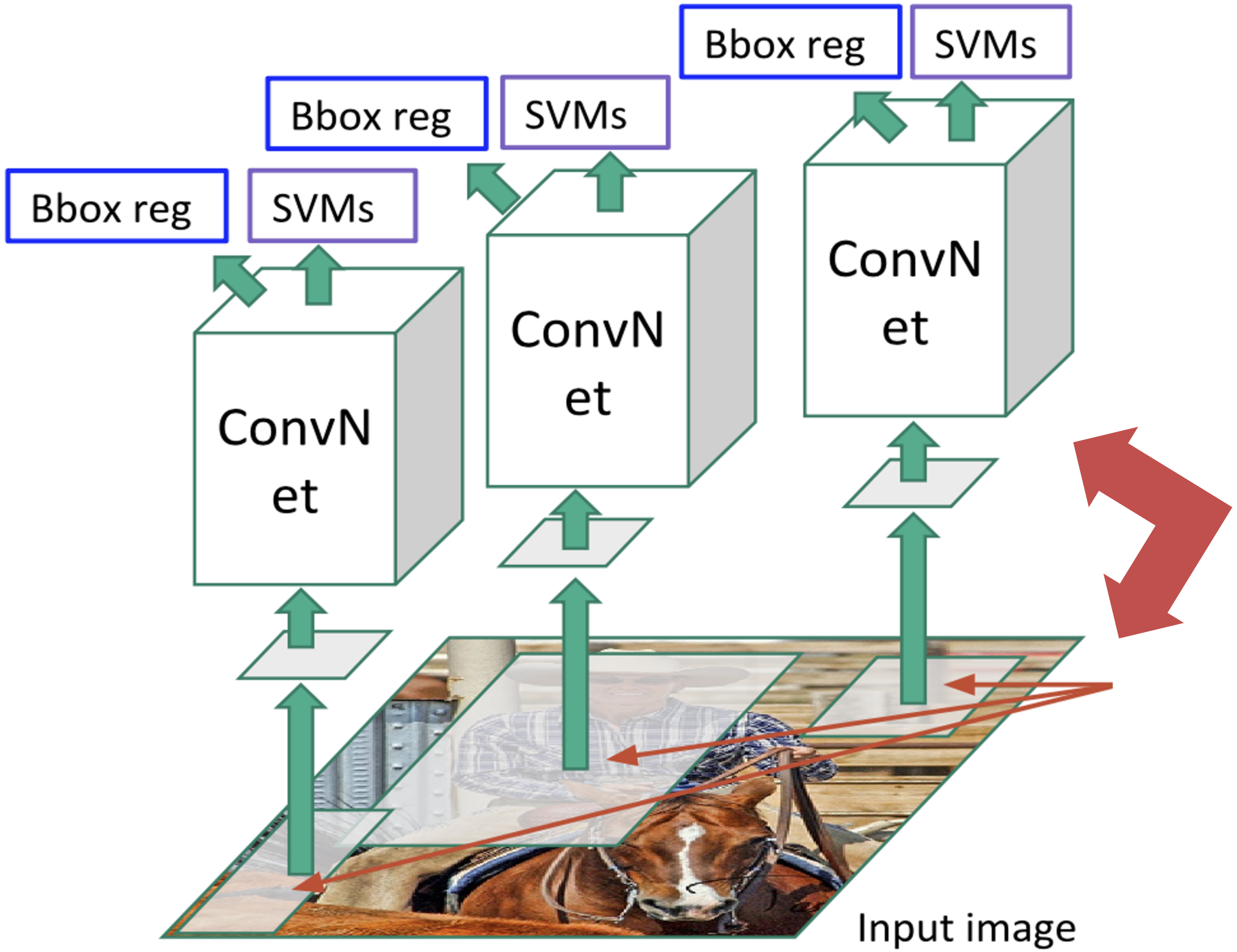
Adapted from Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 "Convolutional Neural Networks for Visual Recognition"
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation" CVPR2014
Ross Girshick, "Fast R-CNN" Slides 2015

The Evolution of R-CNN: R-CNN, Fast R-CNN, Faster R-CNN

- **R-CNN = Region-based CNN**
- Correct BBox by Bbox regressor (dx,dy,dw,dh)
- Forward each region through CNN
- Resize proposed RoI (224x224)

- Region of Interest (RoI) from selective search region proposal (approx 2k)
- **Problem:** need to do 2k independent forward passes for each image! **('slow' R-CNN)**

- **Solution:** can we process the image before cropping?

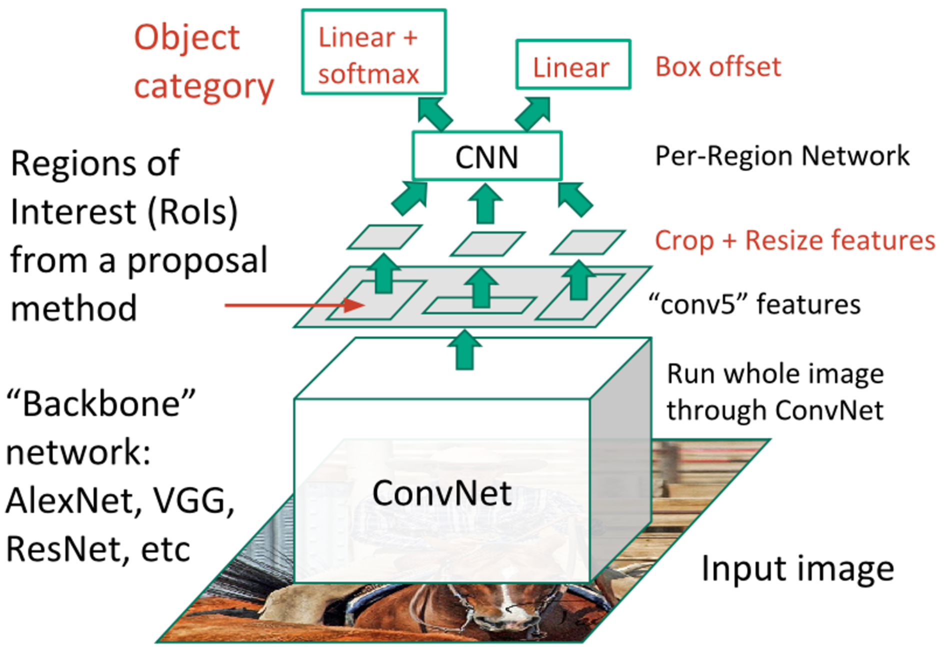


Adapted from Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 "Convolutional Neural Networks for Visual Recognition"
Girshick et al, "Rich feature hierarchies for accurate object detection and semantic segmentation" CVPR2014
Ross Girshick, "Fast R-CNN" Slides 2015

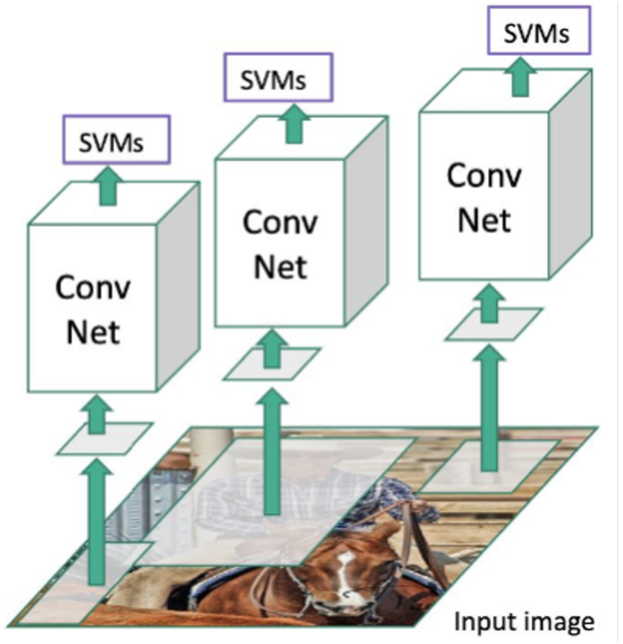
The Evolution of R-CNN: R-CNN, Fast R-CNN, Faster R-CNN

- **Problem:** need to do 2k independent forward passes for each image! **('slow' R-CNN)**
- Even inference is slow: 47s/image with VGG16 [Simonyan & Zisserman, ICLR 15]
- **Solution:** can we process (CNN forward pass) the image before cropping generates 2k regions?

Fast R-CNN



Slow R-CNN

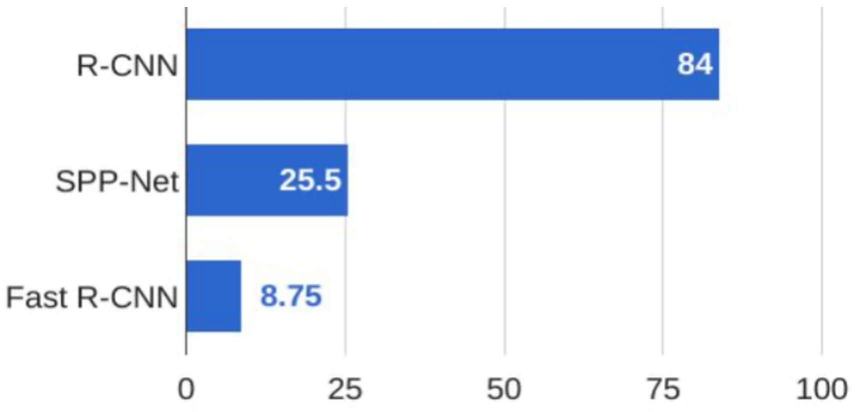


Adapted from Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 "Convolutional Neural Networks for Visual Recognition"
Ross Girshick, "Fast R-CNN" Slides 2015

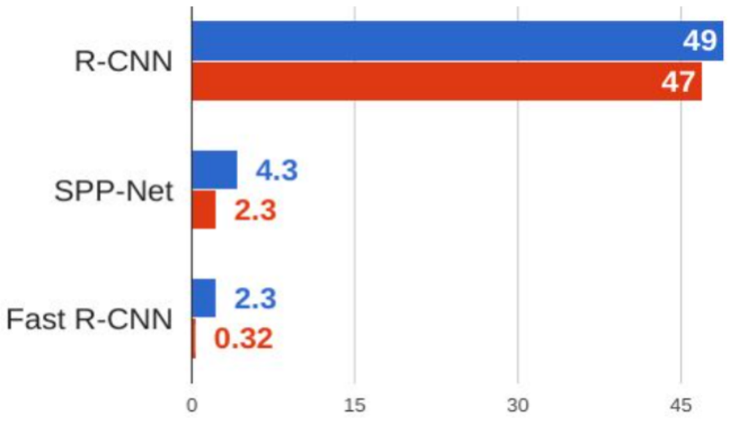
The Evolution of R-CNN: R-CNN, Fast R-CNN, Faster R-CNN

- Fast R-CNN much faster than R-CNN;
- Runtime dominated by region proposals; is an iterative method ('like selective search');
- **Solution:** can we make CNN do proposals!

Training Time (Hours)



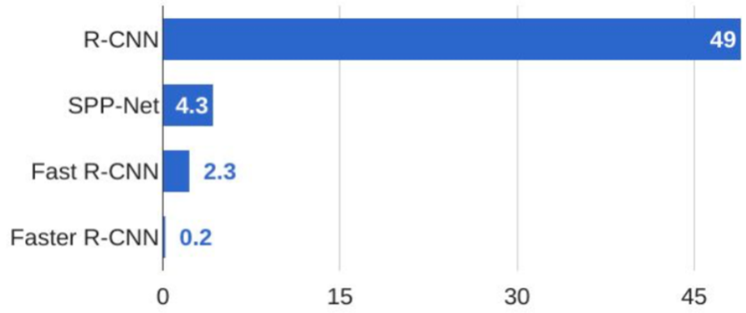
Test Time (Seconds)

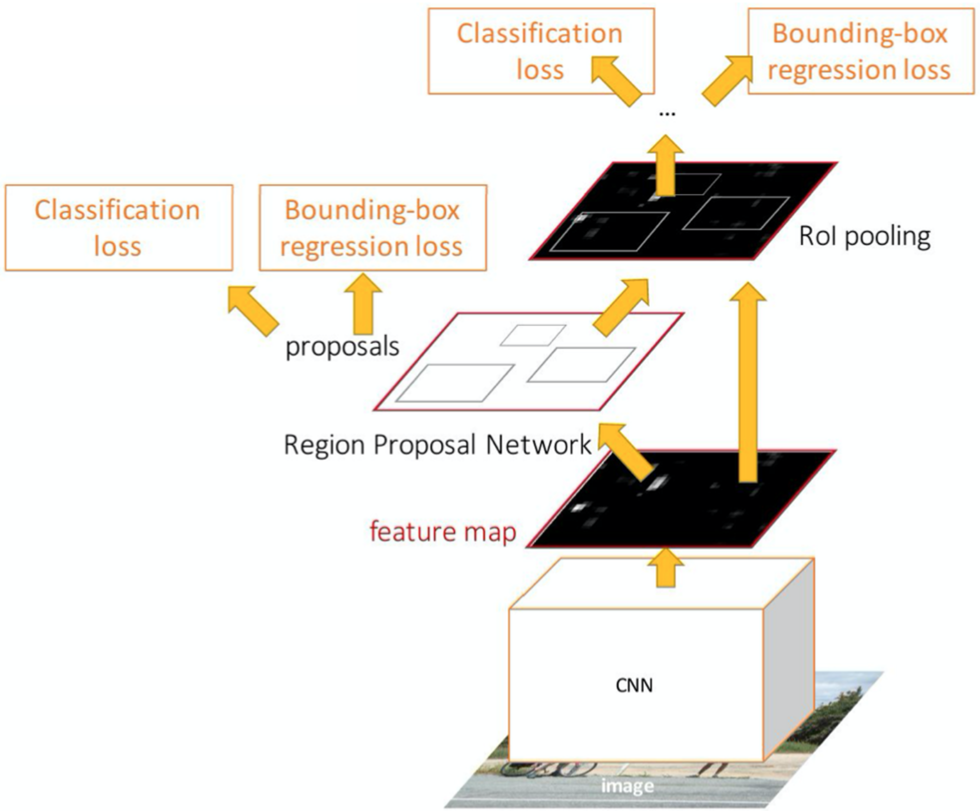


Adapted from Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 "Convolutional Neural Networks for Visual Recognition"
Ross Girshick, "Fast R-CNN" Slides 2015

The Evolution of R-CNN: R-CNN, Fast R-CNN, Faster R-CNN

- **Faster R-CNN**: make CNN to do proposals! (single forward, not iterative selective search)
- **CNN Region Proposal Network (RPN)**: predicting region proposals from features
- Otherwise same as Fast R-CNN: crop and classify
- End-to-end quadruple loss:
    - RPN classify object / not object
    - RPN regress box coordinates
    - Final classification score (object classes)
    - Final box coordinates
- Test-time seconds per image:

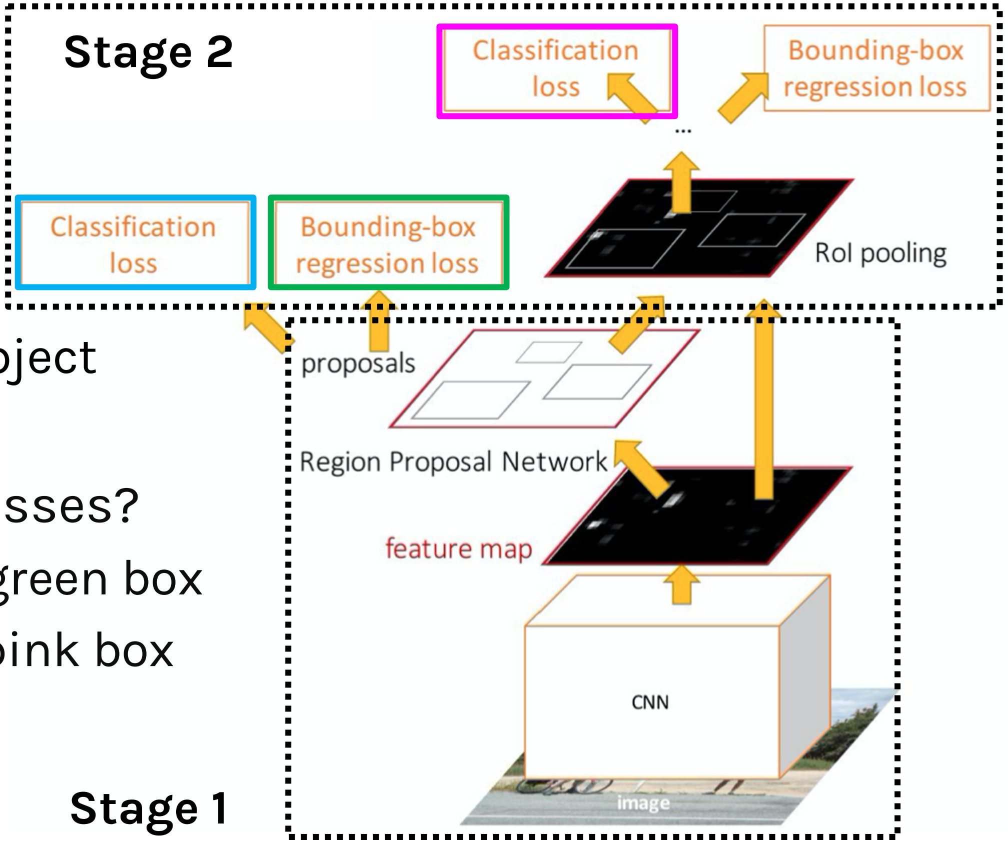




Adapted from Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 "Convolutional Neural Networks for Visual Recognition"
Ross Girshick, "Fast R-CNN" Slides 2015

The Evolution of R-CNN: R-CNN, Fast R-CNN, Faster R-CNN

- Previously we said: “Multiple objects? Thus Need for Region Proposal Networks!”
- **Faster R-CNN is a two-stage object detector**
  - **Stage 1:** backbone network + RPN (once/image)
  - **Stage 2:** crop - predict object & bbox (once/region)
- What is our RPN again?
- RPN runs prediction on many many anchor boxes:
  - **Loss 1:** Tells is does the anchor bbox contain an object
  - **Loss 2:** For the top 300 boxes its adjusts the box
- What is the difference between our 2 classification losses?
  - one is classifying **object** (i.e. object/not object) – green box
  - one is classifying specific **categories** (e.g. dog) – pink box
- Do we really need two stages?

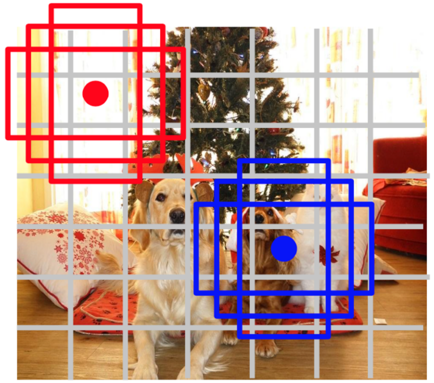


Adapted from Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 “Convolutional Neural Networks for Visual Recognition”
Ross Girshick, “Fast R-CNN” Slides 2015

Single-Stage Detection without Region Proposals: YOLO, SSD

- Within each of the **NxN** grid regress over each **B** base boxes, predict: (x,y,h,w, confidence = 5)
- **Predict C** category specific class scores
  - Output : N x N x S (5 B + C)
- YOLOv3 (Joseph Redmon):
  - predicts at 3 scales, S = 3
  - predicts 3 boxes at each scale, B=3
  - Darknet-53 as feature extractor (similar to ResNet 152, and 2x faster!)

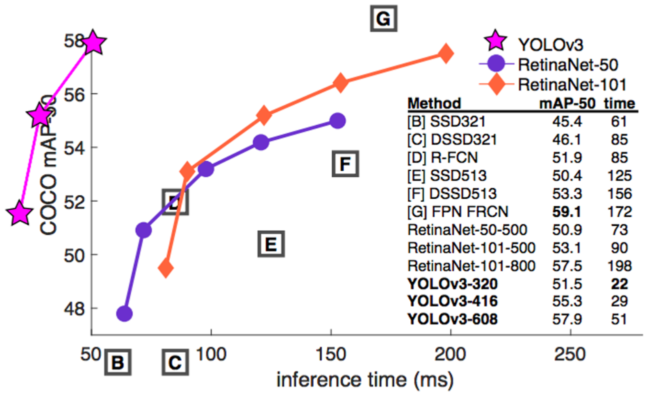
Input image
3 x H x W



Divide image into grid
7 x 7

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

Redmon et al, "You Only Look Once: Unified, Real-Time Object Detection", CVPR 2016
Liu et al, "SSD: Single-Shot MultiBox Detector", ECCV 2016
Lin et al, "Focal Loss for Dense Object Detection", ICCV 2017

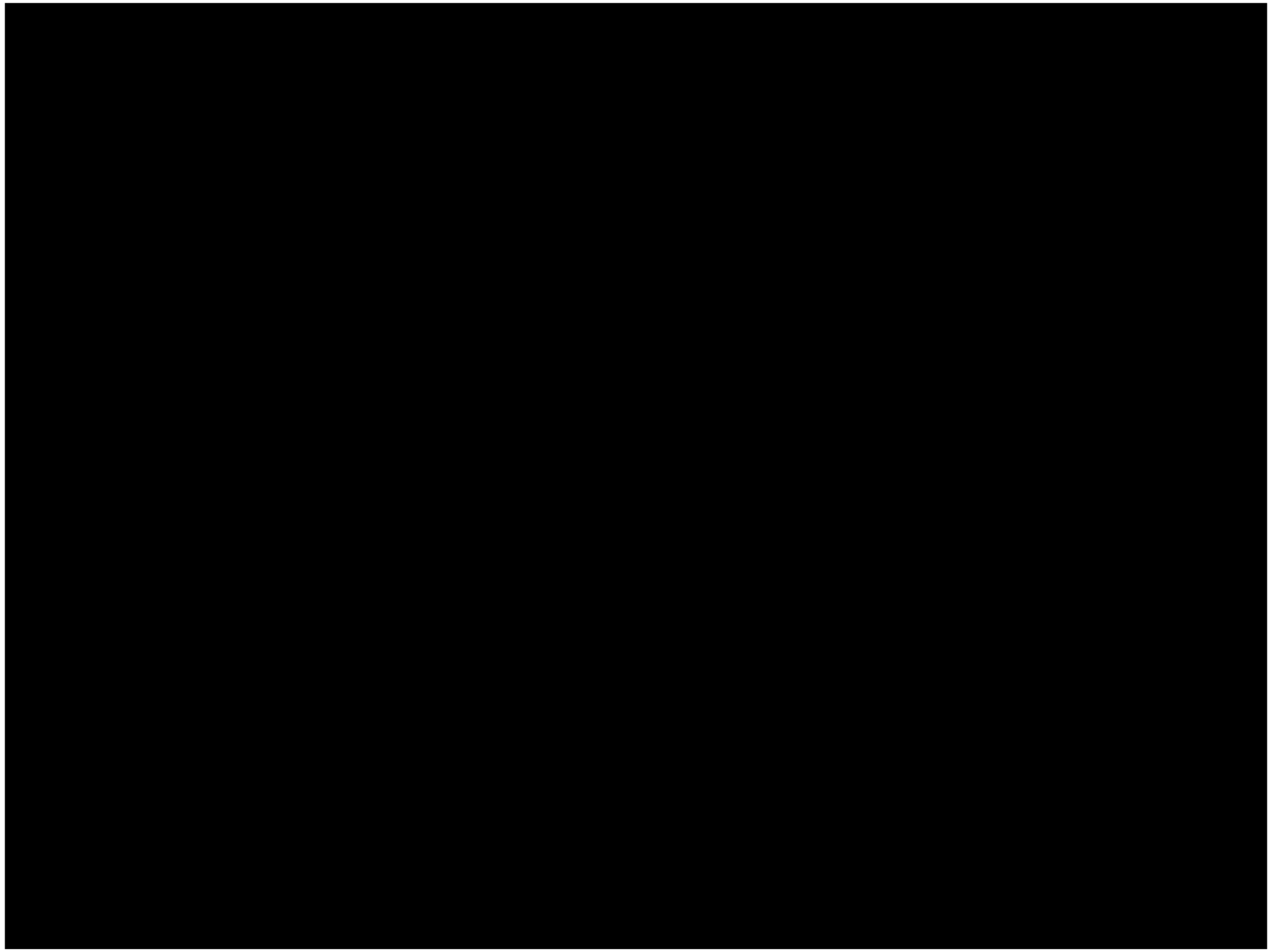


Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

(YOLO) Redmon, "You Only Look Once: Unified, Real-Time Object Detection" CVPR 2015: Cited by 8057 (link)

Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019 "Convolutional Neural Networks for Visual Recognition"

109a versus 295



Outline: Semantic Segmentation and Object Detection

Object Detection: let’s classify and locate

- Sliding Window versus Region Proposals
- Two stage detectors: the evolution of R-CNN , Fast R-CNN, Faster R-CNN
- Single stage detectors: detection without Region Proposals: YOLO / SSD

Semantic segmentation: classify every pixel

- Fully-Convolutional Networks
- SegNet & U-NET
- Faster R-CNN linked to Semantic Segmentation: Mask R-CNN

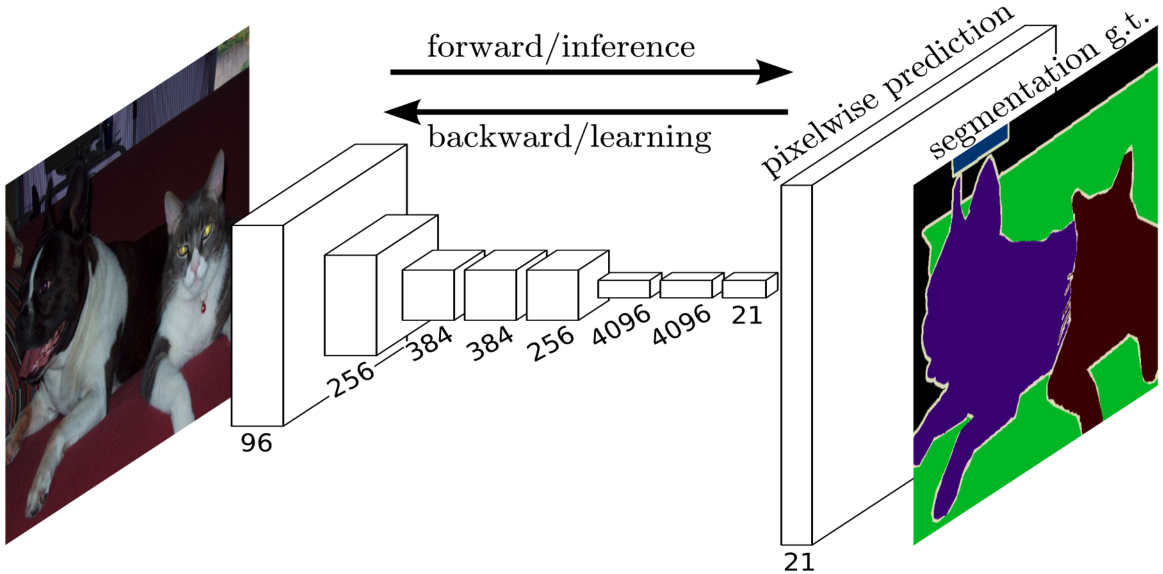
Demo: Using Transfer-Learning to train a U-NET

Semantic segmentation: Classify every pixel

- **Image Classification:** assigning a single label to **the entire picture**
- **Semantic segmentation:** assigning a semantically meaningful **label to every pixel in the image**

So our output shouldn't be a class prediction (C numbers) but a picture (C x *w* x *h*)

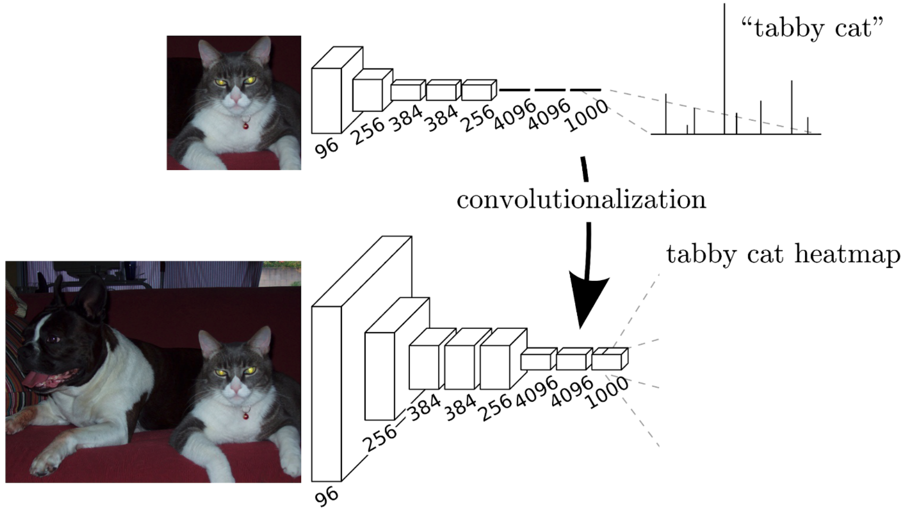
- Can we have a network for each pixel location?
- Sliding window inputs of patches predicting the class of the pixel in the center?
- Many forward passes! Not reusing overlapping patches and features.



(FCN) Long, Shelhamer et al. "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015: Cited by 14480 (link)

Fully-Convolutional Networks

- Semantic segmentation: assigning a semantically meaningful label to every pixel in the image
- So our output shouldn't be a classification prediction (C numbers) but a picture (C x w x h)
  - Maybe we can have a network for each pixel location? Many (w times h) networks!
  - Sliding window inputs of patches predicting the class of the pixel in the center? Many forward passes! Overlapping features not used.
- Solution: FCN = Fully-Convolutional Networks! (not fully-connected)
  - 1 network - 1 prediction would be a lot better
  - Why convolutions? every pixel is very much influenced by its neighborhood



(FCN) Long, Shelhamer et al. "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015: Cited by 14480 (link)

Fig: top, Image Classification (FC), bottom, Image Segmentation (FCN)

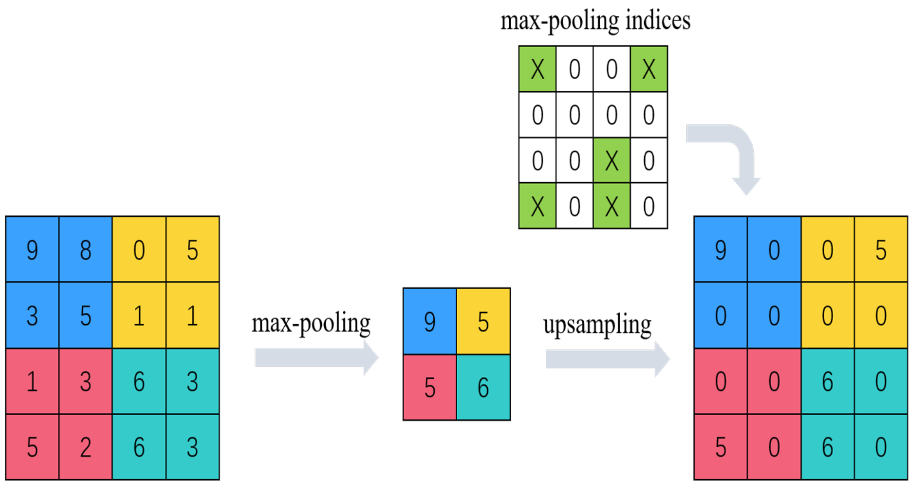
Fully-Convolutional Networks

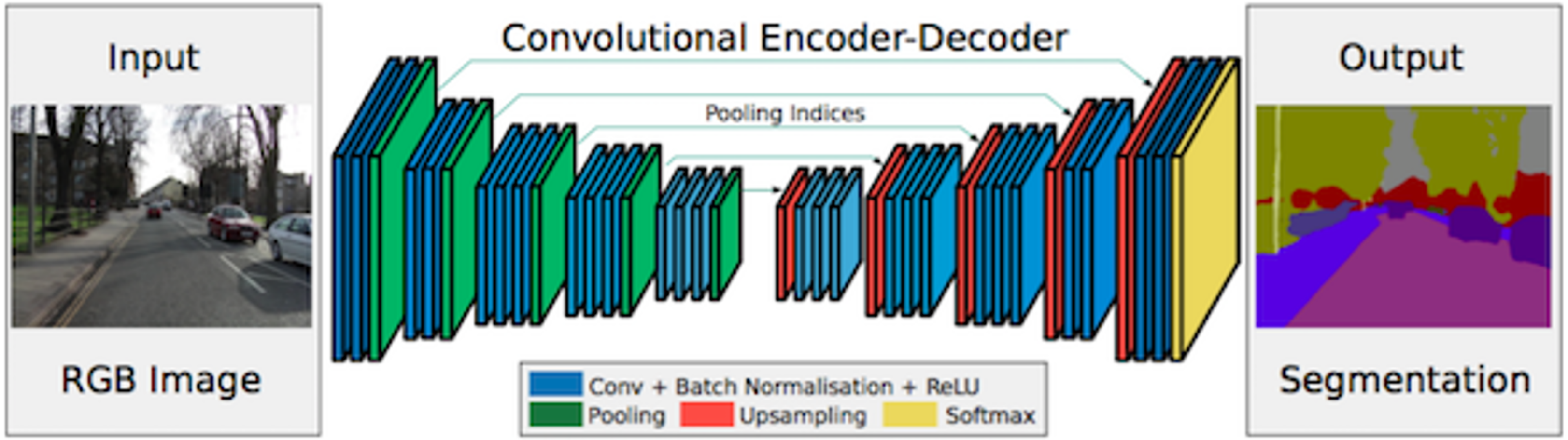
- **FCN:** design a network as a bunch of conv layers to make predictions for all pixels all at once.
  - **Encoder** (= Localization)**:** **downsample** through **convolutions.** Reduces number of params (bottleneck), can make network deeper
  - **Decoder** (= Segmentation): **upsampled** through **transposed convolutions**
  - **Loss:** cross-entropy loss on every pixel.
- **Contribution:**
  - Popularize the use of end-to-end CNNs for semantic segmentation;
  - Re-purpose imagenet pretrained networks for segmentation = Transfer Learning
  - Upsample using transposed layers.
- **Negative:**
  - upsampling = loss of information during pooling;
  - 224x224 image downsampled to 20x20 back upsampled to 224x224.

(FCN) Long, Shelhamer et al. "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015; Cited by 14480

SegNet

- The indices from max pooling down sampling are transferred to the decoder: **pooling indices**
- Improves fine segmentation resolution, we want “pixel-perfect”;
- More efficient since no transposed convolutions to learn.

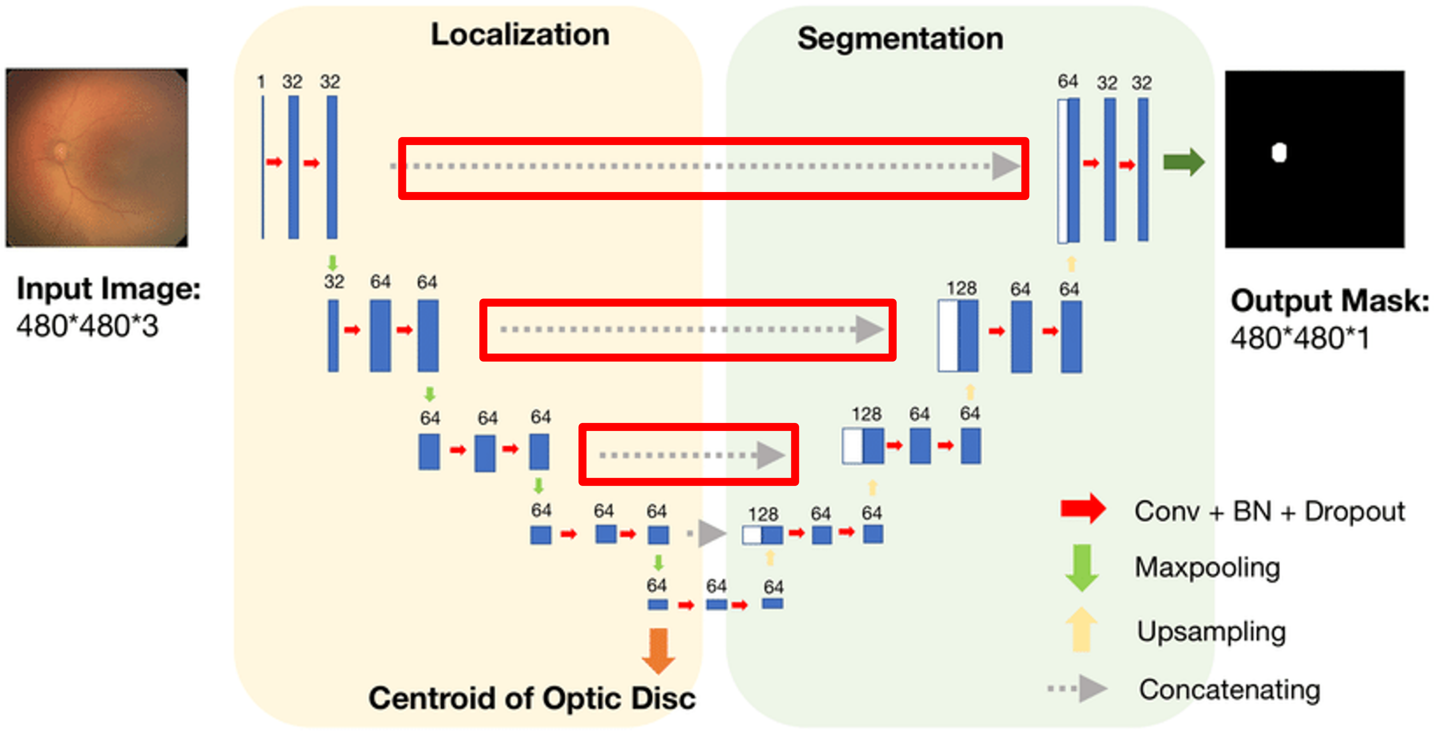




SegNet: A deep Convolutional Encoder-Decoder Architecture for Image Segmentation. (link)

U-NET: long skip connections

- The U-Net is an encoder decoder using:
  - **location information** from the down sampling path of the encoder;
  - **contextual information** in the up sampling path by the "concatenating" long-skip connections.



Outline: Semantic Segmentation and Object Detection

Object Detection: let’s classify and locate

- Sliding Window versus Region Proposals
- Two stage detectors: the evolution of R-CNN , Fast R-CNN, Faster R-CNN
- Single stage detectors: detection without Region Proposals: YOLO / SSD

Semantic segmentation: classify every pixel

- Fully-Convolutional Networks
- SegNet & U-NET
- Faster R-CNN linked to Semantic Segmentation: Mask R-CNN

Demo: Using Transfer-Learning to train a U-NET

References

Presentations:

- Fei-Fei Li & Justin Johnson & Serena Yeung Stanford CS231n 2019/2018 "Conv. Neural Networks for Visual Recognition" Lecture 12 !
    - BTW: Great course / youtube series (youtube 2017)
- Ross Girshick, "Fast R-CNN" Slides 2015 (link)

Papers:

- **VGG** Simonyan, Zisserman. "Very Deep CNNs for Large-scale Image Recognition", ILSVRC 2014: Cited by 34652 (link)
- **Select. Search** Uijlings et al, Selective Search for Object Recognition" IJCV 2013: Cited by 3944 (link)
- **R-CNN** Girshick et al, "Rich feature hierarchies for accurate object detect. & sem. segmentation" CVPR2014: Cited by 12000 (link)
- **Fast-R-CNN** Girshick, 'Fast R-CNN" ICCV 2015: Cited by 8791 (link)
- **Faster- R-CNN** Ren et al, "Faster R-CNN: Real-Time Object Det. with Region Proposal Networks" NEURIPS 2015 Cited by 16688 (link)
- **Mask-R-CNN** He et al, "Mask R-CNN" ICCV 2017: Cited by 5297 (link)
- **YOLO** Redmon, "You Only Look Once: Unified, Real-Time Object Detection" CVPR 2015: Cited by 8057 (link)
- **FCN** Long, Shelhamer et al. "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015: Cited by 14480 (link)
- **SegNet** Badrinarayanan et al. "SegNet: A deep Conv Encoder-Decoder Architecture for Image Segmentation". Cited by 4258 (link)
- **U-Net** Ronneberger et al. "U-Net: Convolutional Networks for Biomedical Image Segmentation". Cited by 12238 (link)

THANK YOU