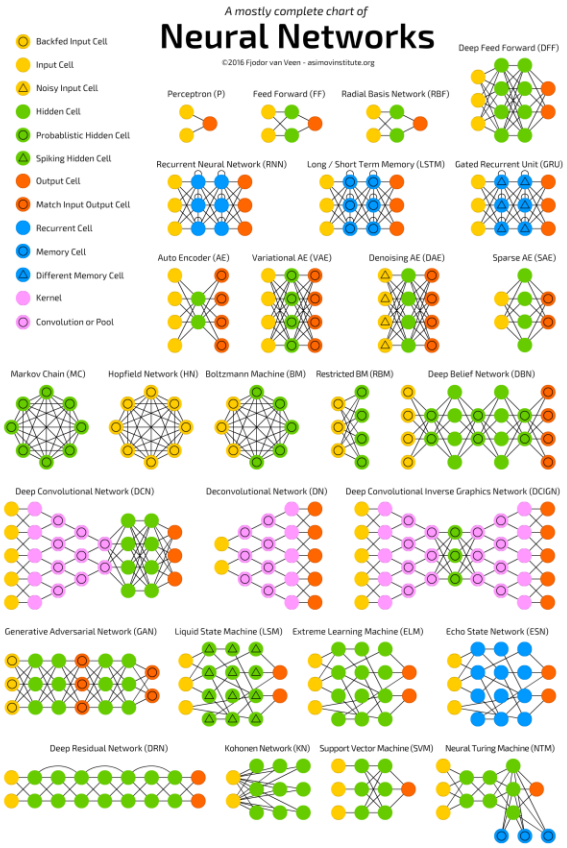
TDS3651
Visual Information Processing



Deep Learning Foundations for Computer Vision (Part 2)

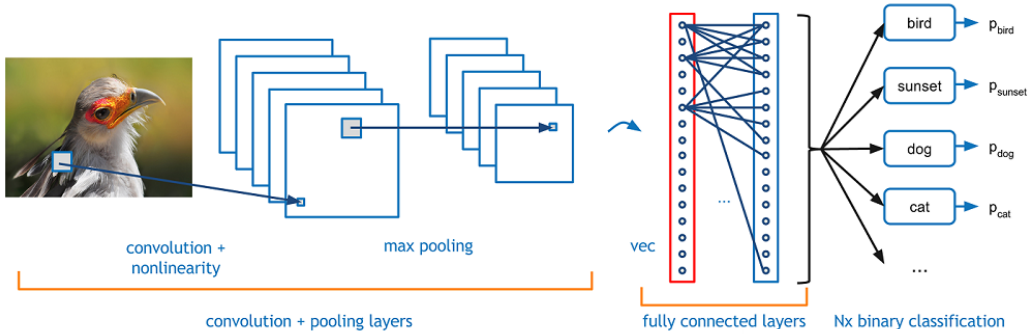
Lecture 12

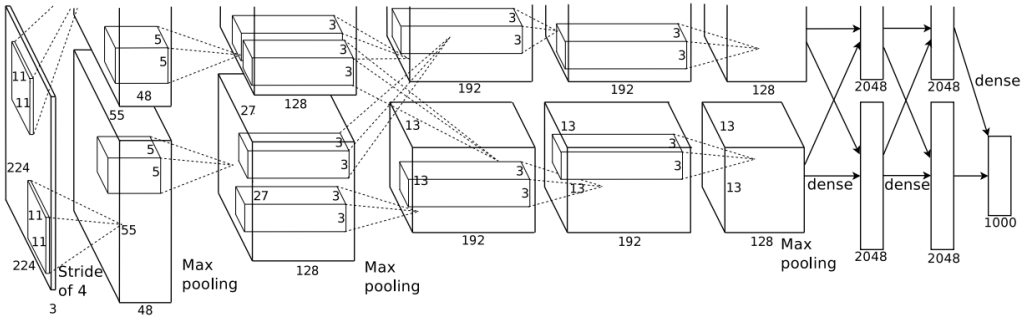
Faculty of Computing and Informatics
Multimedia University
created by Lai-Kuan, Wong
modified by Yuen Peng, Loh

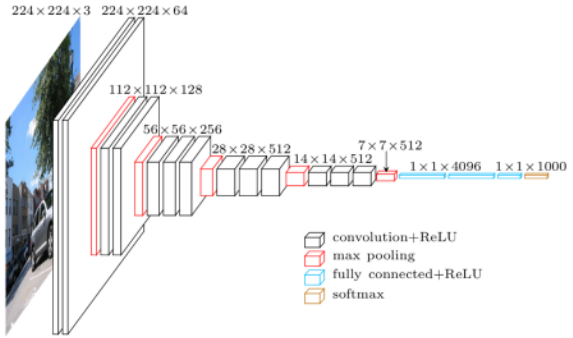
Course Outline

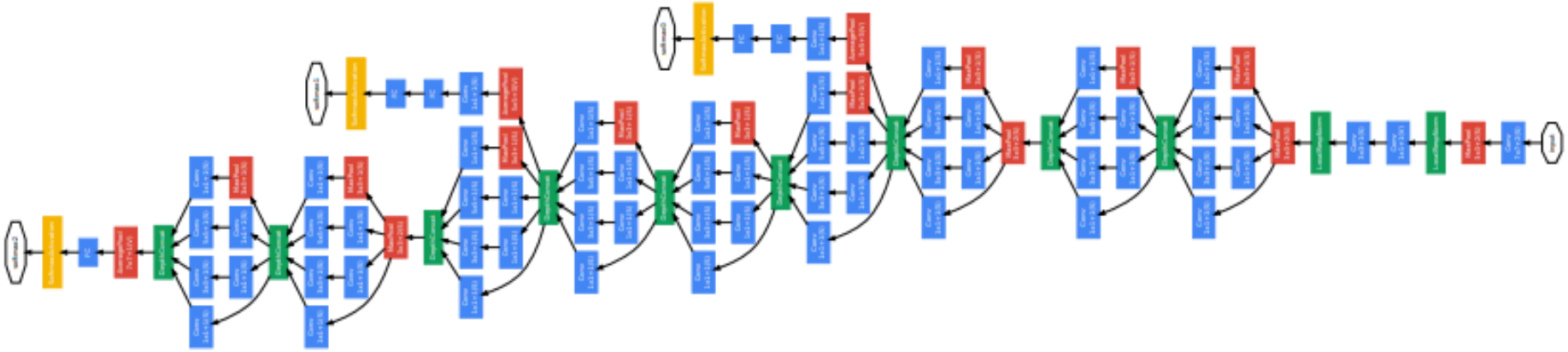
- Transformers for Computer Vision
- Segmentation, Localization & Detection
  - Object Segmentation
    - Upsampling
    - Deconvolution
  - Object Localization and Detection
    - Region proposal
- Boosting Deep Learning Performance
- Further Studies and References

Recap


bird
sunset
dog
cat
p_{bird}
p_{sunset}
p_{dog}
p_{cat}
convolution + nonlinearity
max pooling
vec
convolution + pooling layers
fully connected layers
Nx binary classification

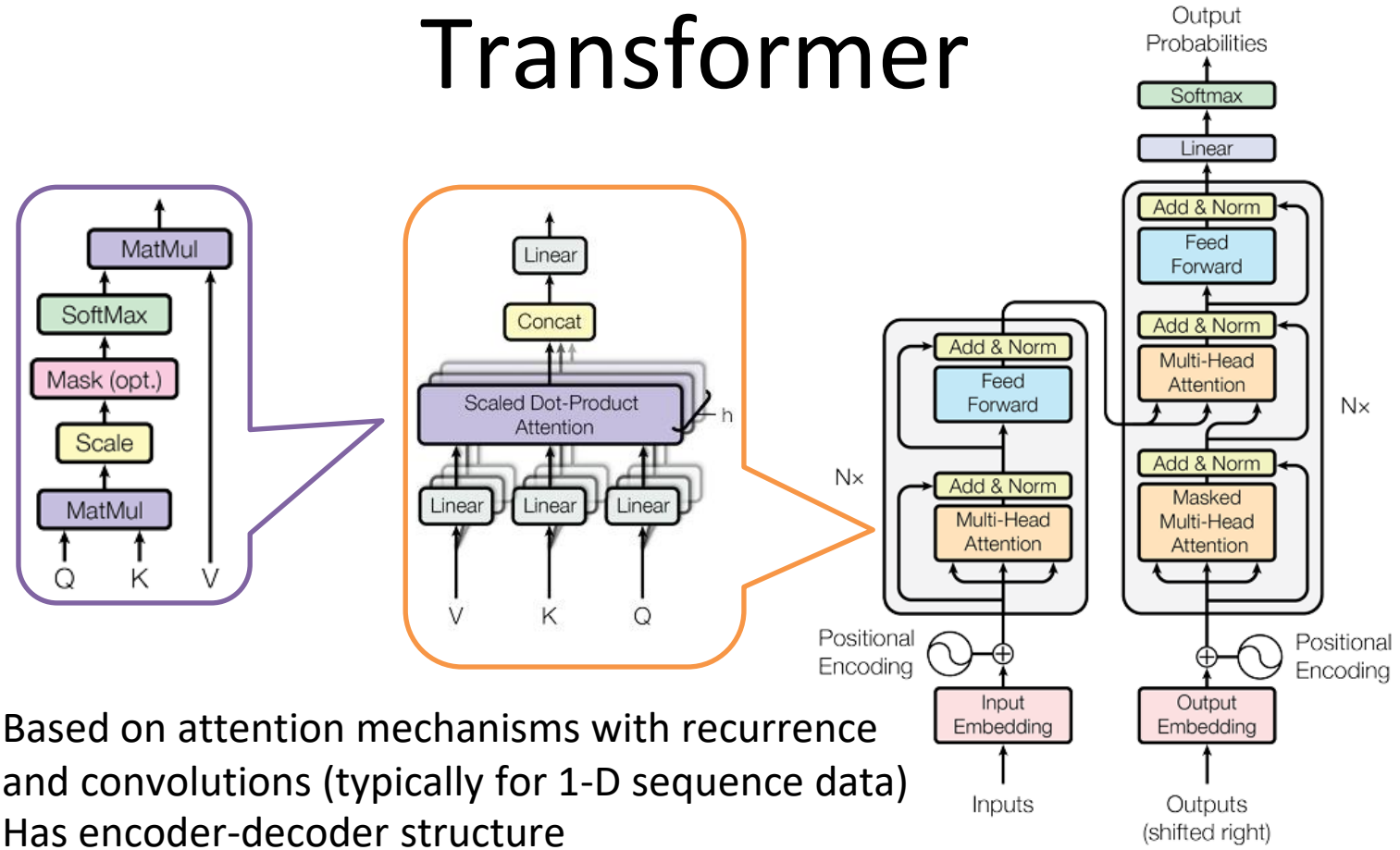

224
11
55
48
27
128
192
13
2048
dense
1000
Stride of 4
Max pooling
3


224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax



Transformer for Computer Vision

Transformer



- Based on attention mechanisms with recurrence and convolutions (typically for 1-D sequence data)
- Has encoder-decoder structure
  - Encoder maps input sequence to a sequence of continuous representations
  - Decoder generates an output sequence, one element at a time
- At each step, the model takes in the generated output as additional input when generating the next in sequence (auto-regressive)

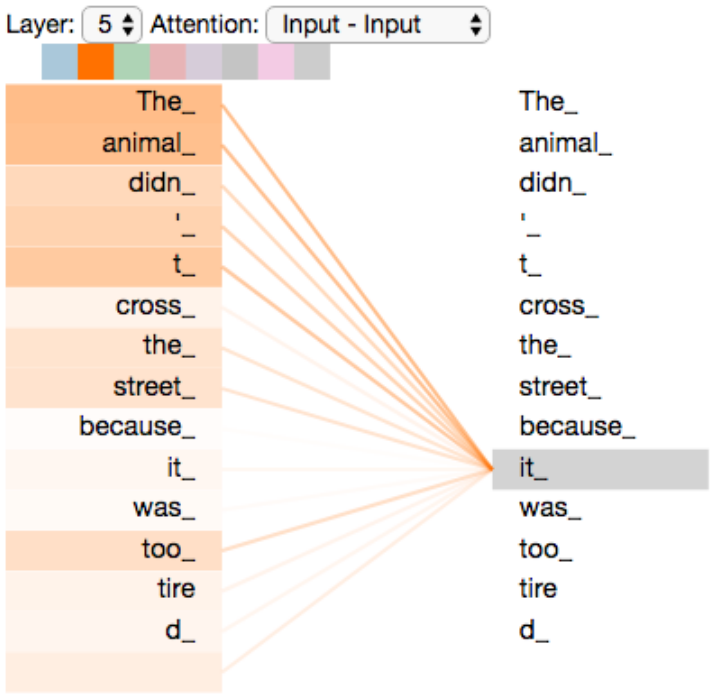
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, *30*.

Self-Attention (Simplified)

- Self-attention / intra-attention
  - Attention mechanism relating different positions of a single sequence to compute sequence representation
- Example:

"The animal didn't cross the street because it was too tired"

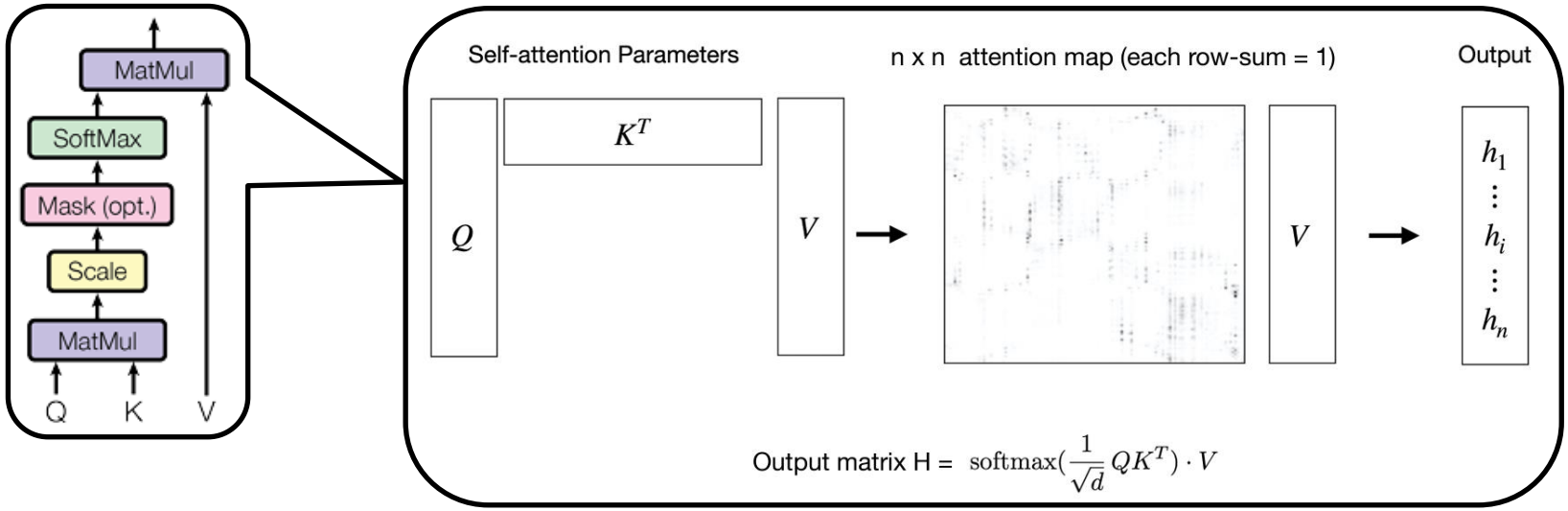
  - Sentence has 11 words/tokens
  - Words next to each other not necessarily relevant contextually ("it" with "because" and "was" vs. "it" with "animal")
  - Self-attention makes the model "look" at words in other positions to get better (more context) embedding by applying weighing or similarity



Source: https://jalammar.github.io/illustrated-transformer/

Transformer Self-Attention

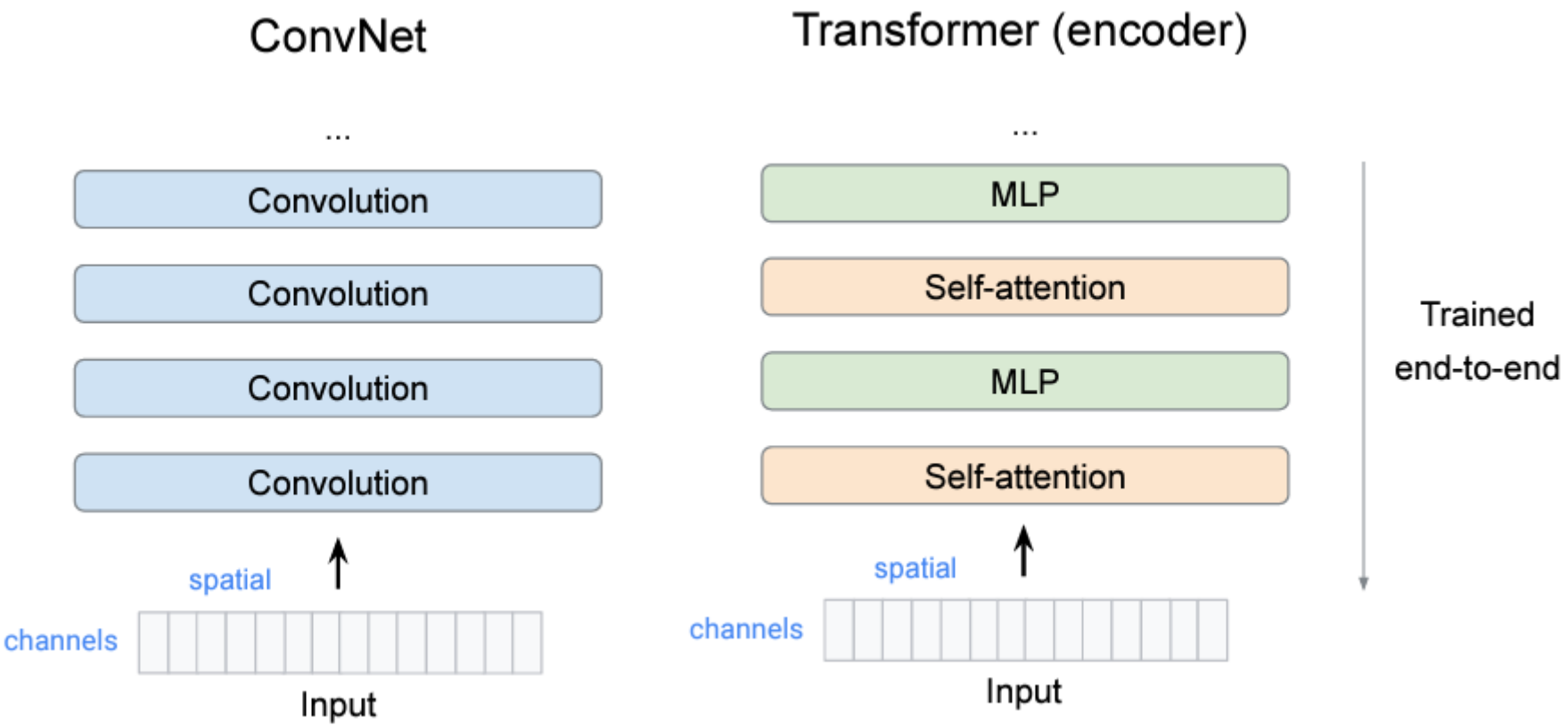
- Self-attention explicitly models interactions between all pairs of input embeddings



Input: **X** (matrix of n embedding vectors, each dim m)
Make 3 version of the input embedding:
Query $\boldsymbol{Q} = XW_Q$
Key $\boldsymbol{K} = XW_K$
Value $\boldsymbol{V} = XW_V$
W_Q, W_K, W_V are learned parameters

Adapted from NYU CSCI-GA.2271-001 notes.

CNN vs. Transformer



Convolutions (with kernels > 1x1) mix both the channels and the spatial locations

MLPs (=1x1 convs) only mix the channels per location
Self-attention mixes the spatial locations (and channels a bit)

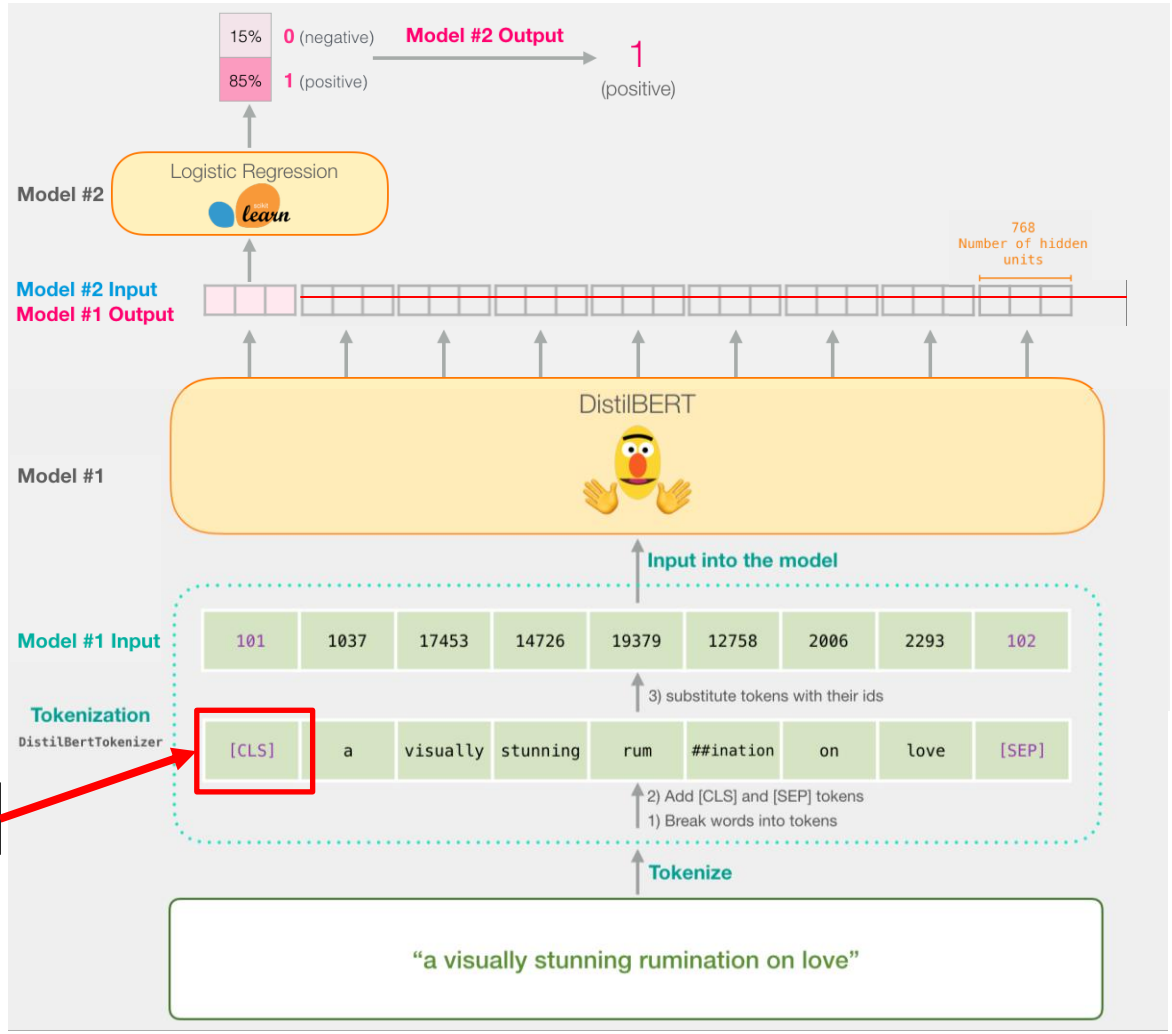
- ResNets have groups of 1x1 convolutions that are nearly identical to transformer's MLPs

Adapted from NYU CSCI-GA.2271-001 notes.

NLP Transformer

- Bidirectional Encoder Representations from Transformers (BERT)
  - Sentence sentiment classification example

"Classification Token"

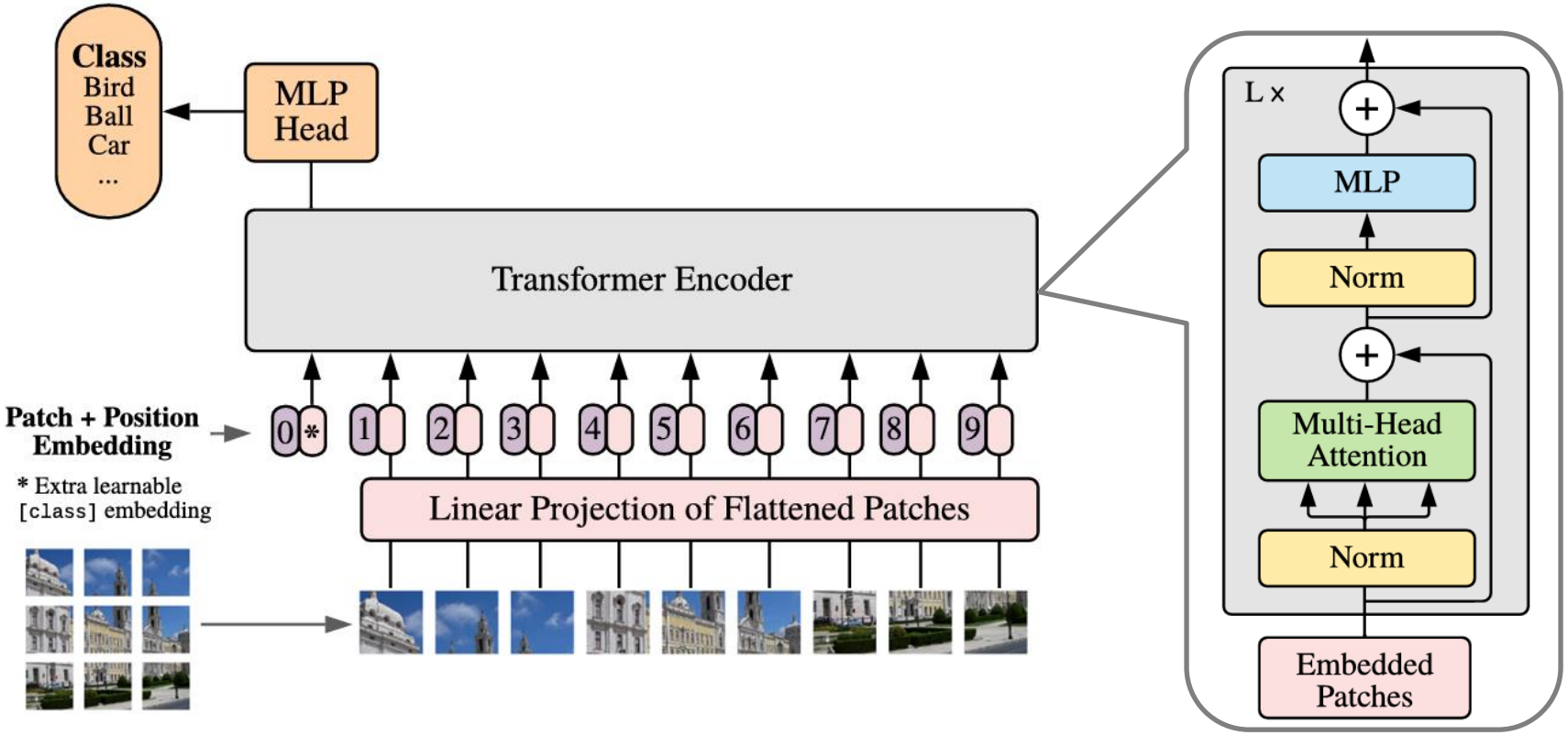


Kenton, J. D. M. W. C., & Toutanova, L. K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of NAACL-HLT* (pp. 4171-4186).

Source: https://jalammar.github.io/a-visual-guide-to-using-bert-for-the-first-time/

Vision Transformer (ViT)

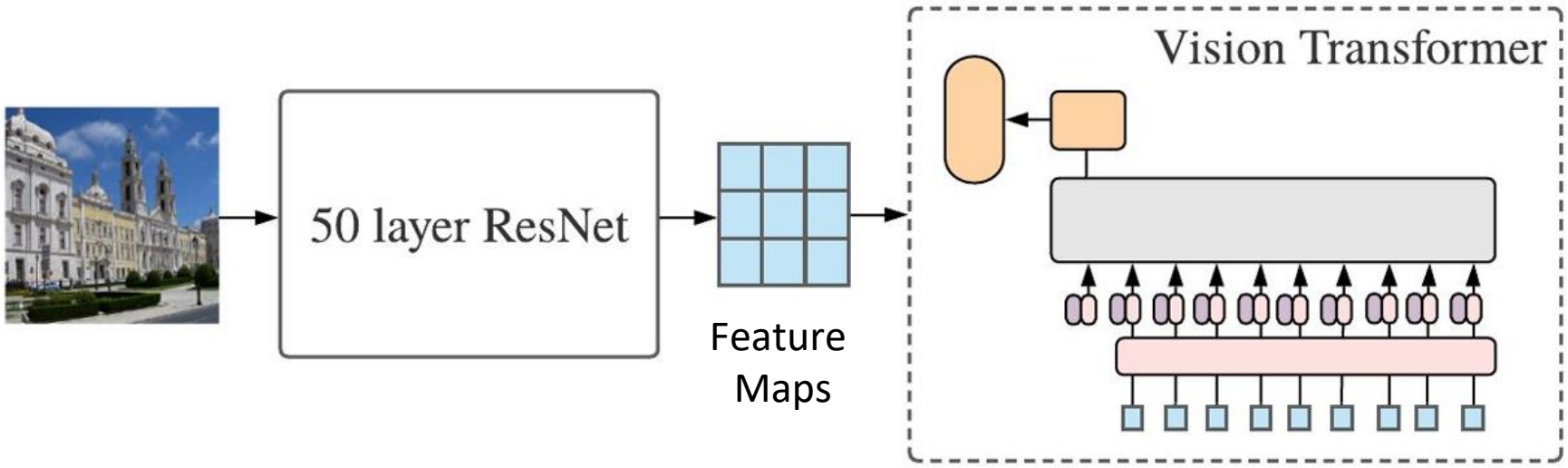
- Applying transformer directly to image patches



Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020, September). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

ResNet-ViT Hybrid

- Feed intermediate feature maps from ResNet into ViT
  - Only better for smaller models



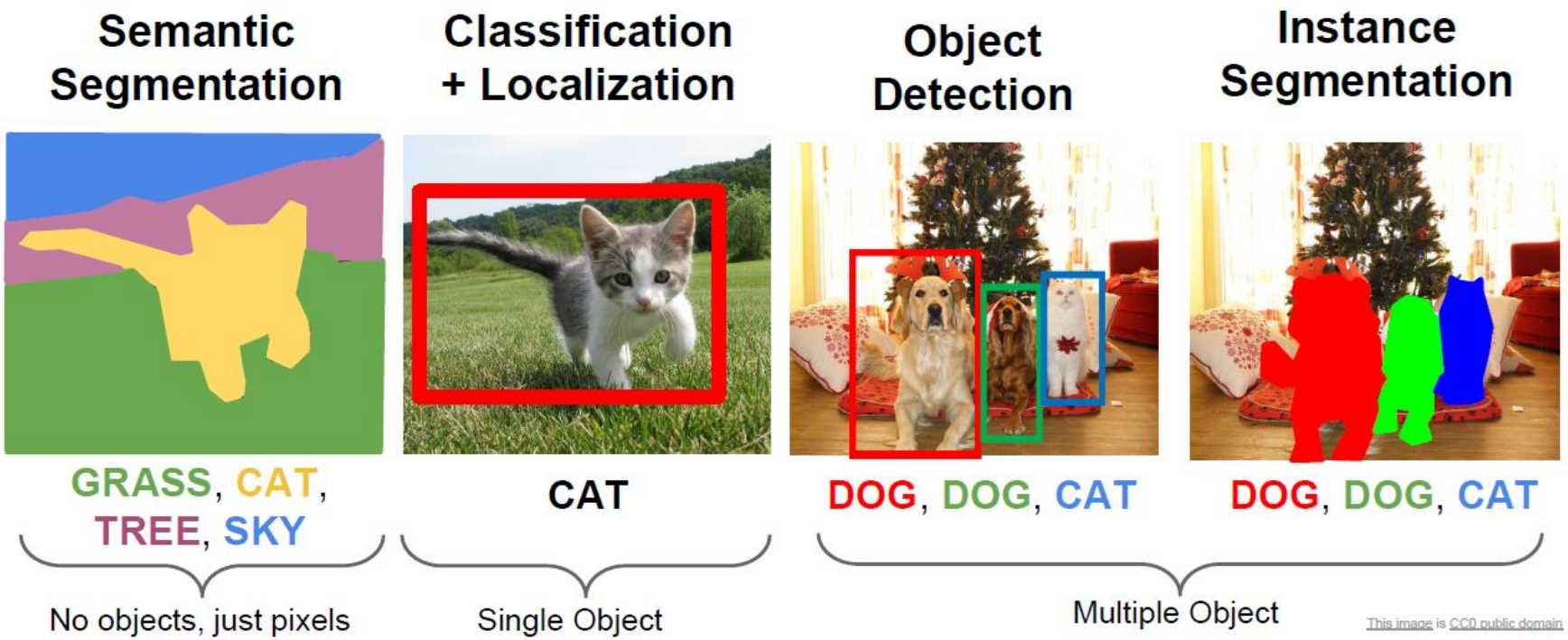
Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020, September). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In *International Conference on Learning Representations*.

Overview and Insights

- Transformer model:
  - Alternating layers of self-attention and MLP
  - Very few assumptions built into model
  - Trained end-to-end
  - Easy to scale to be very wide and deep
  - Originally applied to NLP (sequence of words)
  - Many architectures and applications
- Transformer in vision:
  - Representing image pixels may be to large
  - Use patches position in 2D array
  - CNN/ResNets still superior for small models/data
  - Performs better at very large scale (100M-1B images)

Segmentation, Localization, and Detection

Segmentation, Localization, and Detection



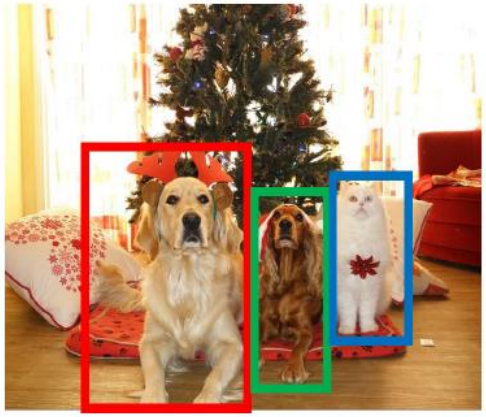
Segmentation, Localization, and Detection

Semantic Segmentation

GRASS, CAT, TREE, SKY

No objects, just pixels

2D Object Detection



DOG, DOG, CAT

Object categories + 2D bounding boxes

3D Object Detection

Car

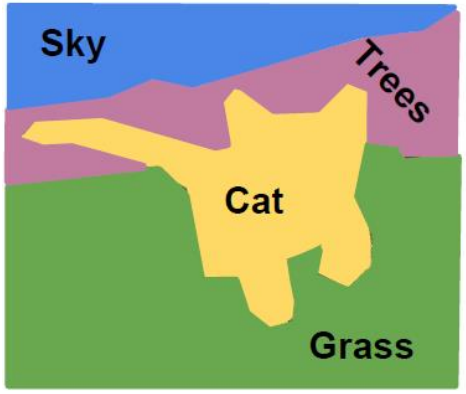
Object categories + 3D bounding boxes

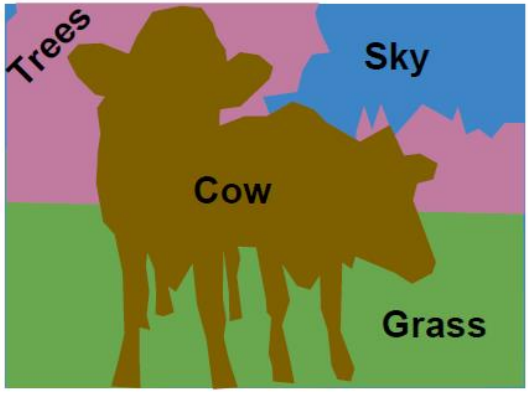
Segmentation

Semantic Segmentation

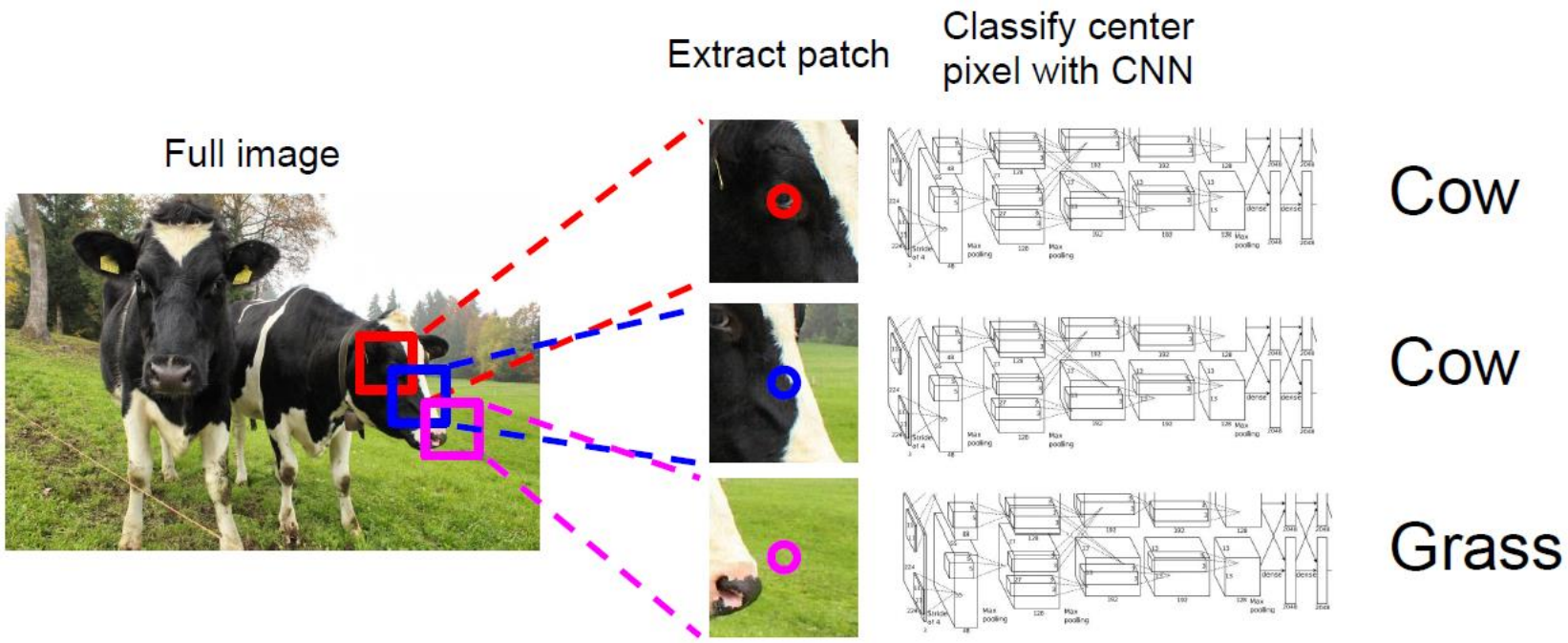
- Label each pixel in the image with a category label

- Don’t differentiate instances, only label the pixels





Semantic Segmentation: Sliding Window



Problem:
Very inefficient. Not re-using shared features between overlapping patches

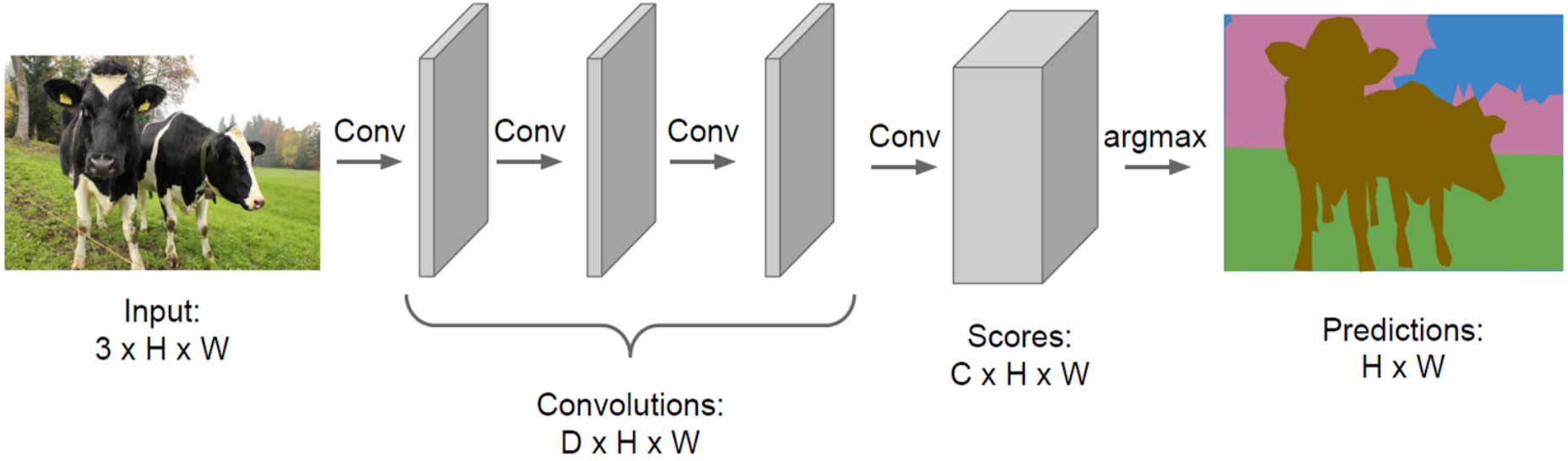
Farabet et al, "Learning Hierarchical Features for Scene Labeling," TPAMI 2013
Pinheiro and Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling", ICML 2014

Semantic Segmentation: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: ??



Problem:
Convolutions at original resolution will be very computationally expensive

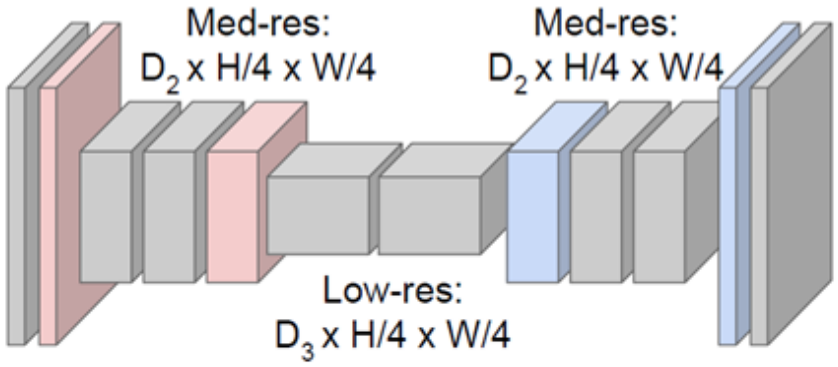
Semantic Segmentation: Fully Convolutional

Downsampling: Pooling, strided convolution

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

Upsampling: Unpooling or strided transpose convolution

Input:
3 x H x W



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

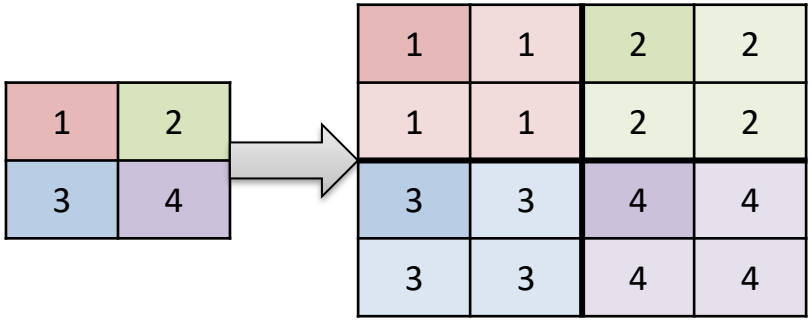
High-res:
D_1 x H/2 x W/2

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

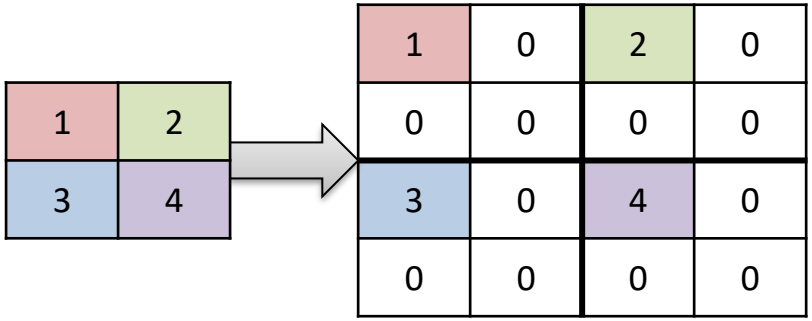
In-Network Upsampling: Unpooling

Nearest Neighbor



Input: 2 x 2 Output: 4 x 4

“Bed of Nails”



Input: 2 x 2 Output: 4 x 4

In-Network Upsampling: 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

5	6
7	8

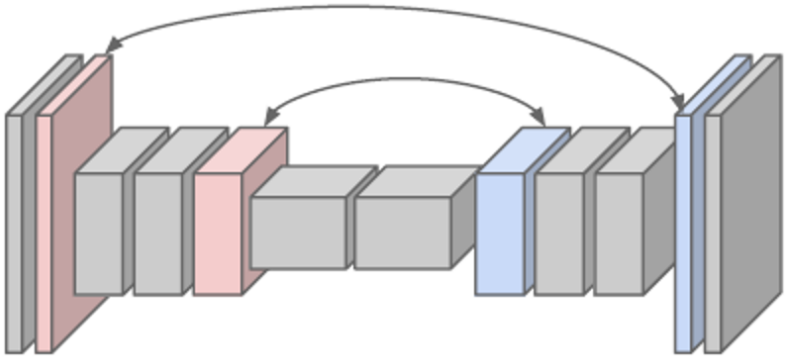
Input: 2 x 2

Max Unpooling
Use positions from pooling layer

0	0	6	0
0	5	0	0
0	0	0	0
7	0	0	8

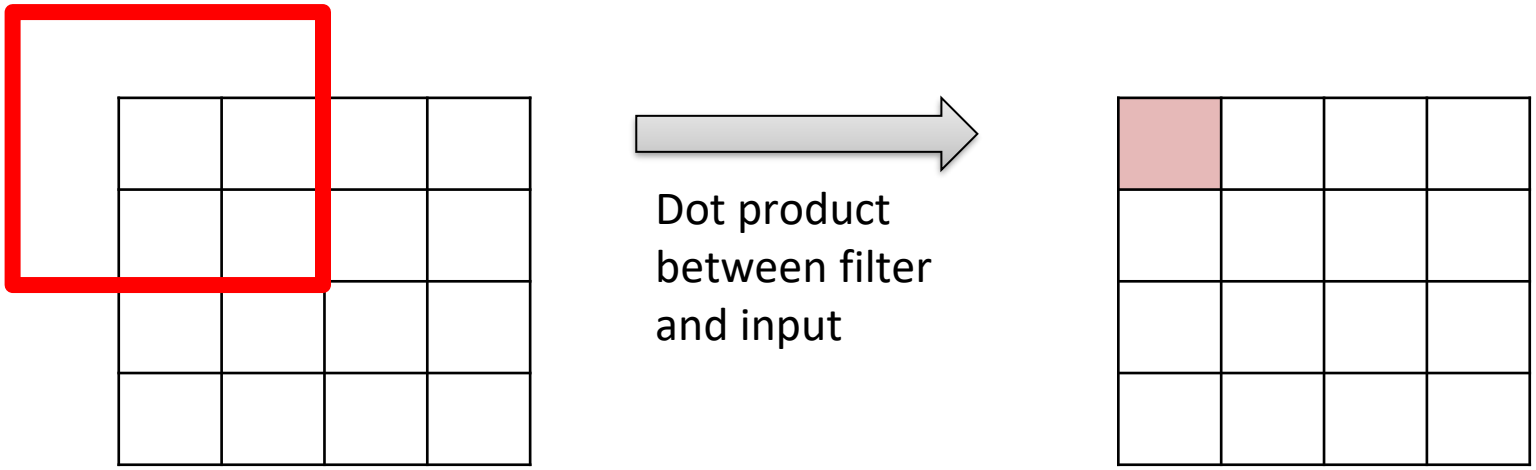
Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



Learnable Upsampling: Transpose Convolution

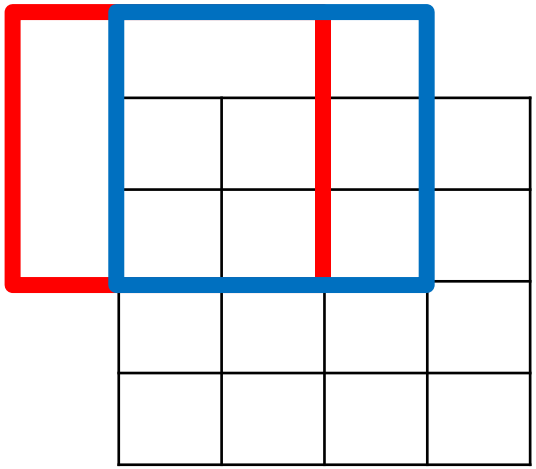
Recap: Typical 3 x 3 convolution, stride 1 pad 1



Slide Credit: CS231n

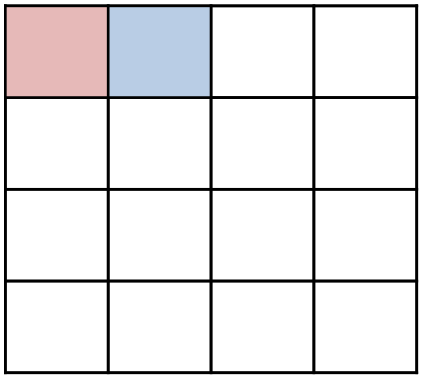
Learnable Upsampling: Transpose Convolution

Recap: Typical 3 x 3 convolution, stride 1 pad 1



Input: 4 x 4

Dot product between filter and input

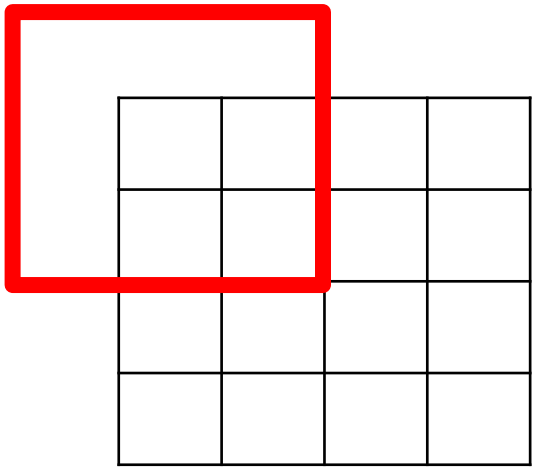


Output: 4 x 4

Slide Credit: CS231n

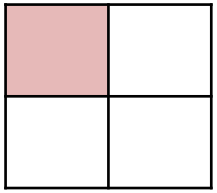
Learnable Upsampling: Transpose Convolution

Recap: Typical 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input

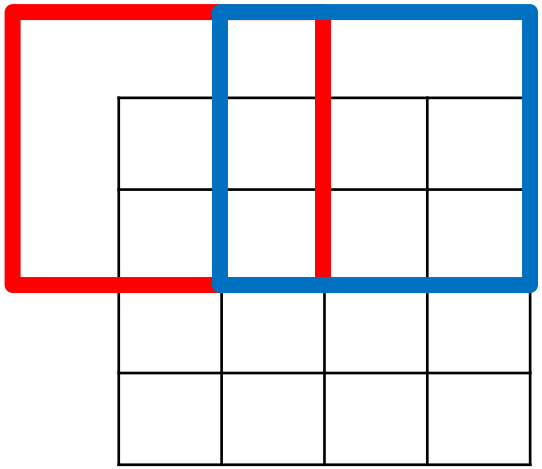


Output: 2 x 2

Slide Credit: CS231n

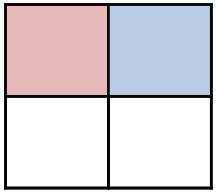
Learnable Upsampling: Transpose Convolution

Recap: Typical 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input

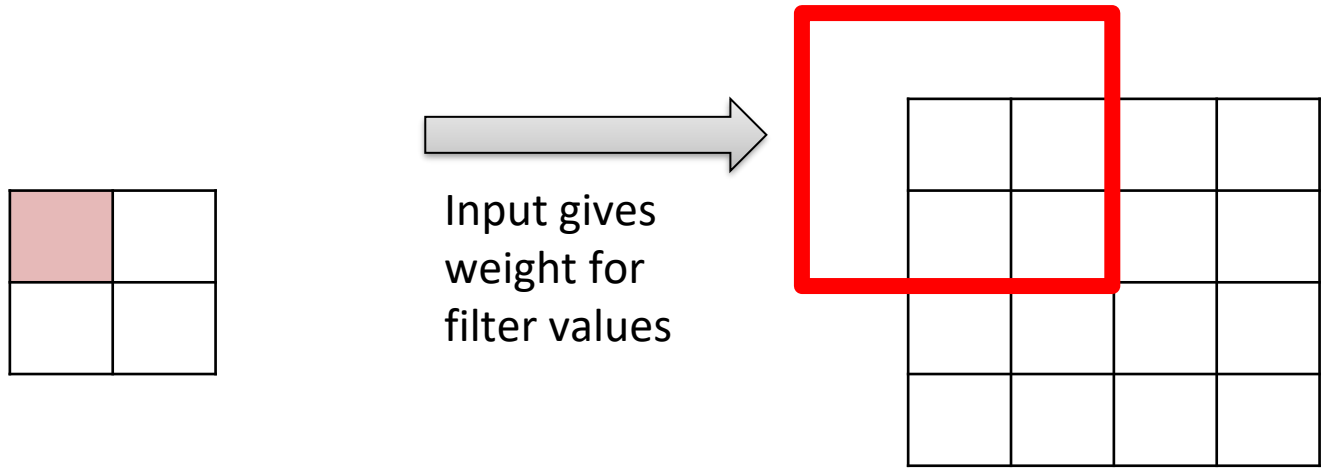


Output: 2 x 2

Slide Credit: CS231n

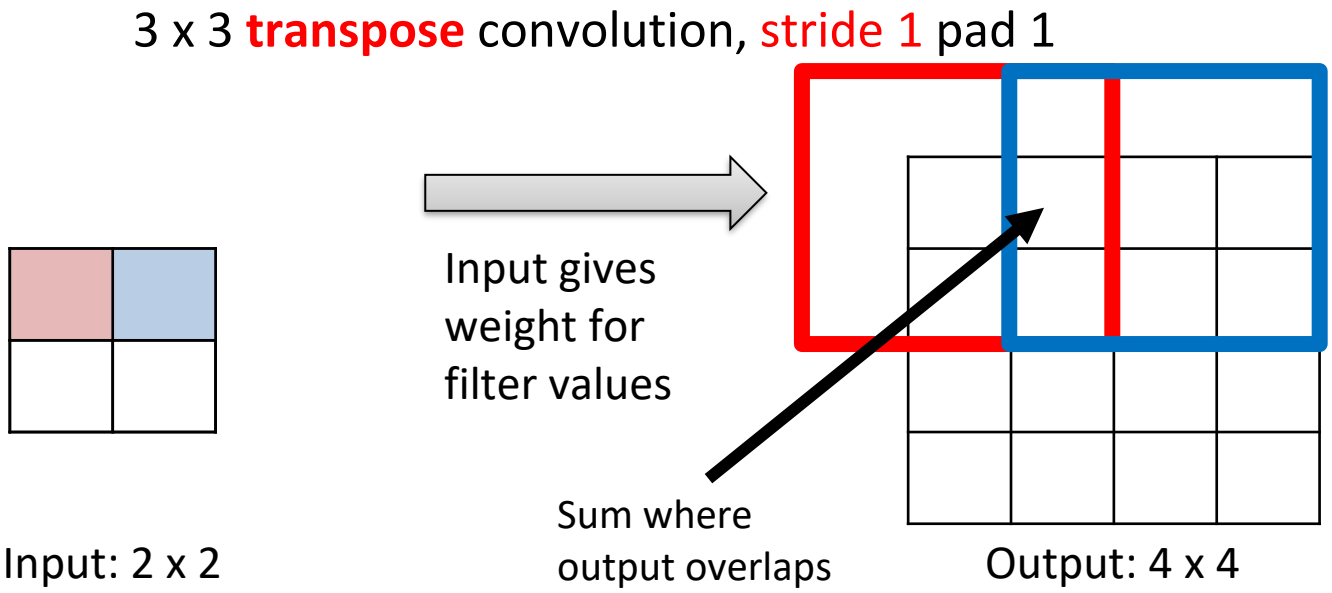
Learnable Upsampling: Transpose Convolution

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



Slide Credit: CS231n

Learnable Upsampling: Transpose Convolution



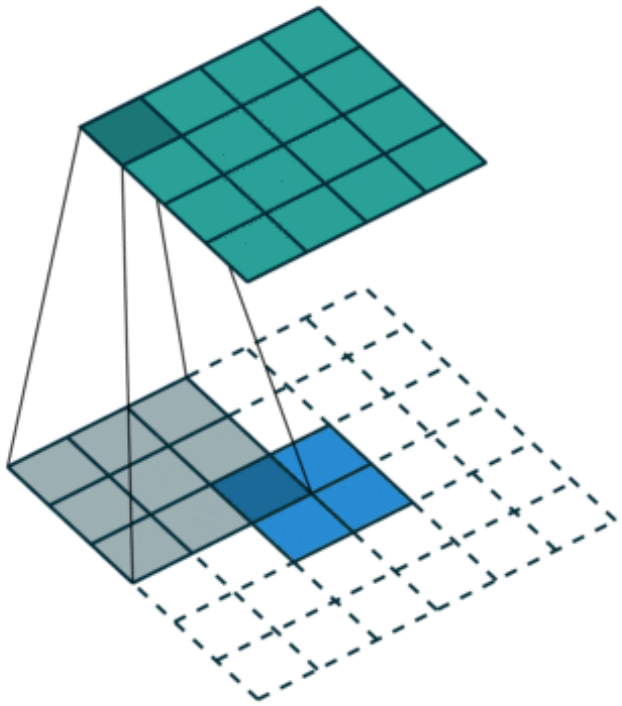
- Filter moves 2 pixels in the **output** for every one pixel in the **input**
- Stride gives ratio between movement in output and input
- Other names: Deconvolution, Upconvolution, Fractionally strided convolution, Backward strided convolution

Slide Credit: CS231n

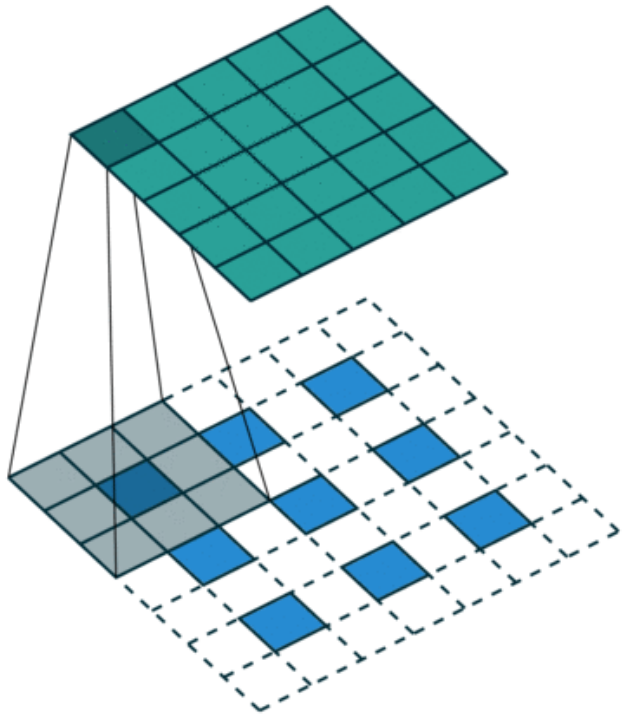
Learnable Upsampling: Transpose Convolution

Transposed convolutional layer

Stride 1



Stride 2



Slide Credit: CS231n

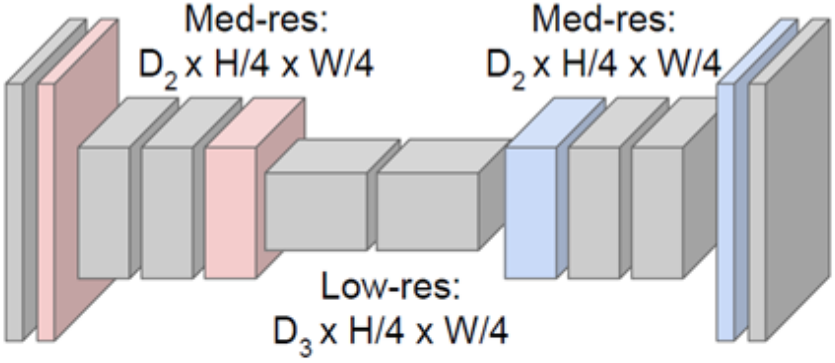
Learnable Upsampling: Transpose Convolution

Downsampling:
Pooling, strided convolution

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

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W

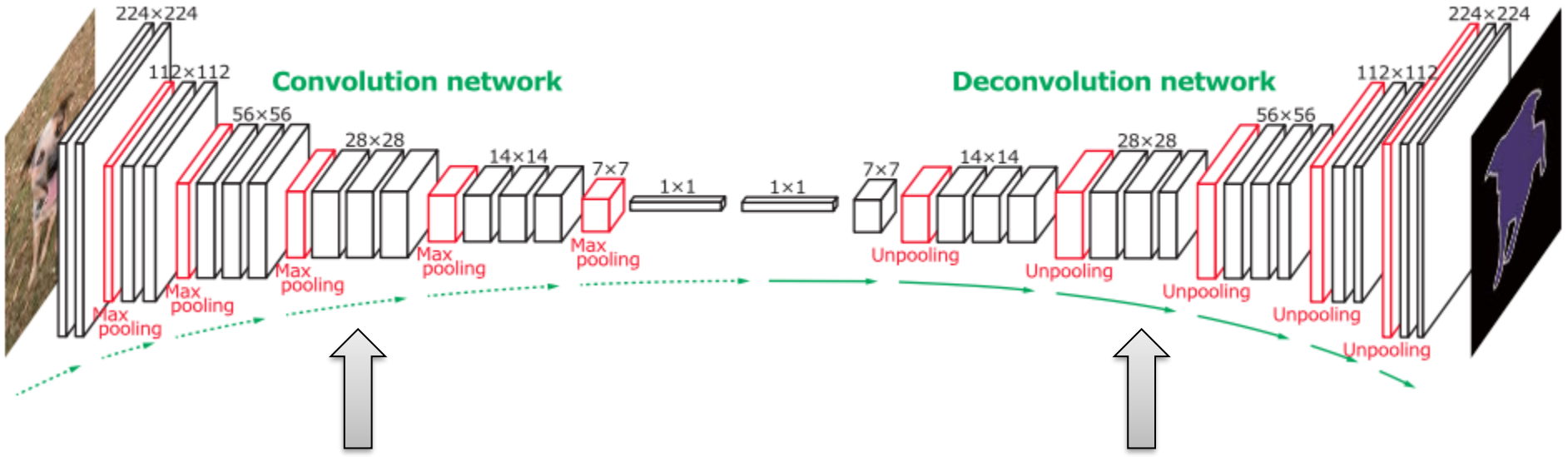


Predictions:
H x W

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

Slide Credit: CS231n

Deconvolutional Neural Network (DeConv)



Normal VGG

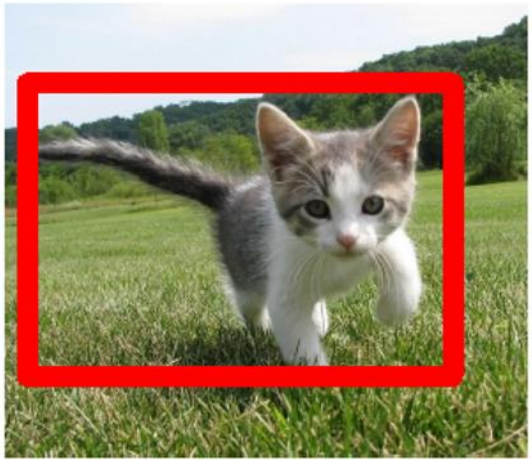
"Upside-down" VGG

Slide Credit: CS231n

Localization and Detection

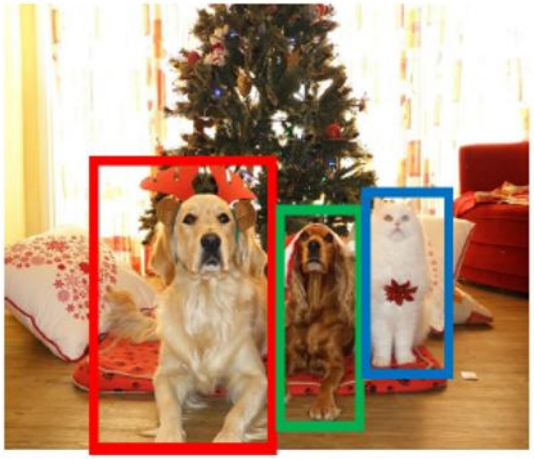
2D Object Detection

Classification + Localization



CAT

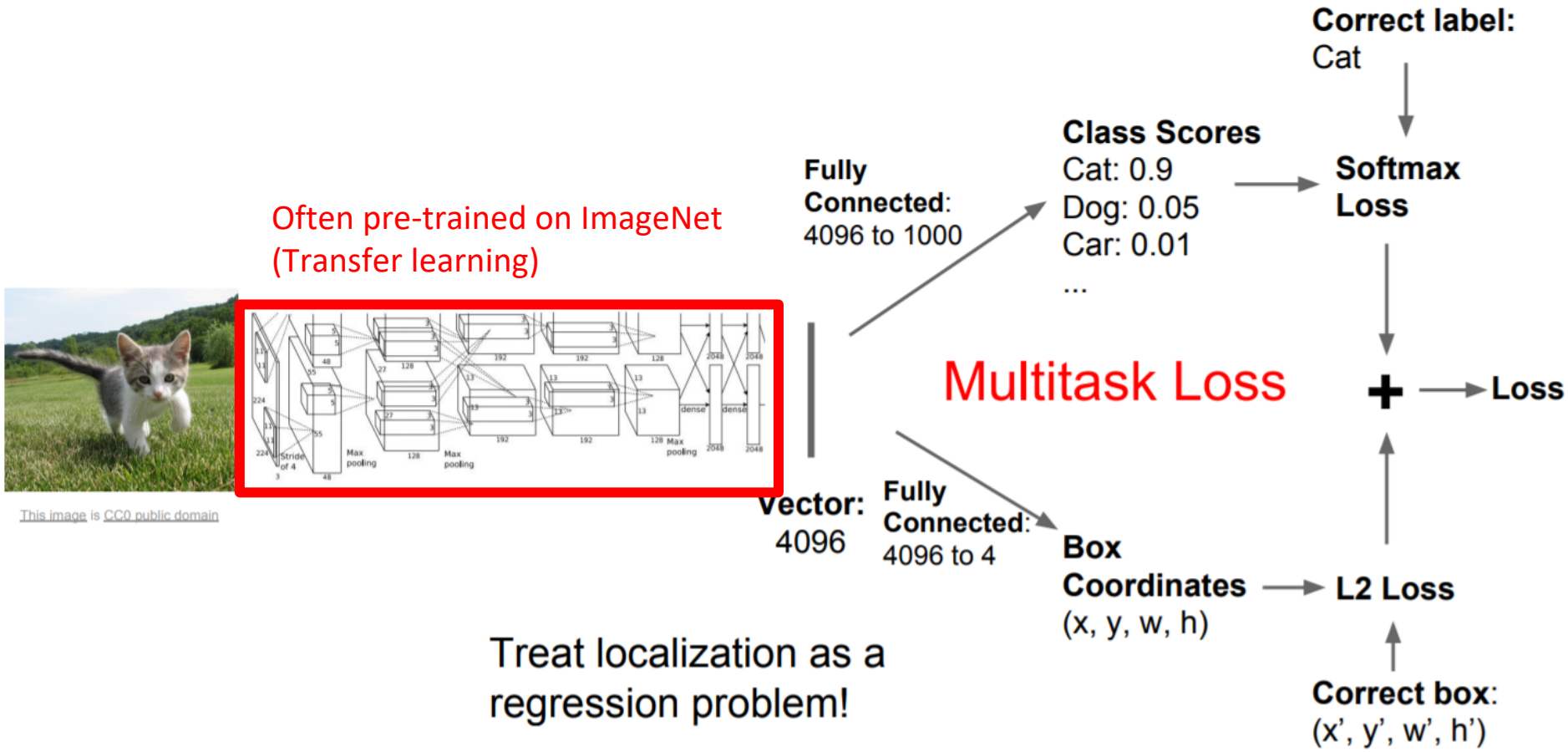
Object Detection



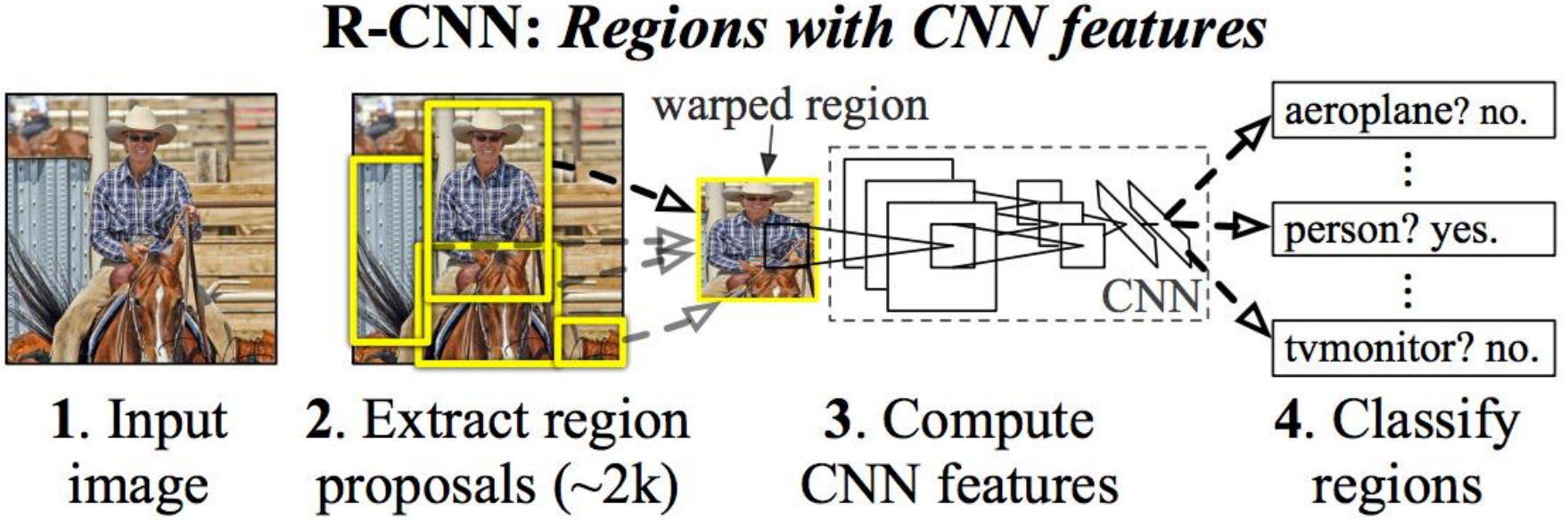
DOG, DOG, CAT

Object categories + 2D bounding boxes

Classification + Localization

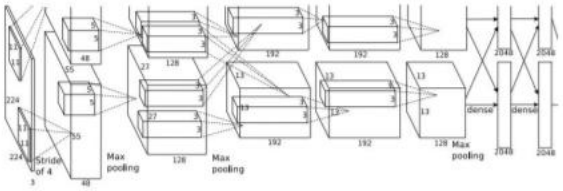


Classification + Localization



Object Detection as Regression

Each image needs a different number of outputs



CAT: (x, y, w, h)

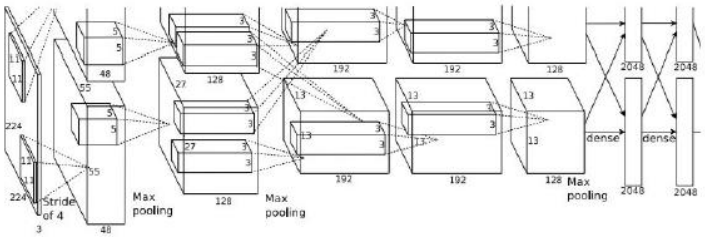
4 values

16 values

Many more values

Object Detection as Classification: Sliding Window

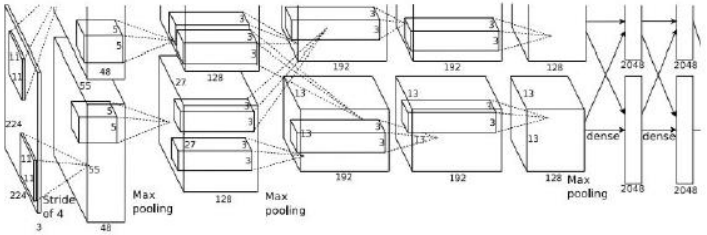
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? NO
Cat? NO
Background? YES

Object Detection as Classification: Sliding Window

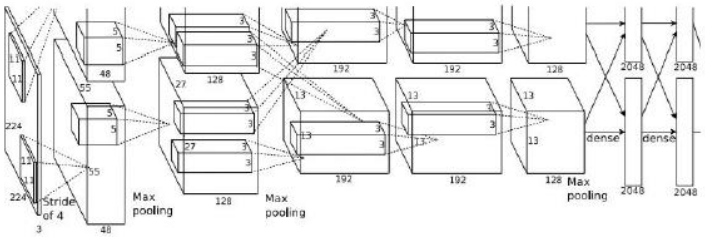
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

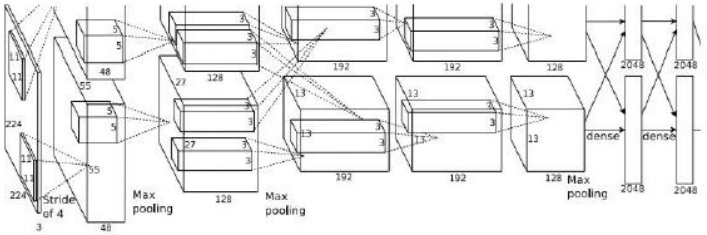
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



Dog? YES
Cat? NO
Background? NO

Object Detection as Classification: Sliding Window

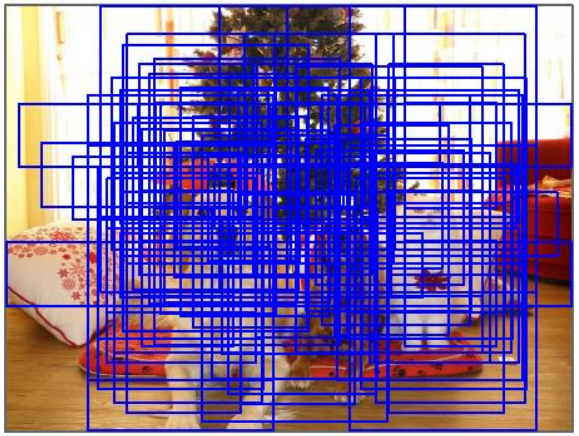
Apply a CNN to many different crops of the image, CNN classifies each crop as object or background

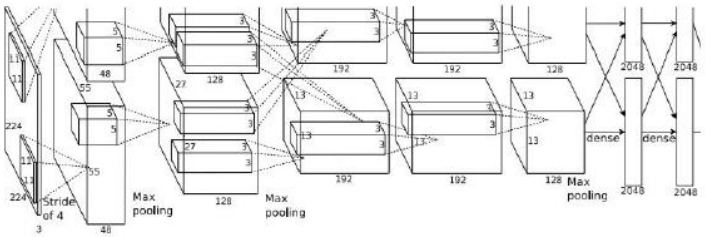


Dog? NO
Cat? YES
Background? NO

Object Detection as Classification: Sliding Window

Apply a CNN to many different crops of the image, CNN classifies each crop as object or background



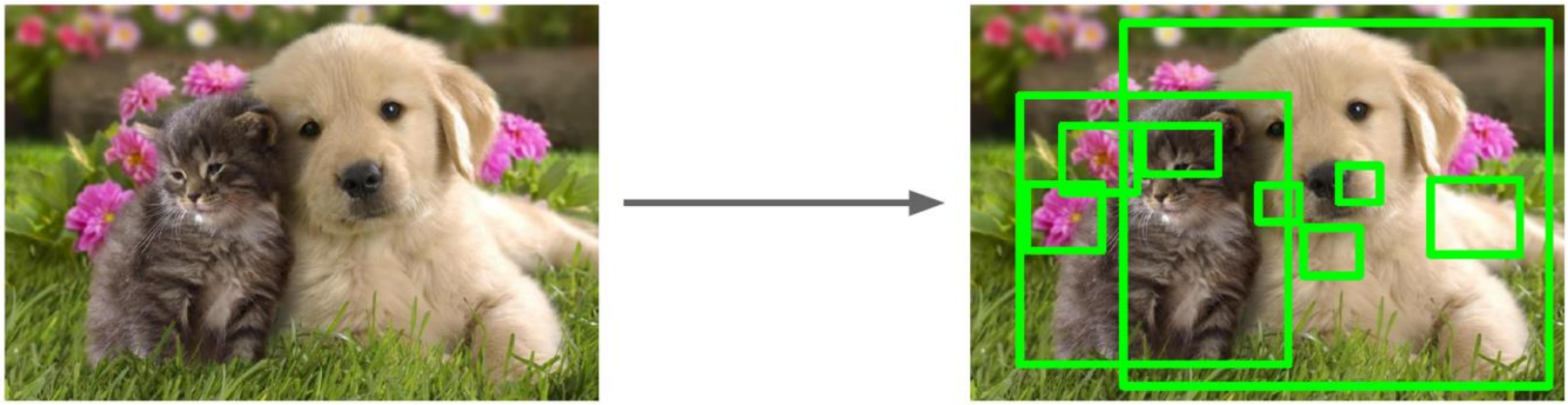


Dog? NO
Cat? YES
Background? NO

Problem: Need to apply CNN to huge number of locations, scales, and aspect ratios, very computationally expensive.

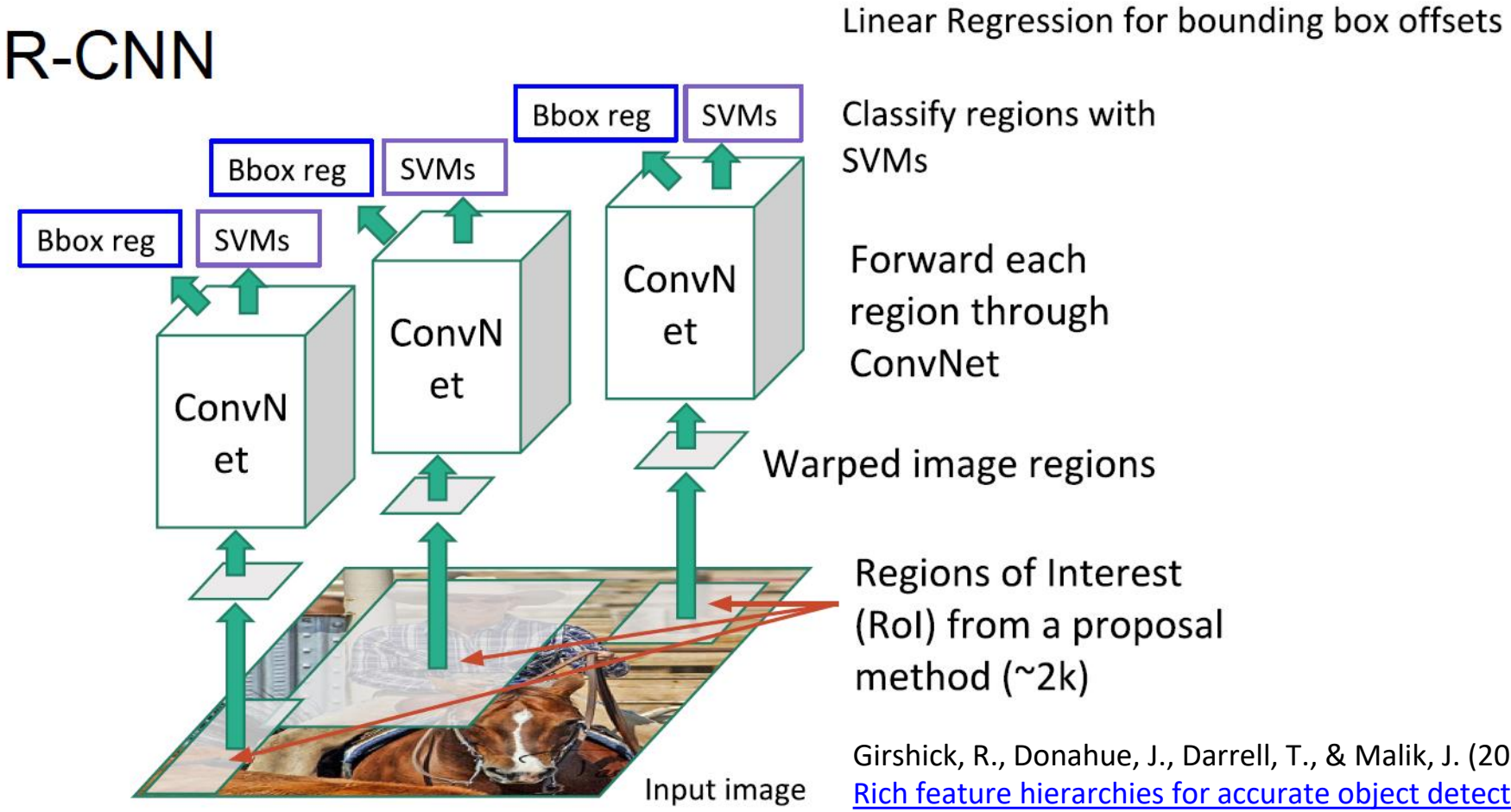
Region Proposal/Selective Search

- Find “blobby” image regions that are likely to contain objects
- Relatively fast to run
  - E.g. Selective Search gives 2000 region proposals in a few seconds on CPU



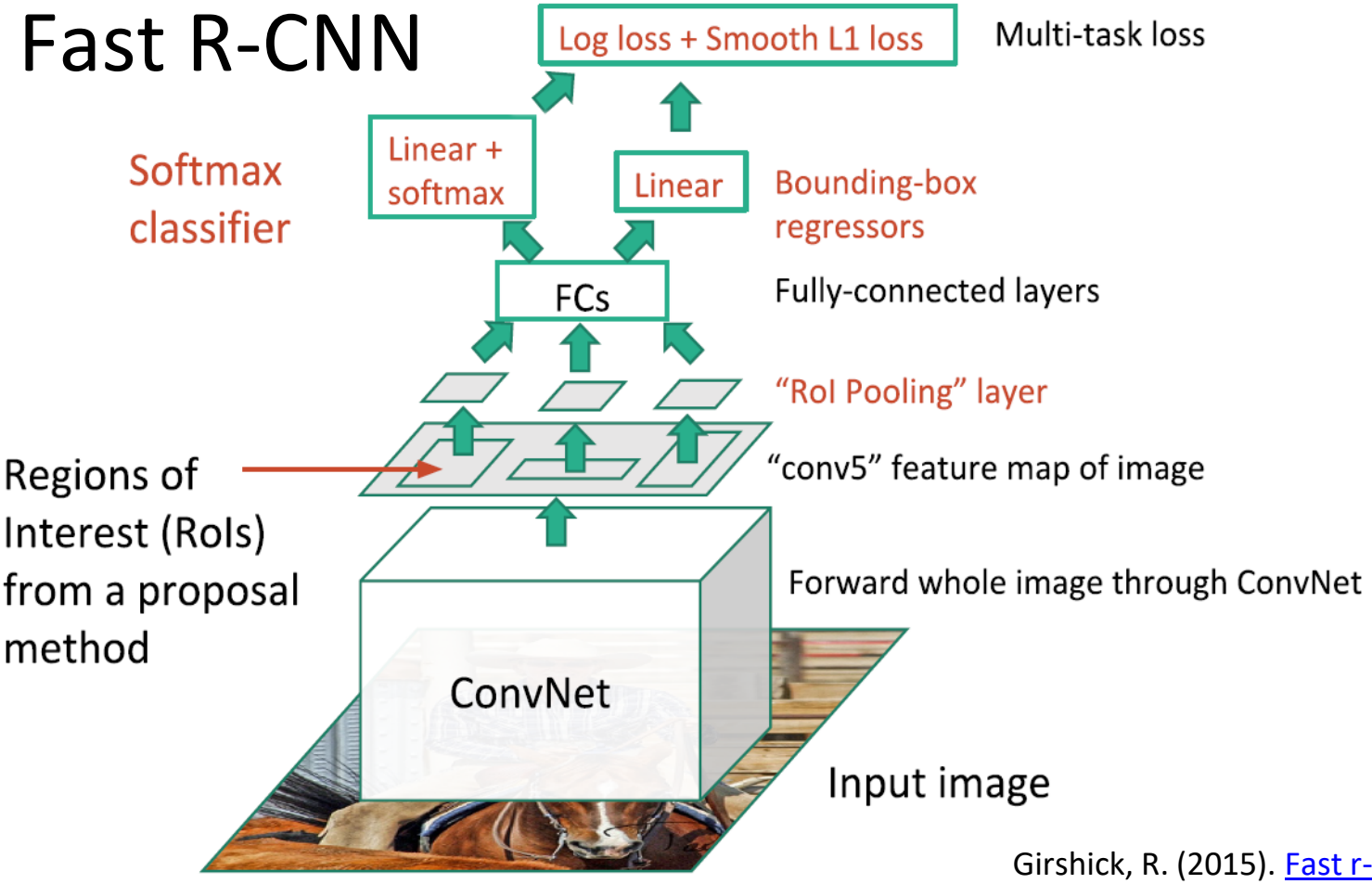
Alexe et al, “Measuring the objectness of image windows”, TPAMI 2012
Uijlings et al, “Selective Search for Object Recognition”, IJCV 2013
Cheng et al, “BING: Binarized normed gradients for objectness estimation at 300fps”, CVPR 2014
Zitnick and Dollar, “Edge boxes: Locating object proposals from edges”, ECCV 2014

Region Proposal/Selective Search



Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

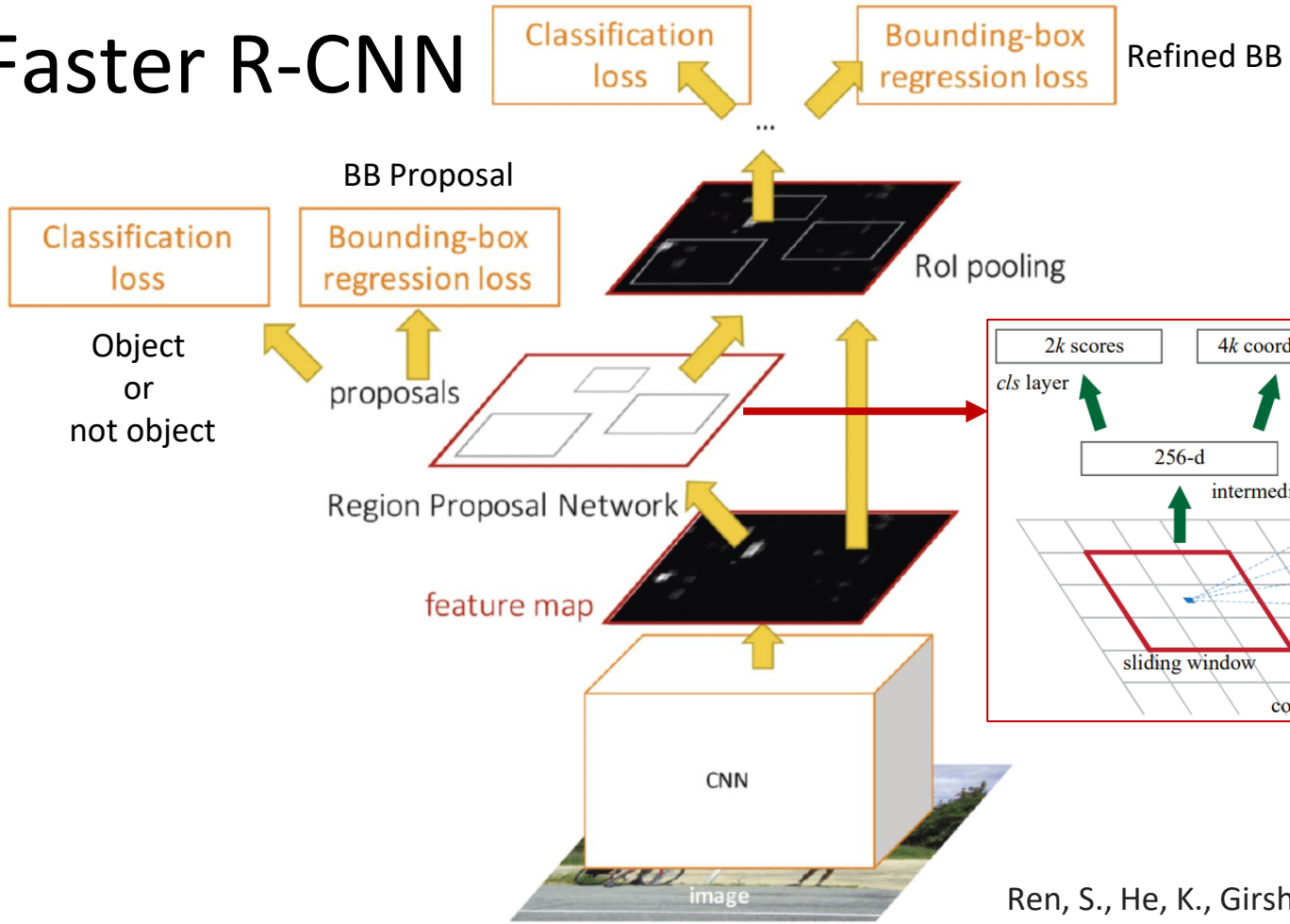
Region Proposal/Selective Search

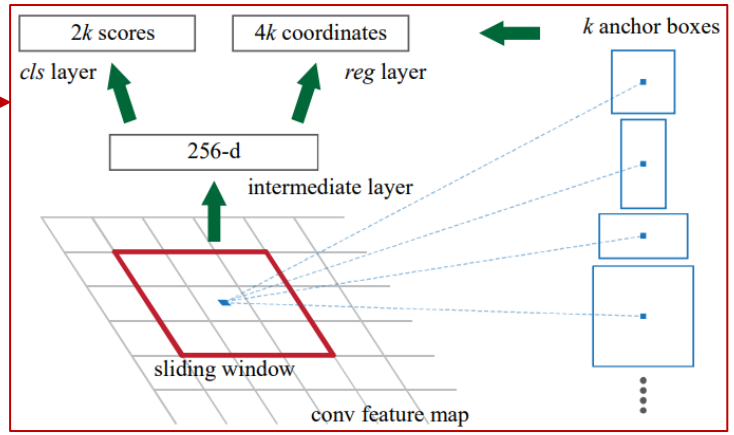


Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 1440-1448).

Region Proposal Network + Fast R-CNN

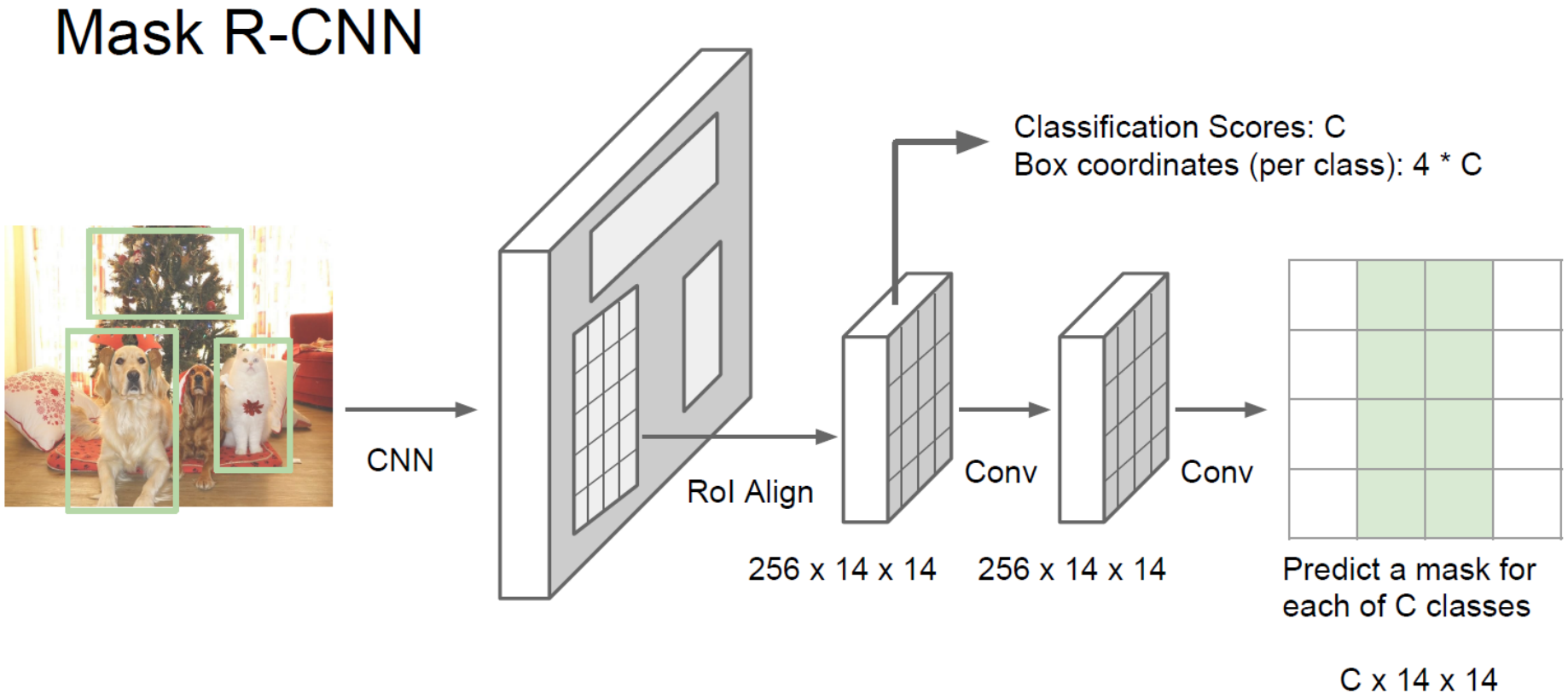
Faster R-CNN





Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, *28*, 91-99.

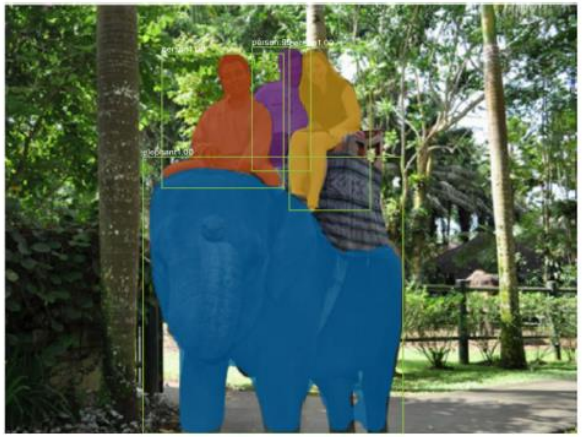
Instance Segmentation

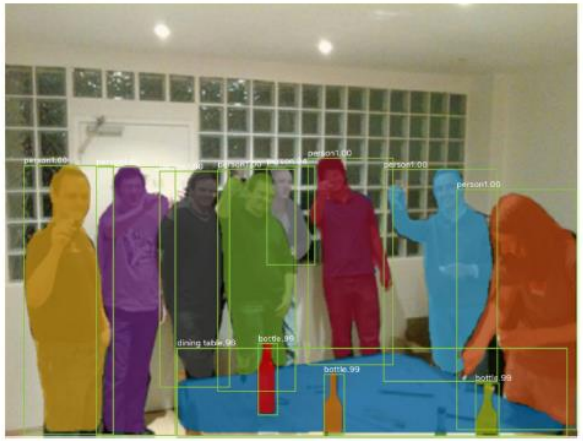


He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

Instance Segmentation

Mask R-CNN: Very Good Results!





He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

Instance Segmentation

Mask R-CNN
Also does pose

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961-2969).

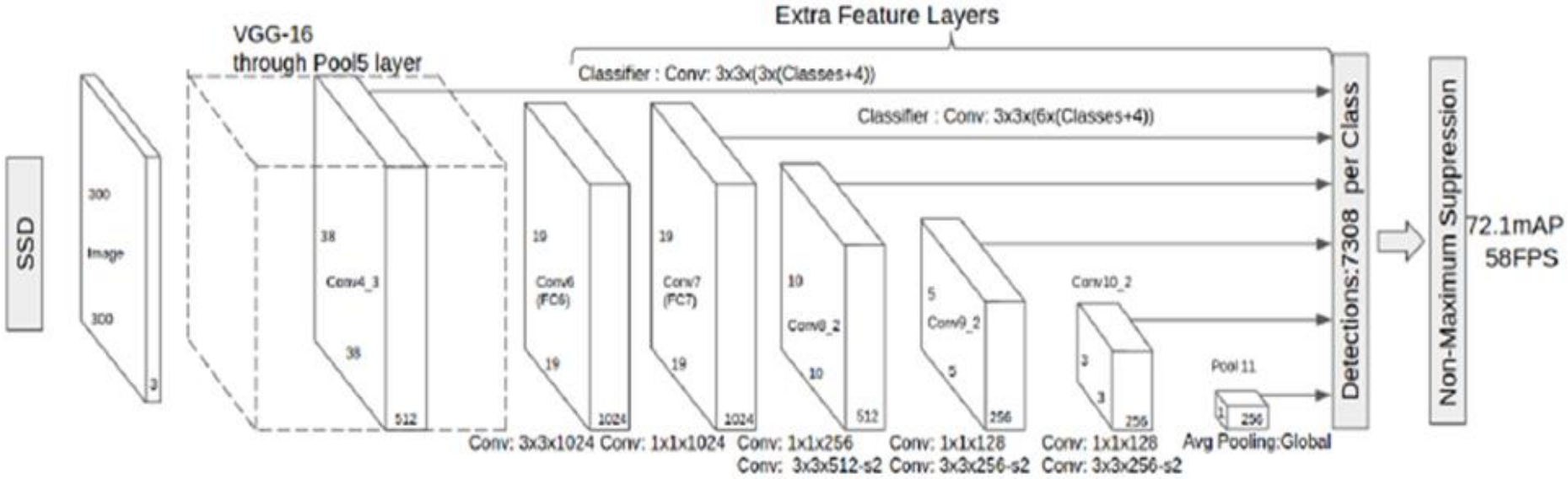
Detection without Proposals: YOLO / SSD

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779-788).
Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016, October). Ssd: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

Single Shot Multibox Detector (SSD)

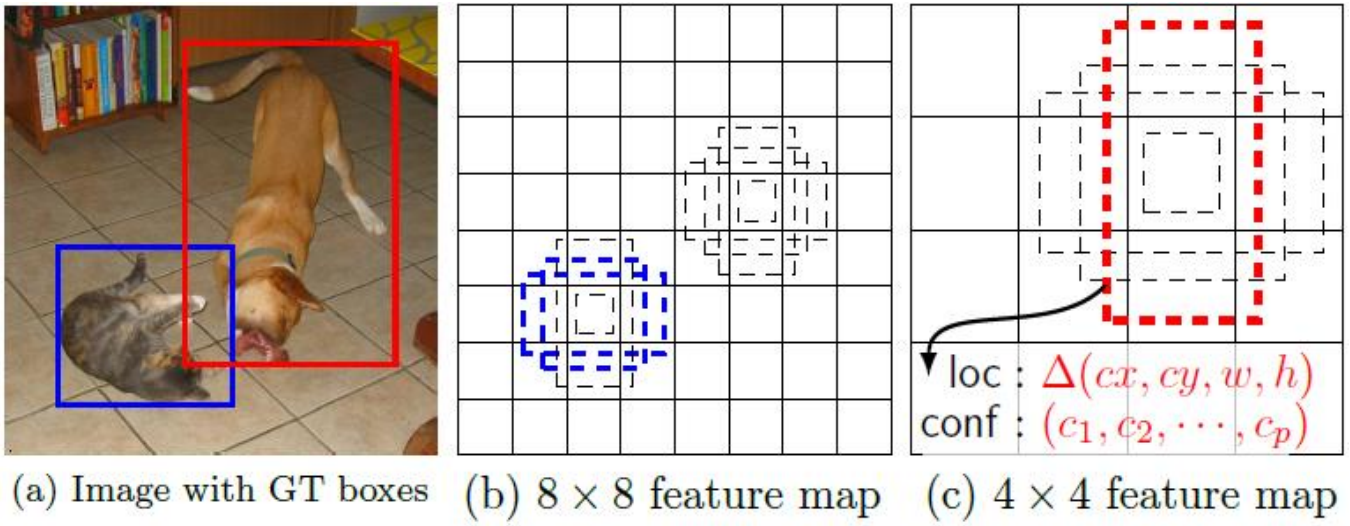
- A single deep neural network
- Related methods are structure-complicated and hard to bring high speed and good accuracy
- Advantages
  - Provides a unified framework
  - Much faster
  - Better accuracy (even with smaller input image size)

Single Shot Multibox Detector (SSD)



- Structure advantage for speed
  - Eliminate bounding box proposals and subsequently pixel or feature resampling
  - Adding convolution feature layers to the end of the base network to predict detections at multiple scales

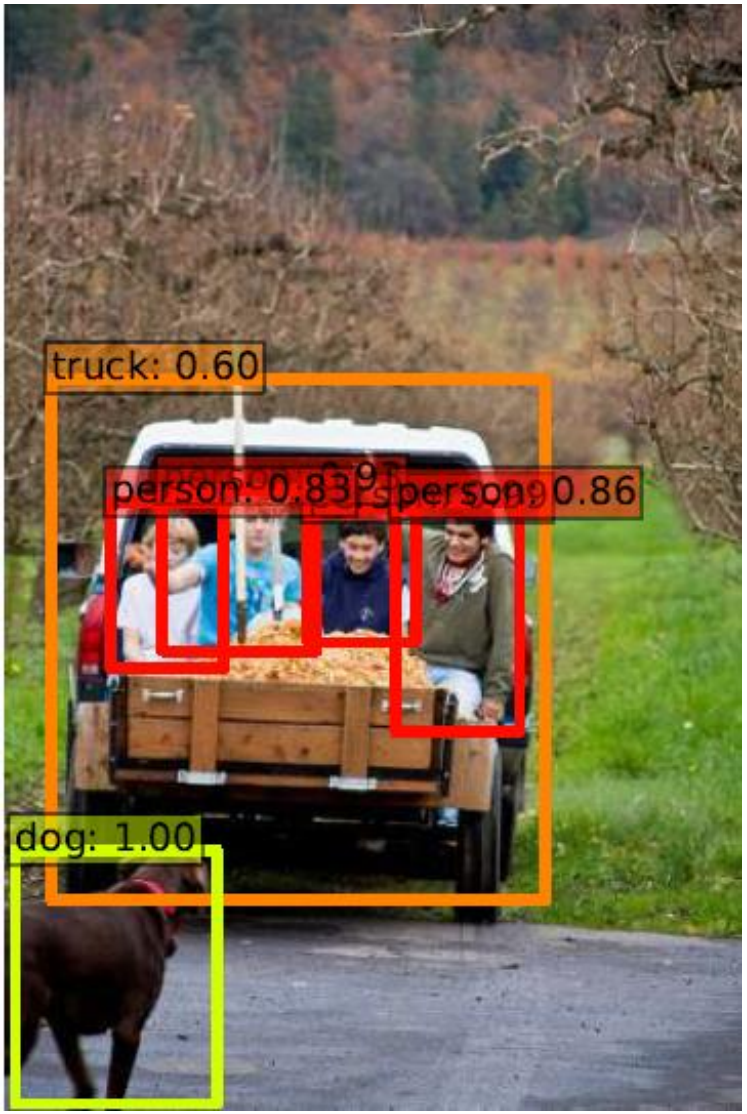
Mechanics of SSD

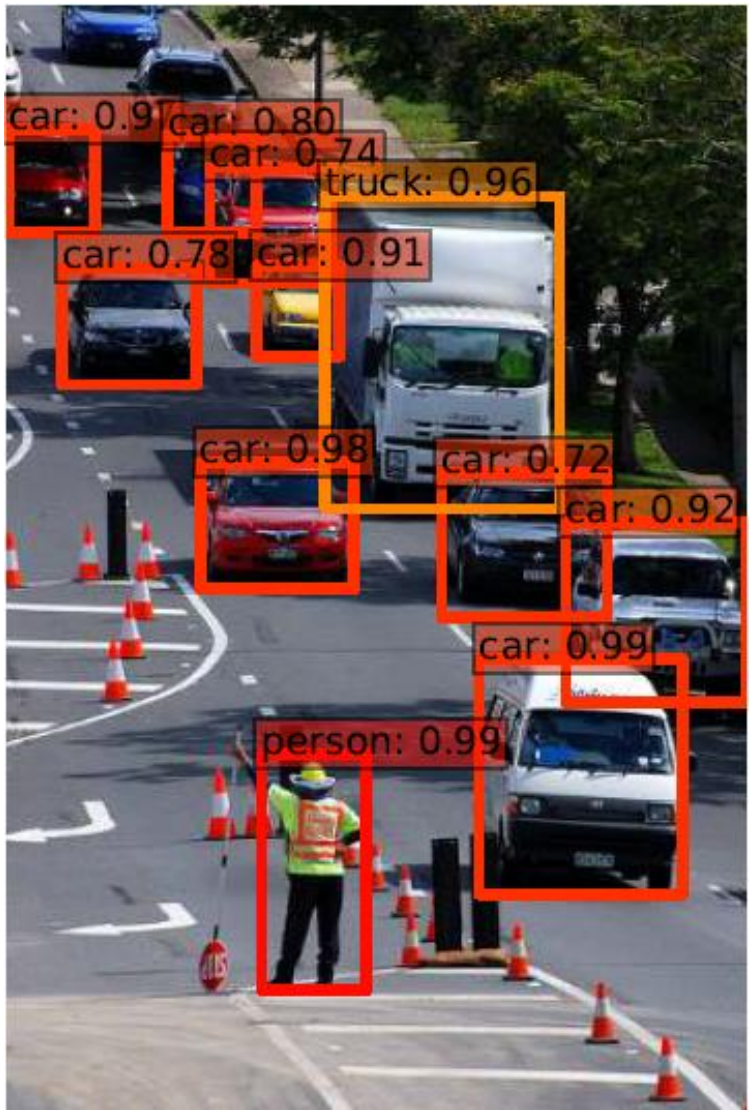


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

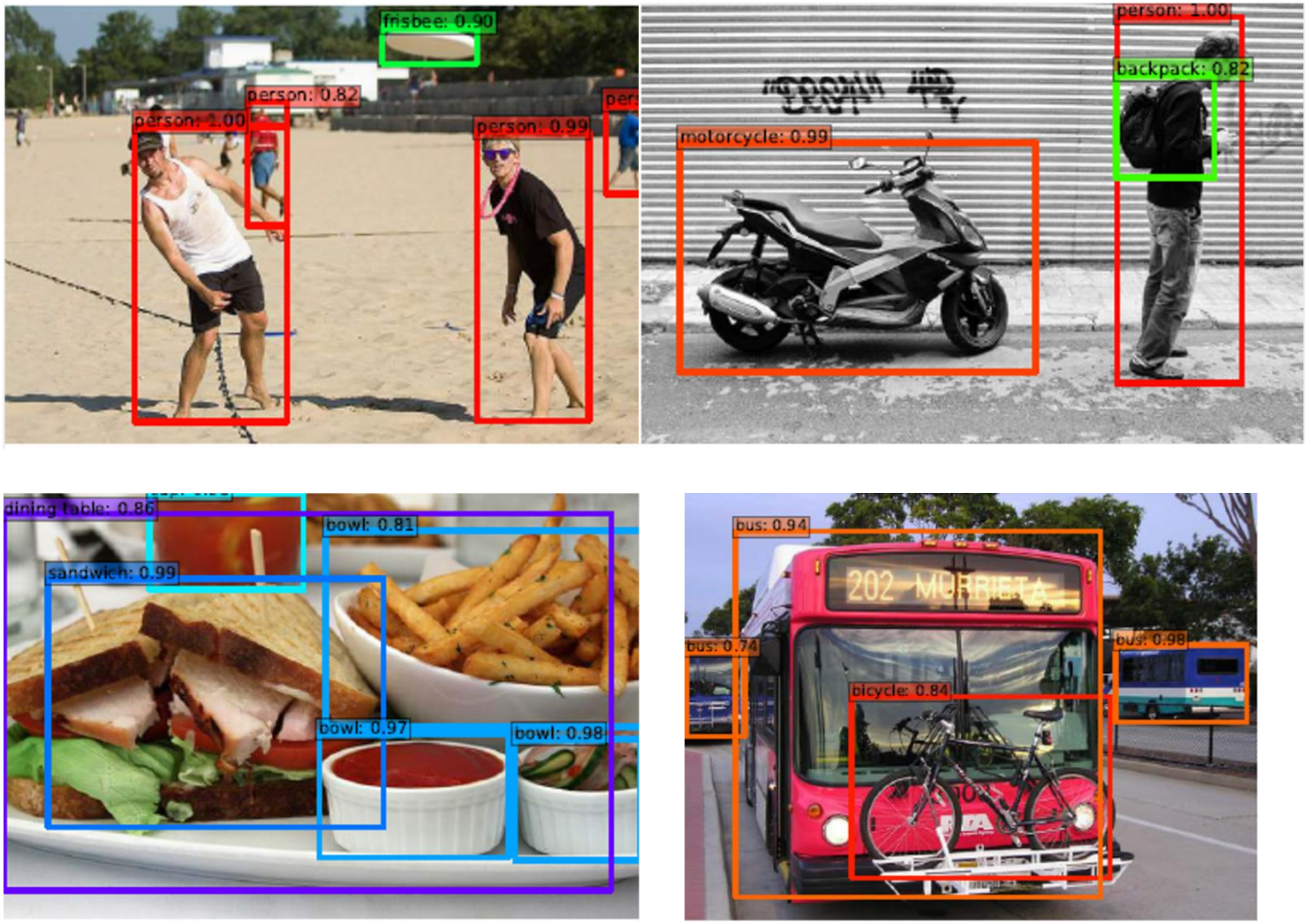
- Ground truth boxes needed for each object during training
- Evaluate a small set of default boxes of different aspect ratios at each locations in several feature maps with different scales
- Filters match these default boxes to the ground truth boxes and predict both the shape offsets and confidence for all object categories for each default box
- The feed-forward convolutional network produces a fixed-size collection of bounding boxes and scores for the presence of object class in those boxes

SSD Results


truck: 0.60
person: 0.83
person: 0.86
dog: 1.00


car: 0.9
car: 0.80
car: 0.74
truck: 0.96
car: 0.78
car: 0.91
car: 0.98
car: 0.72
car: 0.92
car: 0.99
person: 0.99

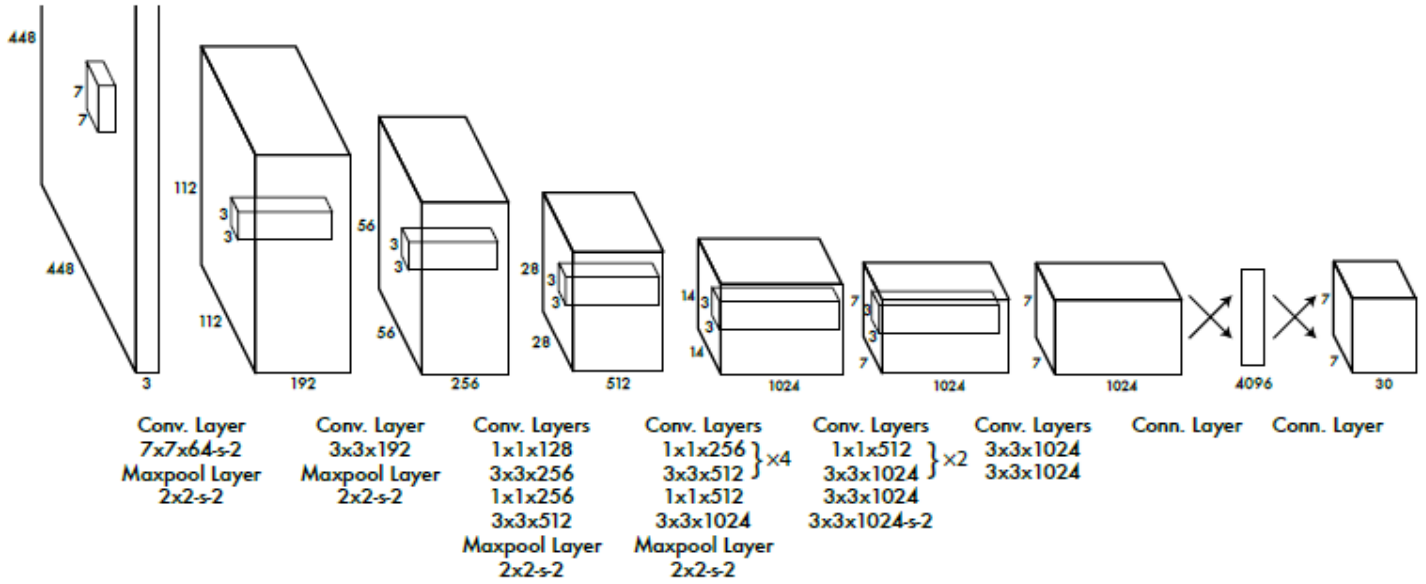
SSD Results


frisbee: 0.90
person: 0.82
person: 1.00
person: 0.99
motorcycle: 0.99
person: 1.00
backpack: 0.82
dining table: 0.86
bowl: 0.81
sandwich: 0.99
bowl: 0.97
bowl: 0.98
bus: 0.94
bus: 0.74
bicycle: 0.84
bus: 0.98

You Only Look Once (YOLO): Unified Real-Time Object Detection

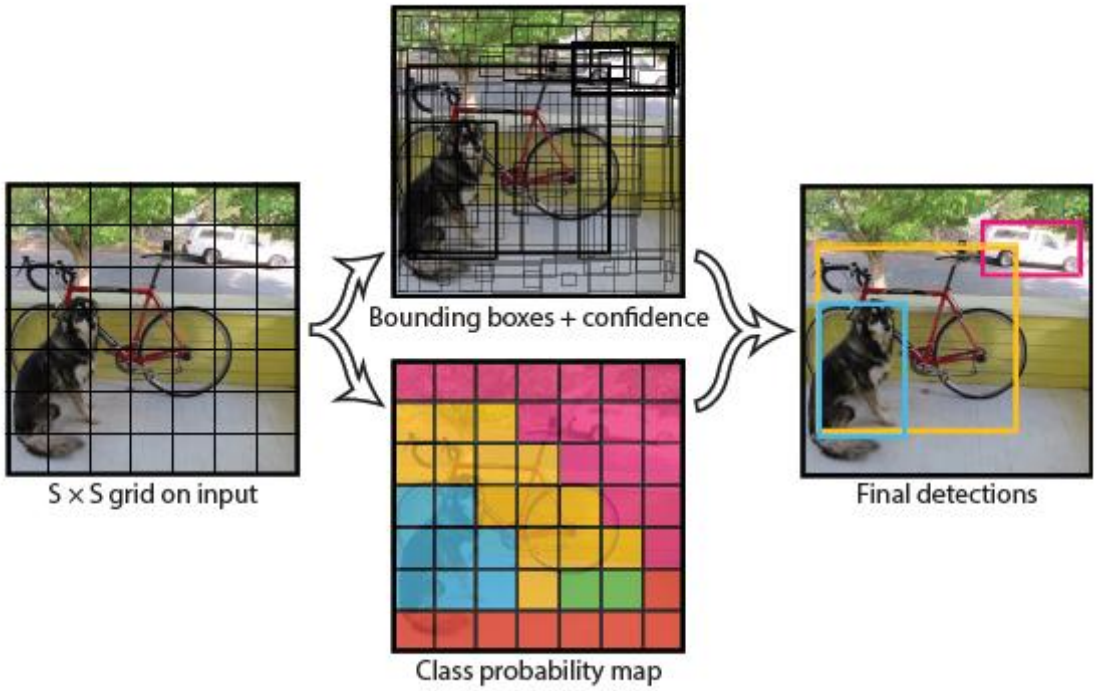
- A single neural network, unified architecture
- Related methods are slow, not real-time and devoid of generalization ability
- Advantages
  - Simpler structure of network
  - Much faster, even with real-time property (150 fps), able to process streaming video in real-time with less than 25 milliseconds of latency
  - Maintaining a proper accuracy range

You Only Look Once (YOLO): Unified Real-Time Object Detection



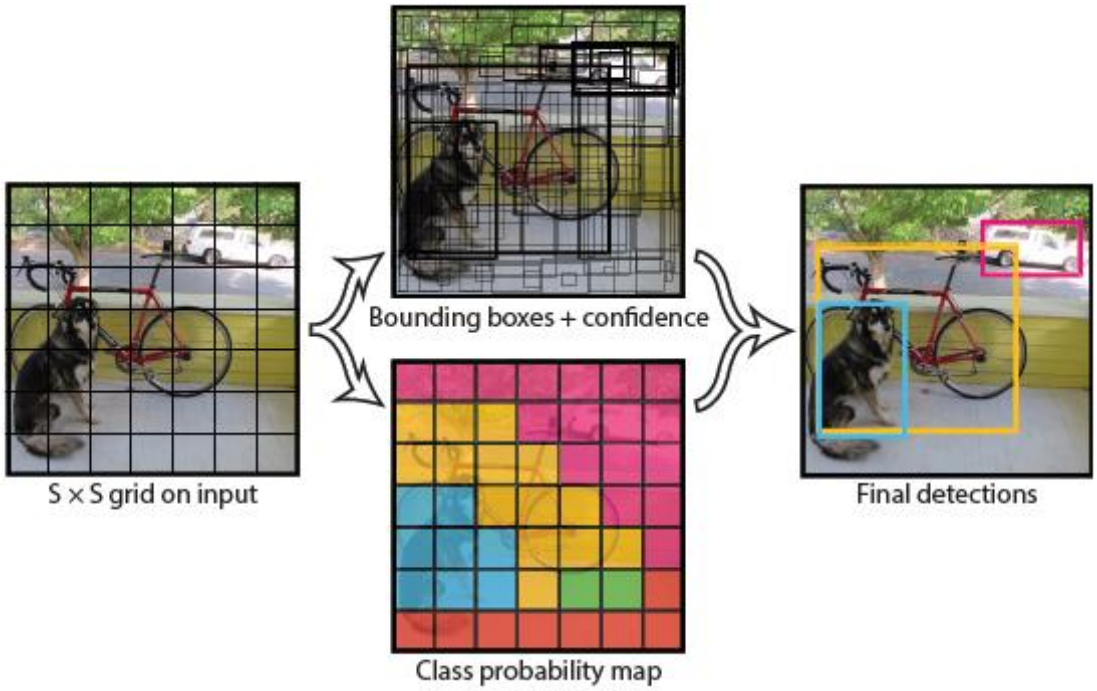
- Structure advantage for speed
  - No bounding box proposals and subsequent pixel or feature resampling stage
  - A neural network predicts bounding boxes and class probabilities directly from full images in one evaluation

Mechanics of YOLO



- Uses features from the entire image to predict each bounding box
- Predicts all bounding boxes across all classes for an image simultaneously
- Divides the input image into an s x s grid. If the center of an object falls into a grid cell, the cell is responsible for detecting that object
- Each grid cell predicts B bounding boxes and confidence scores for those boxes
- Each grid also predicts C conditional class probabilities

Mechanics of YOLO



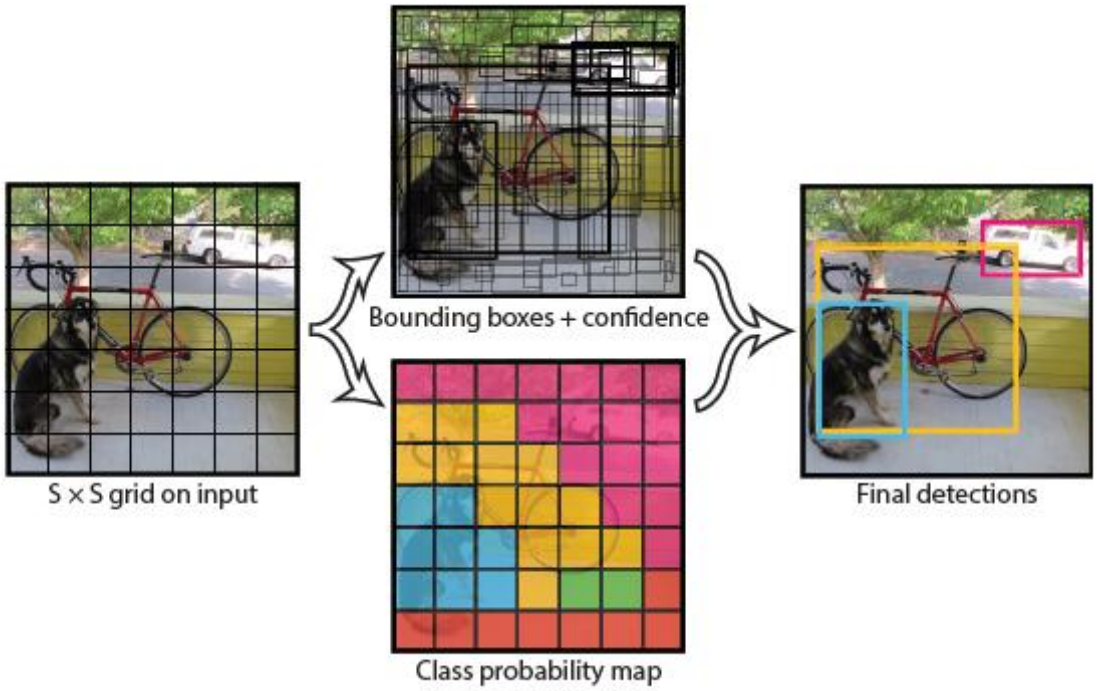
- **Confidence scores**: reflect how confident that the box contains and object and how accurate the box is

$$\Pr(\text{Object}) * \text{IoU}_{\text{pred}}^{\text{truth}}$$

- **Conditional class probabilities**: conditioned on the grid cell containing an object

$$\Pr(\text{Class}_i|\text{Object})$$

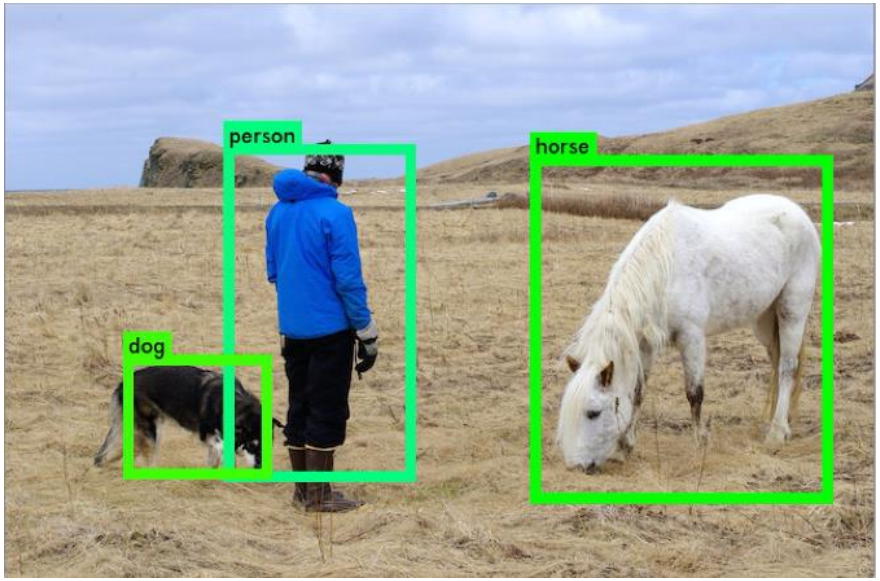
Mechanics of YOLO

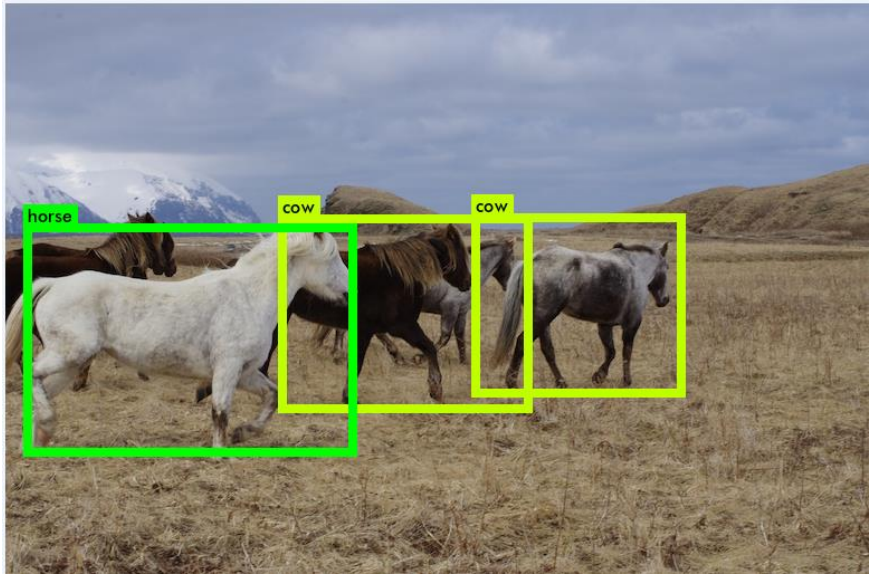


$$\Pr(\text{Class}_i|\text{Object}) * \Pr(\text{Object}) * \text{IoU}_{\text{pred}}^{\text{truth}} = \Pr(\text{Class}_i) * \text{IoU}_{\text{pred}}^{\text{truth}}$$

- At test time, multiply the conditional class probabilities and the individual box confidence predictions, giving class-specific confidence scores for each box

YOLO Results





Insights

- SSD:
  - Simplify its structure and speed up
- YOLO:
  - Make each cell predict more bounding boxes
  - Also have the idea of multi-scale in order to process small object features
- References:
  - SSD: http://machinethink.net/blog/object-detection/
  - YOLO: https://www.dlology.com/blog/gentle-guide-on-how-yolo-object-localization-works-with-keras/
  - YOLOv7 New Version - Improvements And Evaluation: https://arxiv.org/abs/2207.02696 https://github.com/WongKinYiu/yolov7

Object Detection:
Considerable Variables

Base Network
VGG16
ResNet-101
Inception V2
Inception V3
Inception
ResNet
MobileNet

Object Detection architecture
Faster R-CNN
R-FCN
SSD

Image Size
 # Region Proposals
…

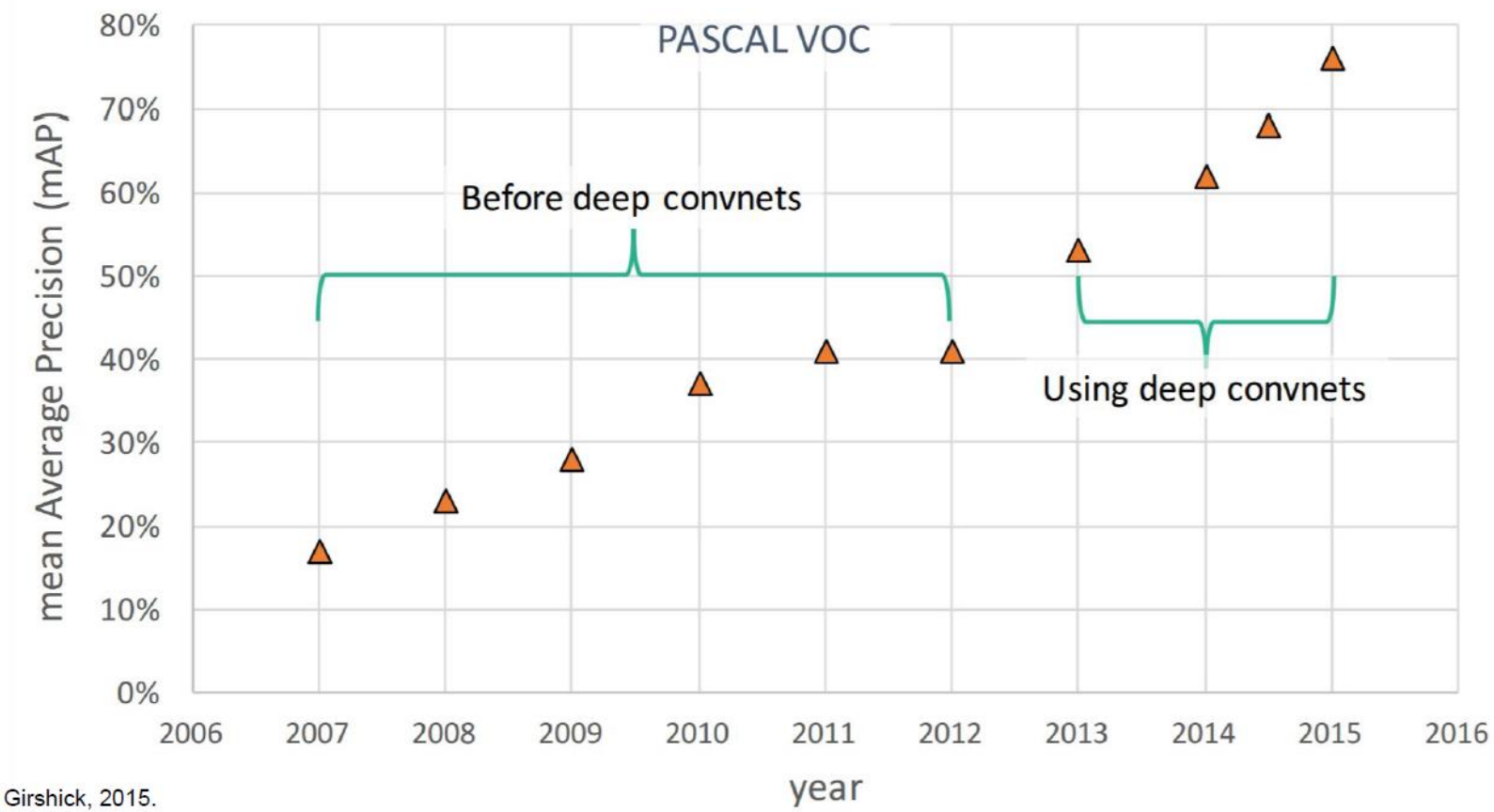
Takeaways
Faster R-CNN is slower but more accurate

SSD is much faster but not as accurate

Huang et al, "Speed/accuracy trade-offs for modern convolutional object detectors", CVPR 2017

R-FCN: Dai et al, "R-FCN: Object Detection via Region-based Fully Convolutional Networks", NIPS 2016
Inception-V2: Ioffe and Szegedy, "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift", ICML 2015
Inception V3: Szegedy et al, "Rethinking the Inception Architecture for Computer Vision", arXiv 2016
Inception ResNet: Szegedy et al, "Inception-V4, Inception-ResNet and the Impact of Residual Connections on Learning", arXiv 2016
MobileNet: Howard et al, "Efficient Convolutional Neural Networks for Mobile Vision Applications", arXiv 2017

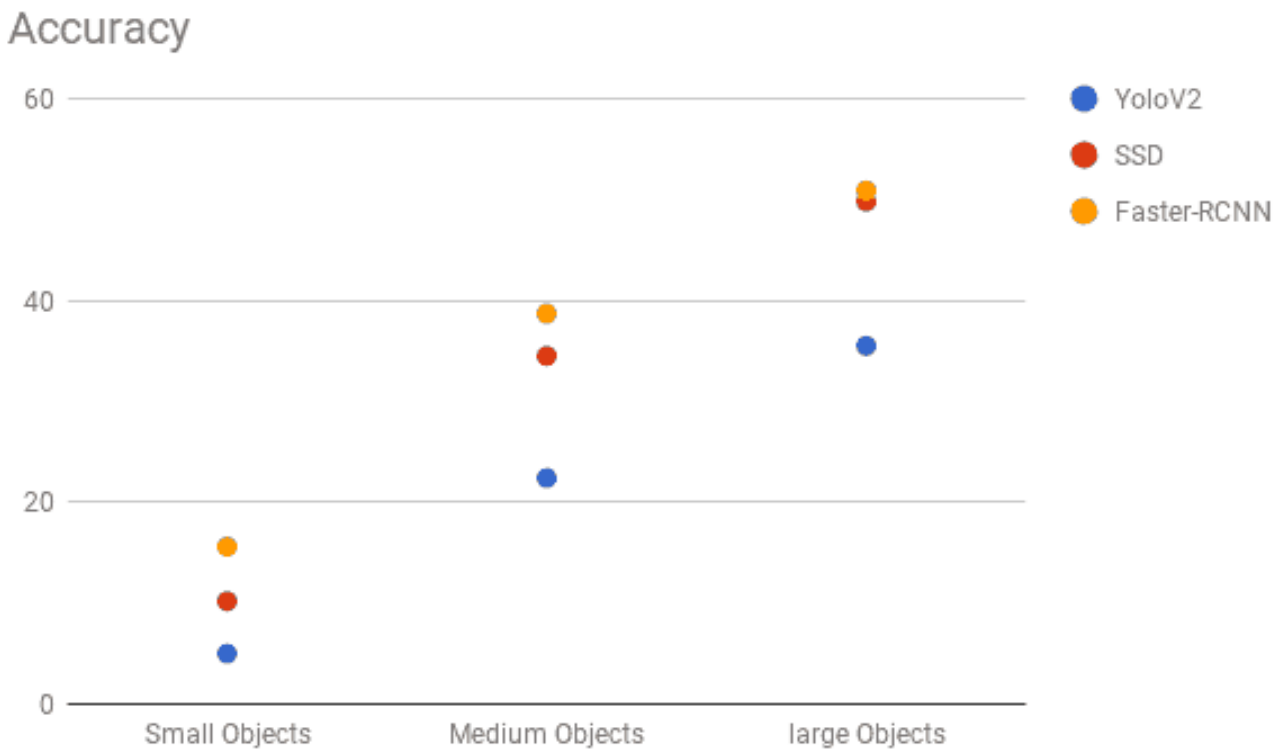
Object Detection:
Impact of Deep Learning



Object Detection:
Speed vs Accuracy trade-off

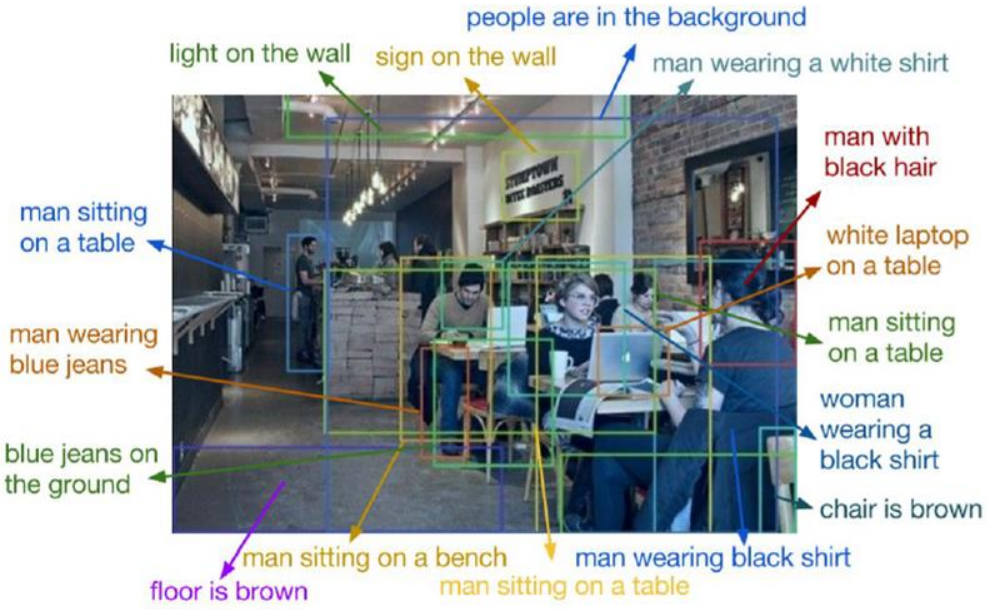
- SSD can be a good choice – can run on video and accuracy trade-off is little
- Faster-RCNN is the choice if accuracy is important.
- If speed is more important than accuracy, the YOLO will be the way to go.

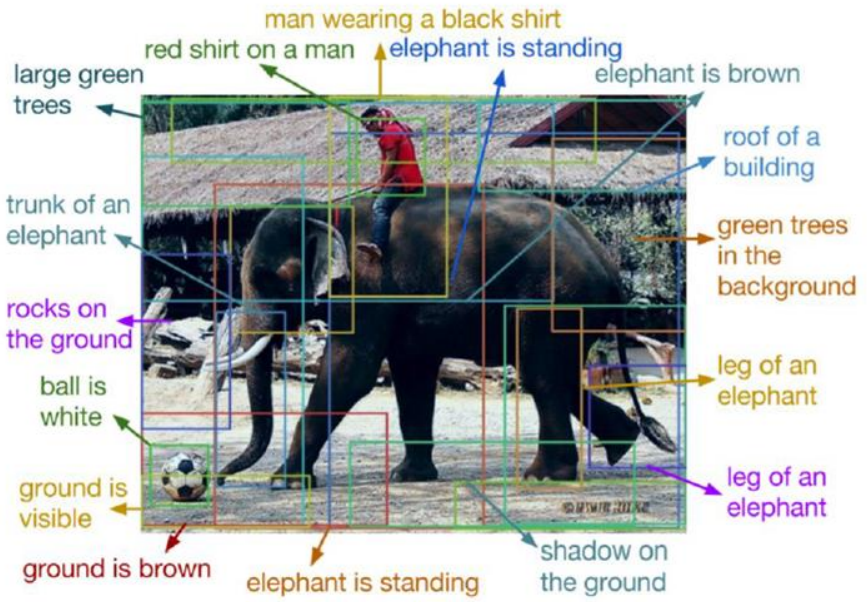
Object Detection:
Speed vs Accuracy trade-off



- At large sizes, SSD seems to perform similarly to Faster-RCNN.
- However, when the object size is small, the gap widens.

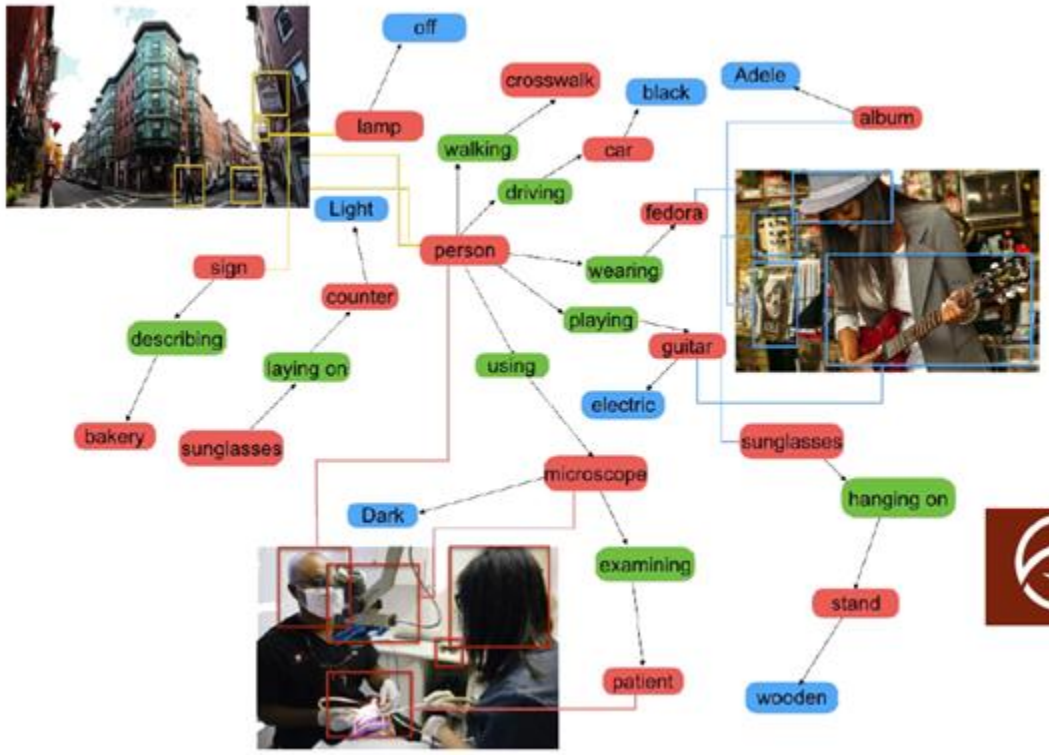
Object Detection + Captioning = Dense Captioning





Johnson, J., Karpathy, A., & Fei-Fei, L. (2016). Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4565-4574).

Visual Genome

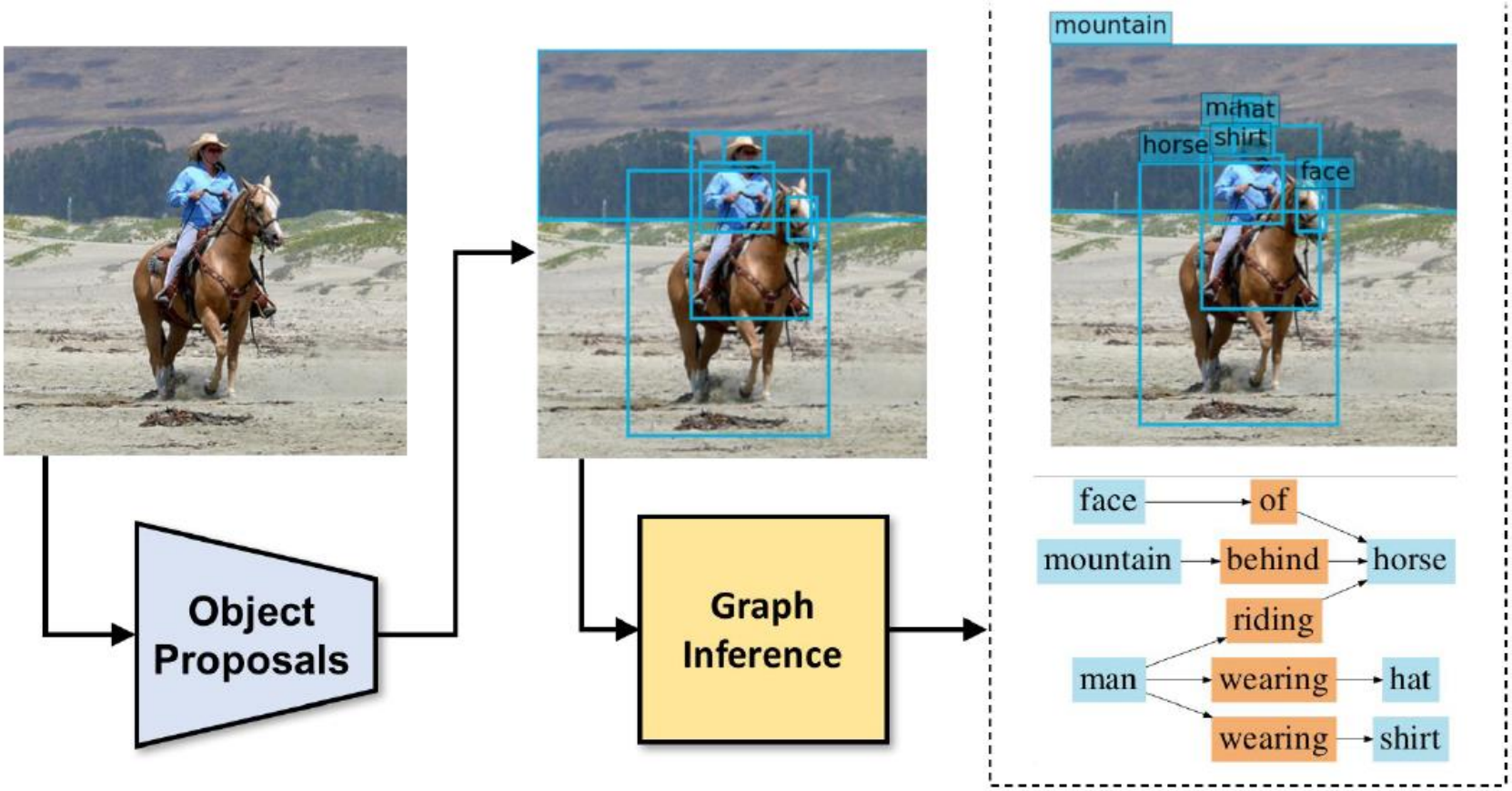


108,077 Images
5.4 Million Region Descriptions
1.7 Million Visual Question Answers
3.8 Million Object Instances
2.8 Million Attributes
2.3 Million Relationships
Everything Mapped to Wordnet Synsets

VISUALGENOME

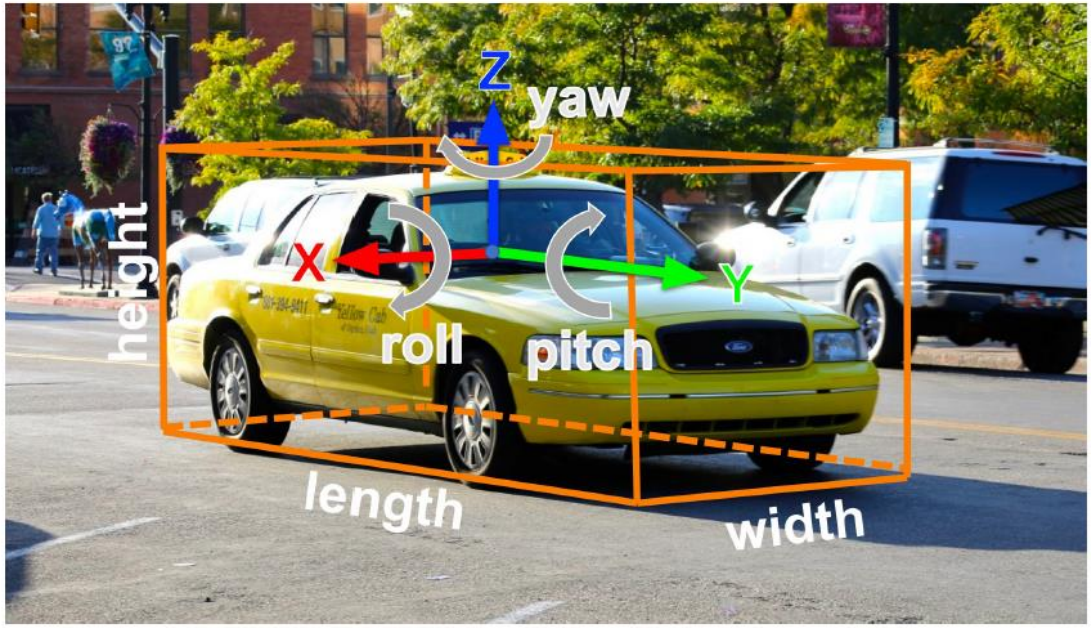
Krishna, R., Zhu, Y., Groth, O., Johnson, J., Hata, K., Kravitz, J., ... & Fei-Fei, L. (2017). Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International journal of computer vision*, *123*(1), 32-73.

Scene Graph Generation



Xu, D., Zhu, Y., Choy, C. B., & Fei-Fei, L. (2017). Scene graph generation by iterative message passing. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 5410-5419).

3D Object Detection



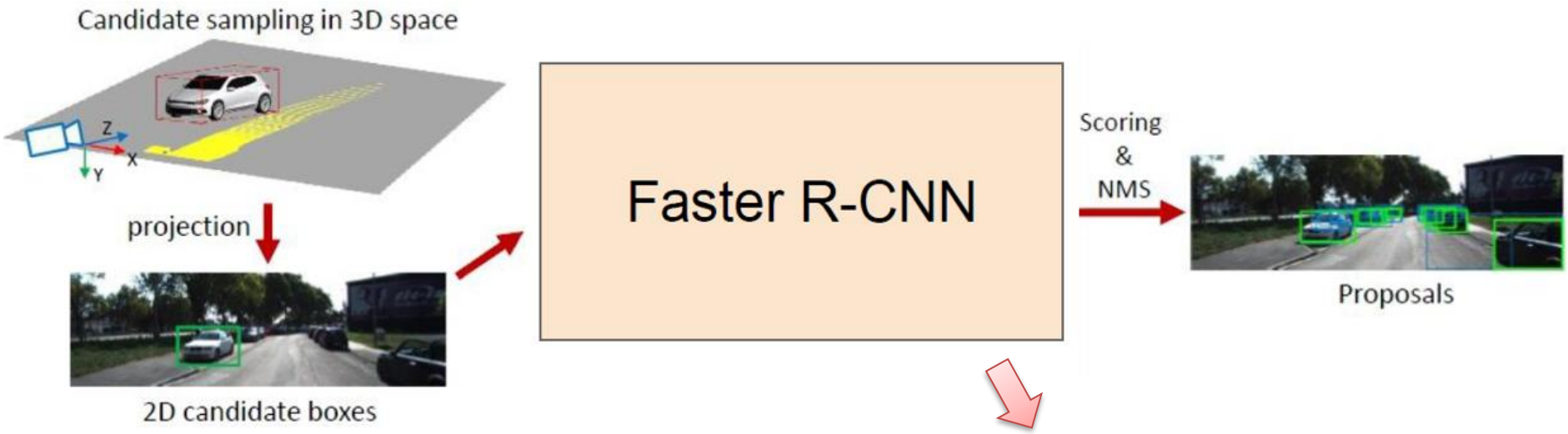
2D Object Detection:
2D bounding box
(x, y, w, h)

3D Object Detection:
3D oriented bounding box
(x, y, z, w, h, l, r, p, y)

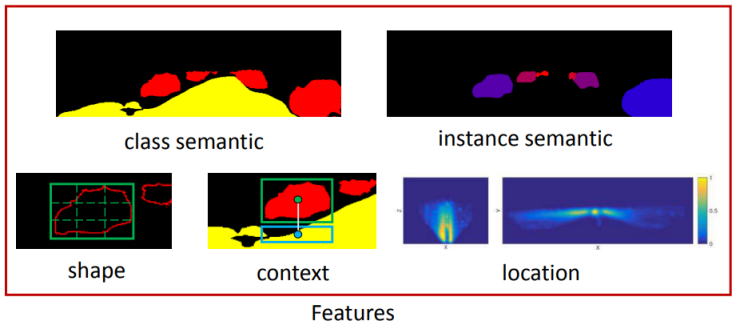
Simplified bbox: no roll & pitch

Much harder problem than 2D object detection!

3D Object Detection: Monocular Camera



- Same idea as Faster RCNN, but proposals are in 3D
- 3D bounding box proposal, regress 3D box parameters + class score



Chen, X., Kundu, K., Zhang, Z., Ma, H., Fidler, S., & Urtasun, R. (2016). Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2147-2156).

Boosting Deep Learning Performance

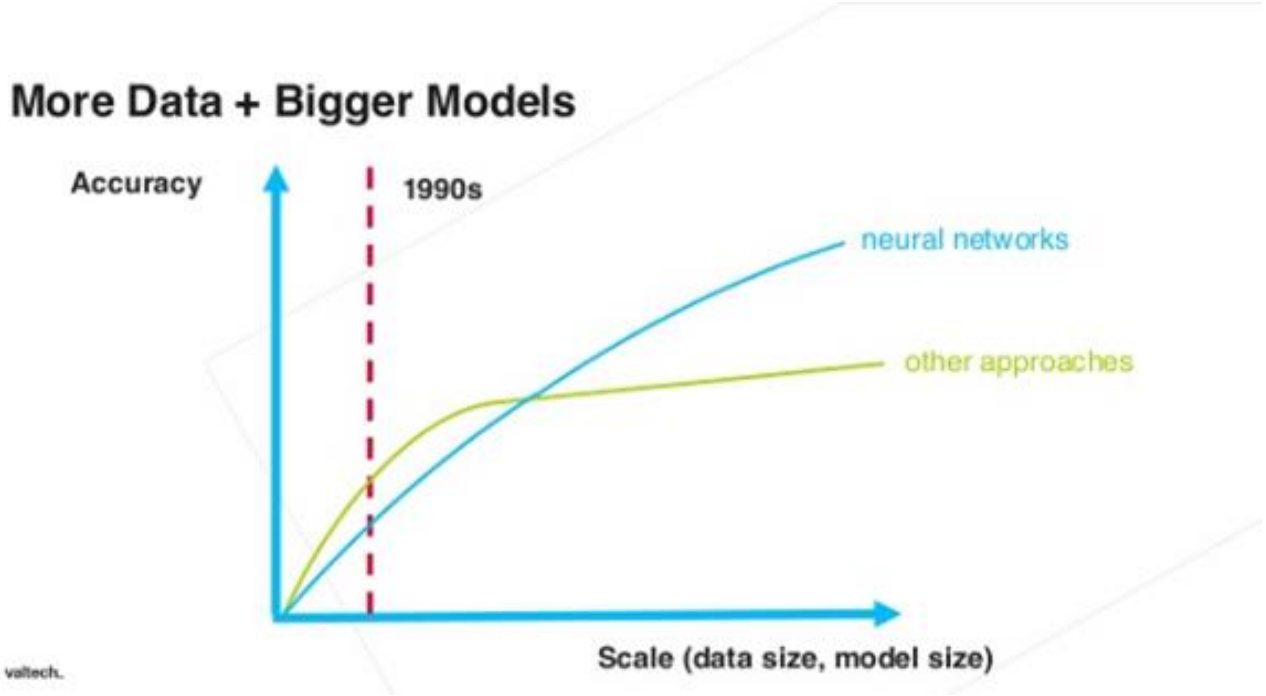
Strategies

- Data optimization
- Hyper-parameter tuning
  - Learning rate
  - Batch size and number of epochs
  - Early stopping
  - Regularization
- Ensemble

Data Optimization

- Balance the dataset
  - One of the easiest ways to increase performance for underperforming deep learning models (particularly for classification problems)
- Common methods
  - **Subsample majority class**: subsample from classes with large amount of images
  - **Sample minority class**: sampling with replacement to increase proportion of classes with small amount of images

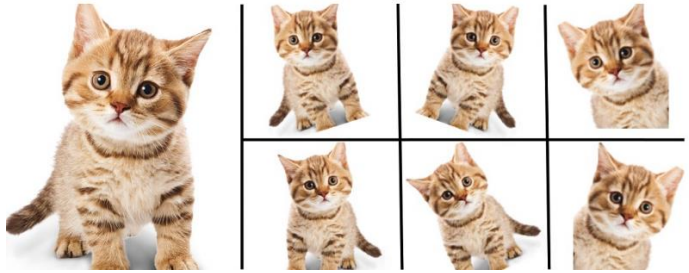
Data Optimization

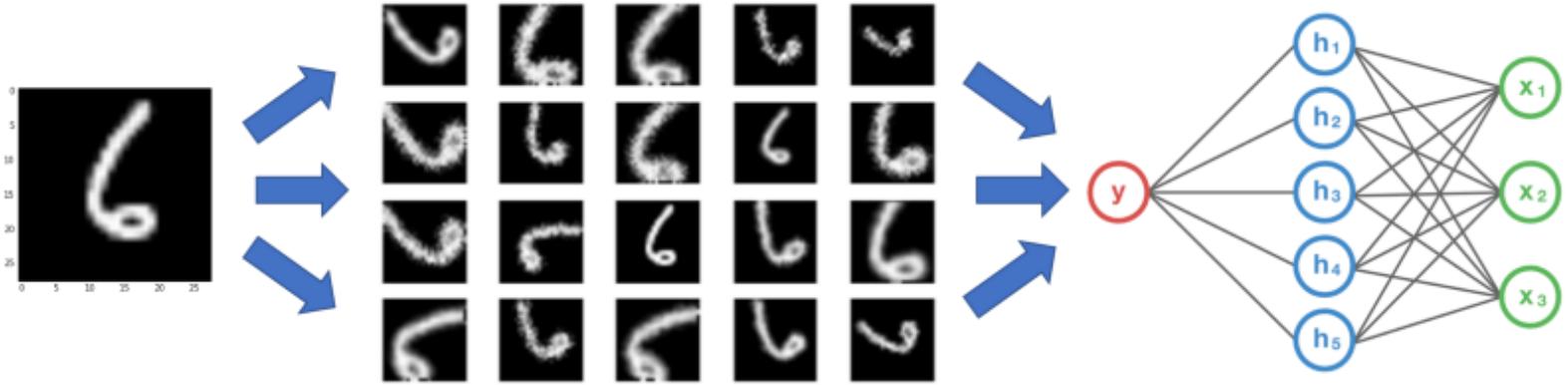


- For better model performance, more data is needed
- Depending on budget, might opt for creating more labeled data or collecting more unlabeled data for longer training on the feature extraction sub-model

Data Optimization

- Data augmentation
  - Getting more data when don't have "more data"
  - Make minor changes such as flips or translations or rotations to the existing dataset
  - Neural network treat them as distinct images





Data Optimization

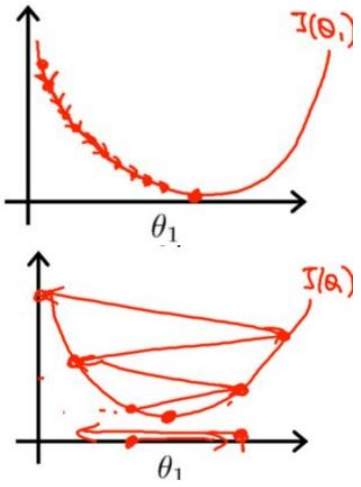
- Data augmentation
  - Can augmentation help even if there are already lots of data?
  - **Example**: A dataset consists of **two brans of cars**, as show below.

    Assume all cars of **brand A (Ford)** and brand **B (Chevrolet)** are aligned exactly like the picture in the left and in the right respectively


  - Feed the dataset to a "state-of-the-art" neural network
  - Given the car on the right (Ford), can the network classify it correctly?
  - **Conclusion**: By performing augmentation, can prevent neural network from learning irrelevant patterns, essentially boosting overall performance.

Hyper-parameter Optimization

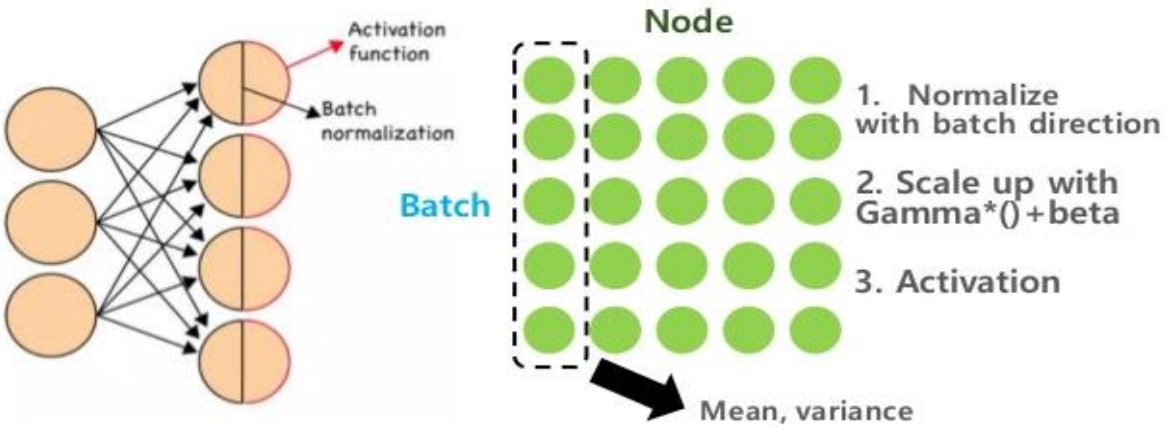
- Learning rates
  - Deep learning models are typically trained by a stochastic gradient descent optimizer (SGD); Adam, RMSProp, Adagrad, etc.
  - These optimizers let you set the learning rate, which tells the optimizer how far to move the weights in the direction opposite of the gradient for a mini-batch.
  - **Low learning rate**: training is more reliable, optimization will take a lot of time because steps towards the minimum of the loss function are tiny.
  - **High learning rate**: training may not converge or even diverge. Weight changes can be so big that the optimizer overshoots the minimum and makes the loss worse.



  - Estimating an Optimal Learning Rate For a Deep Neural Network: https://towardsdatascience.com/estimating-optimal-learning-rate-for-a-deep-neural-network-ce32f2556ce0

Hyper-parameter Optimization

- Batch size and number of epochs
  - A standard procedure is using large batch sizes with a large number of epochs
  - Experiment with the size of your batches and the number of training epochs.
  - Batch Normalization (transforming the inputs to be mean 0 and unit variance).



Hyper-parameter Optimization

- Early Stopping
  - To improve real-world performance
  - Excellent method for reducing the generalization error of your deep learning system.
  - Continual training might improve accuracy on your data set, but at a certain point, it starts to reduce the model's accuracy
  - Stop the training before model accuracy gets worse

Hyper-parameter Optimization

- Regularization
  - Robust method to stop overfitting.
  - Ways to perform regularization:
    - **Dropout**: Randomly turn off a percentage of neurons during training. Dropout helps prevent groups of neurons from overfitting to themselves.
    - **Weight penalty L1 and L2**: Weights that explode in size (gradient explosion) can be a real problem in deep learning and reduce accuracy. One of the ways to combat this is to add decay to all weights. Keep all of the weights in the networks as small as possible unless there are large gradients to counteract it.

Ensemble

- Ensemble the outputs from the different models and get better accuracy.
- There are two steps for every one of these algorithms:
  - Produce a distribution of simple ML models on subsets of the original data
  - Combining the distribution into one "Aggregated" model
- **Combined Models/Views (Bagging)**
  - Train a few different models, which are different in some way, on the same data and average out the outputs to create the final output.
  - Bagging has the effect of reducing variance in the model.
- **Stacking**
  - Similar to bagging – the difference here is that you don't have an empirical formula for combined output.
  - Instead, create a meta-level learner that based on the input data chooses how to weight the answers from different models to produce the final output

Summary

- Deep Learning
  - Segmentation
  - Detection, and Localization
- Boosting deep learning performance
  - Data optimization
  - Hyper-parameter tuning
  - Ensemble

Further Studies and Resources

- Deep Learning Courses
  - deeplearning.ai: https://www.deeplearning.ai/
  - Stanford: http://cs231n.stanford.edu/
  - MIT: https://deeplearning.mit.edu/
  - Coursera: https://www.coursera.org/specializations/deep-learning
- YouTube Lectures
  - Assoc. Prof. Lee Hung-Yi, National Taiwan University (NTU): https://www.youtube.com/@HungyiLeeNTU/playlists
- Vision Transformer Coding Tutorial: https://keras.io/examples/vision/image_classification_with_vision_transformer/
- Visualization of deep features: https://github.com/jacobgil/keras-grad-cam