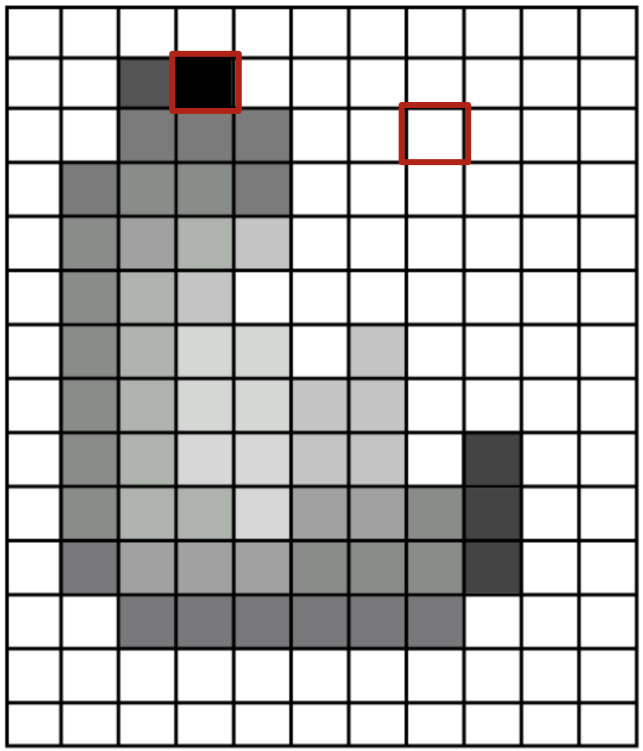
Topic 3: Deep Learning in Computer Vision

Image Representation: Matrices

- **Matrix**: Grayscale images



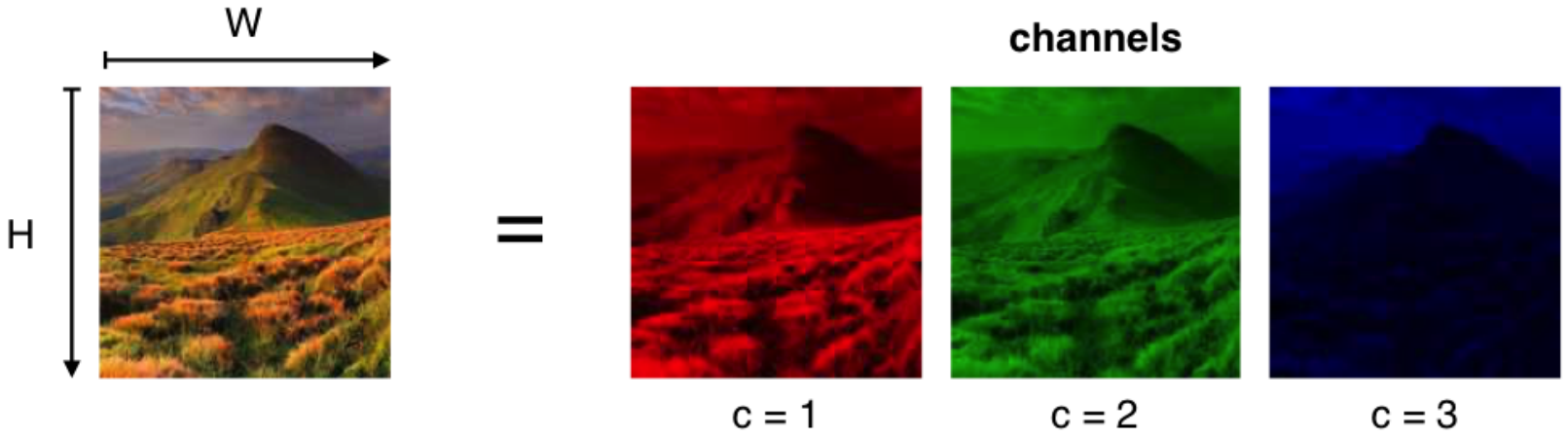
=

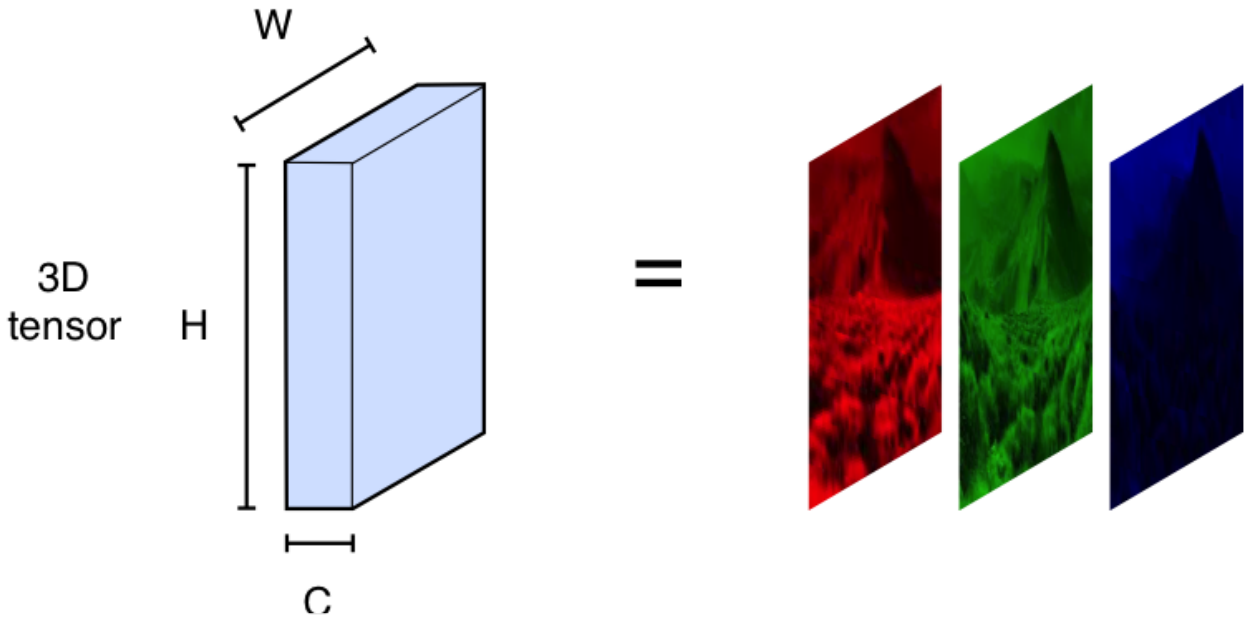
255	255	255	255	255	255	255	255	255	255	255
255	255	20	0	255	255	255	255	255	255	255
255	255	75	75	255	255	255	255	255	255	255
255	75	95	95	75	255	255	255	255	255	255
255	96	127	145	175	255	255	255	255	255	255
255	127	145	175	175	175	255	255	255	255	255
255	127	145	200	200	175	175	95	255	255	255
255	127	145	200	200	175	175	95	47	255	255
255	127	145	145	175	127	127	95	47	255	255
255	74	127	127	127	95	95	95	47	255	255
255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255

0 = black; 255 = white

Image Representation: Tensors

- **Tensor**: Most common form of representation



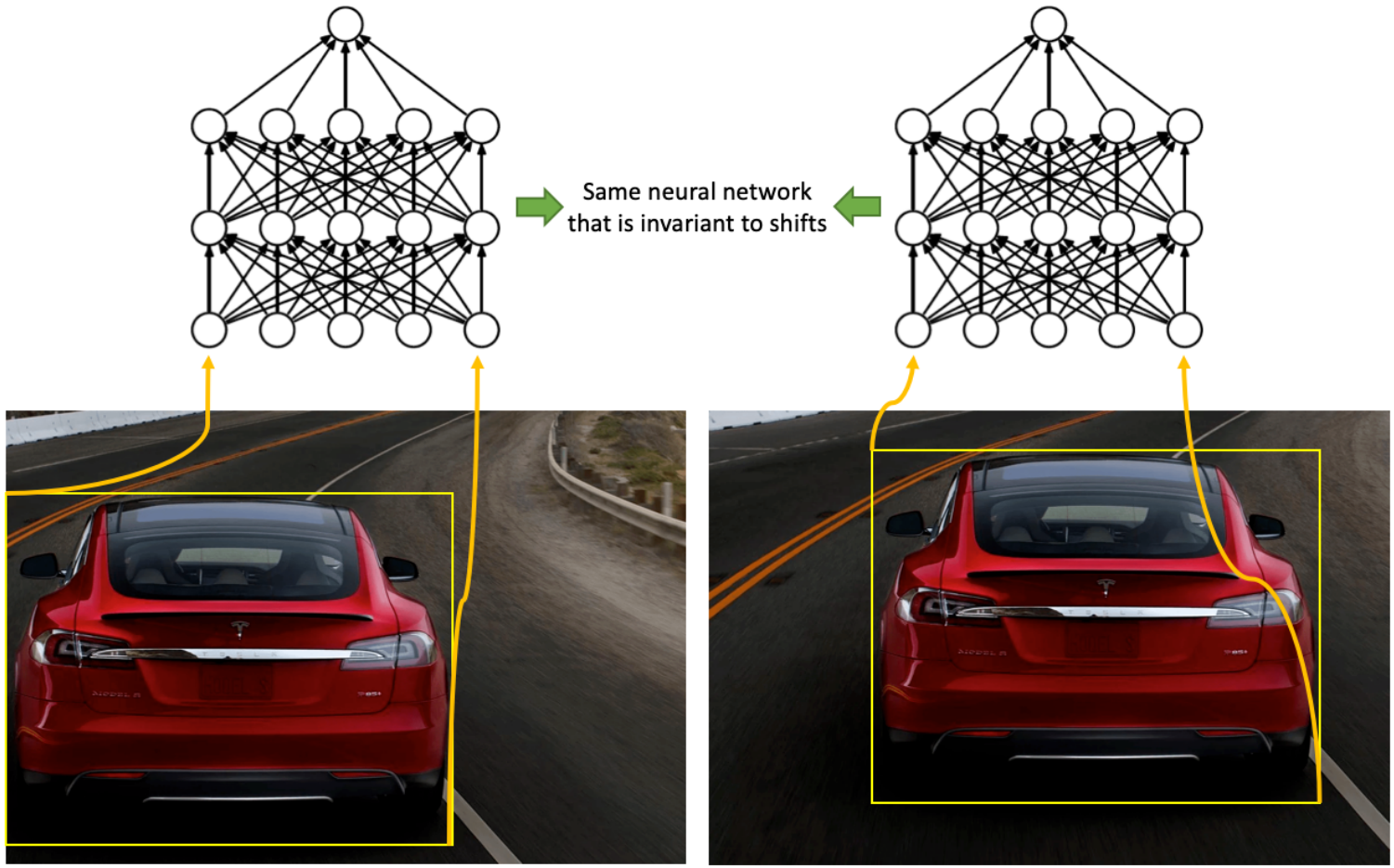


Computer Vision: An Collection of Exciting/Challenging Problems

- Object Detection
- Object Localization
- Object Tracking
- Segmentation
- Alignment

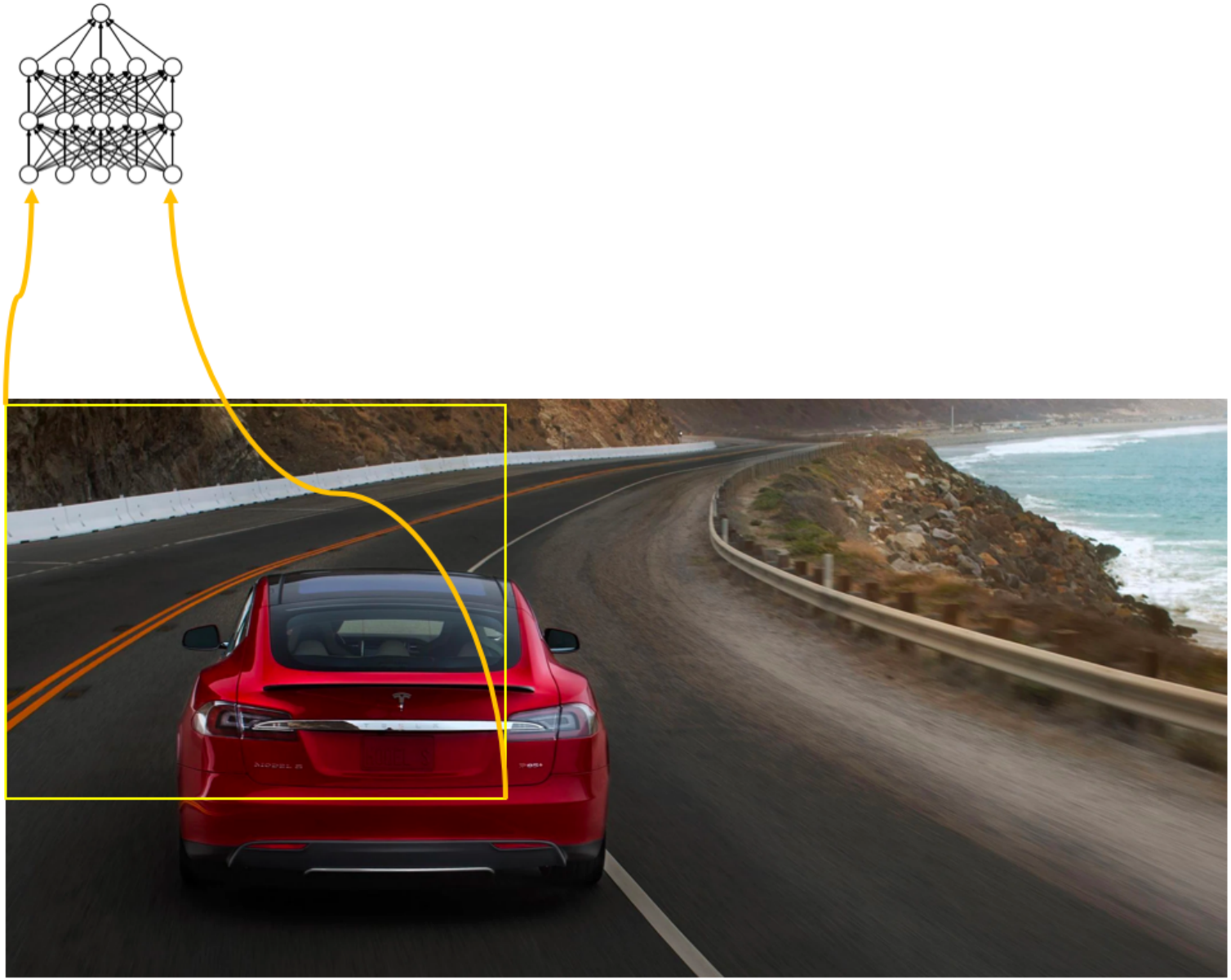
Object Detection with NNs: Need for Shift Invariance

- Will a neural network trained to detect a car in the left corner of the image, detect a car in a different location?
  - Conventional neural networks are sensitive to *shifts*.
  - Need one that detects the presence of car regardless of its location.



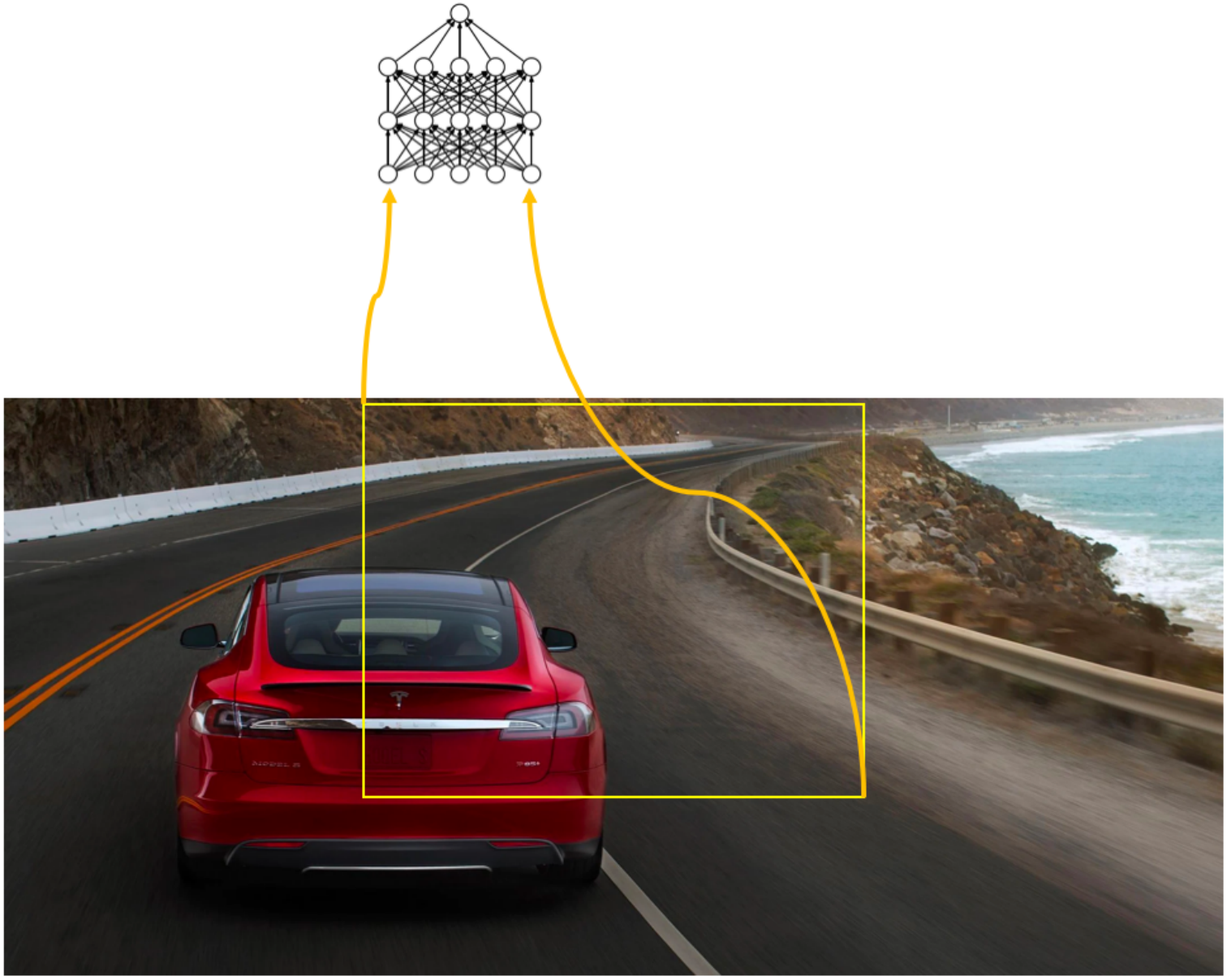
Solution: Scan the Image

- Scan for the target object at each location using a patch (kernel) of size $K \times K$.



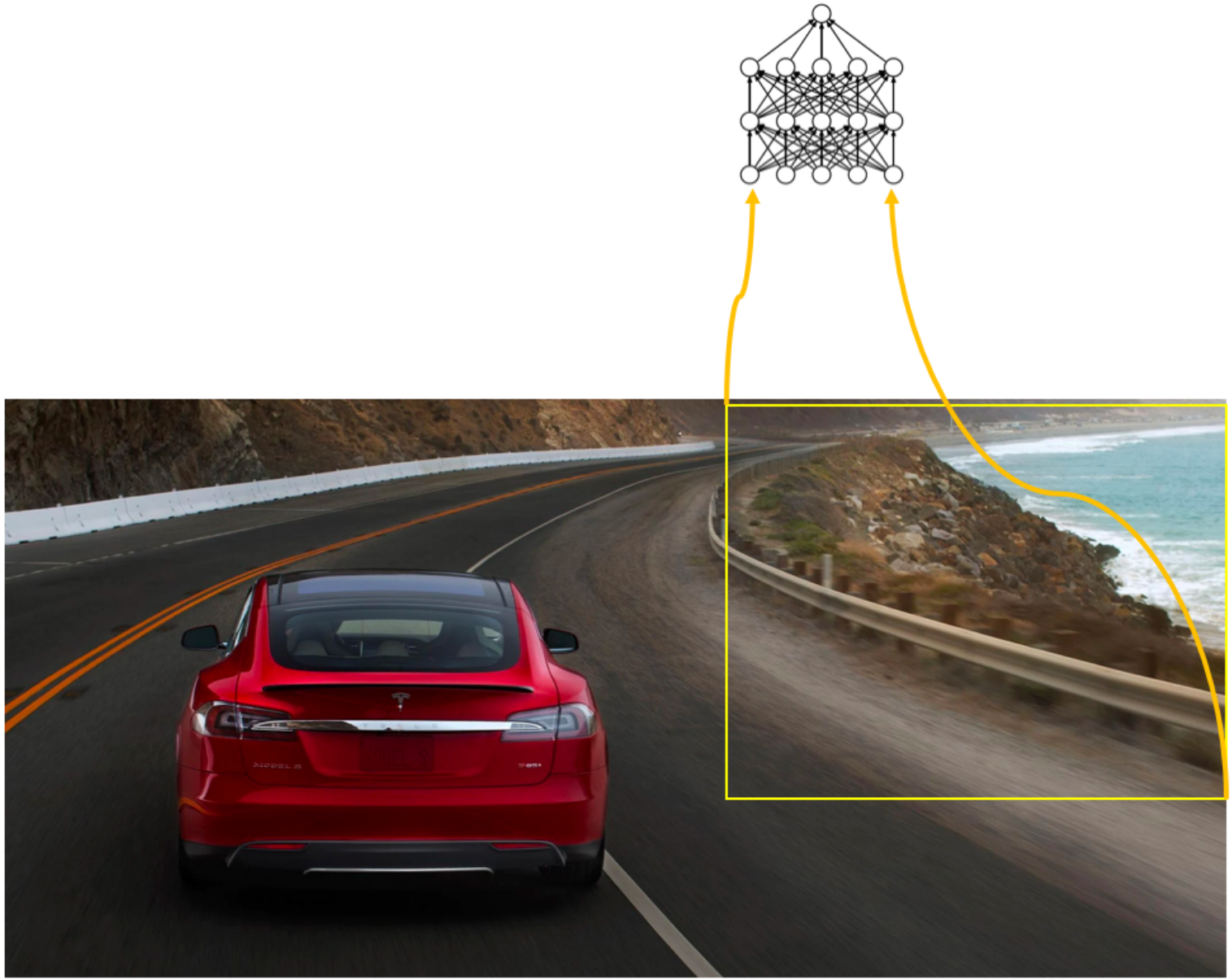
Solution: Scan the Image

- Scan for the target object at each location using a patch (kernel) of size $K \times K$.



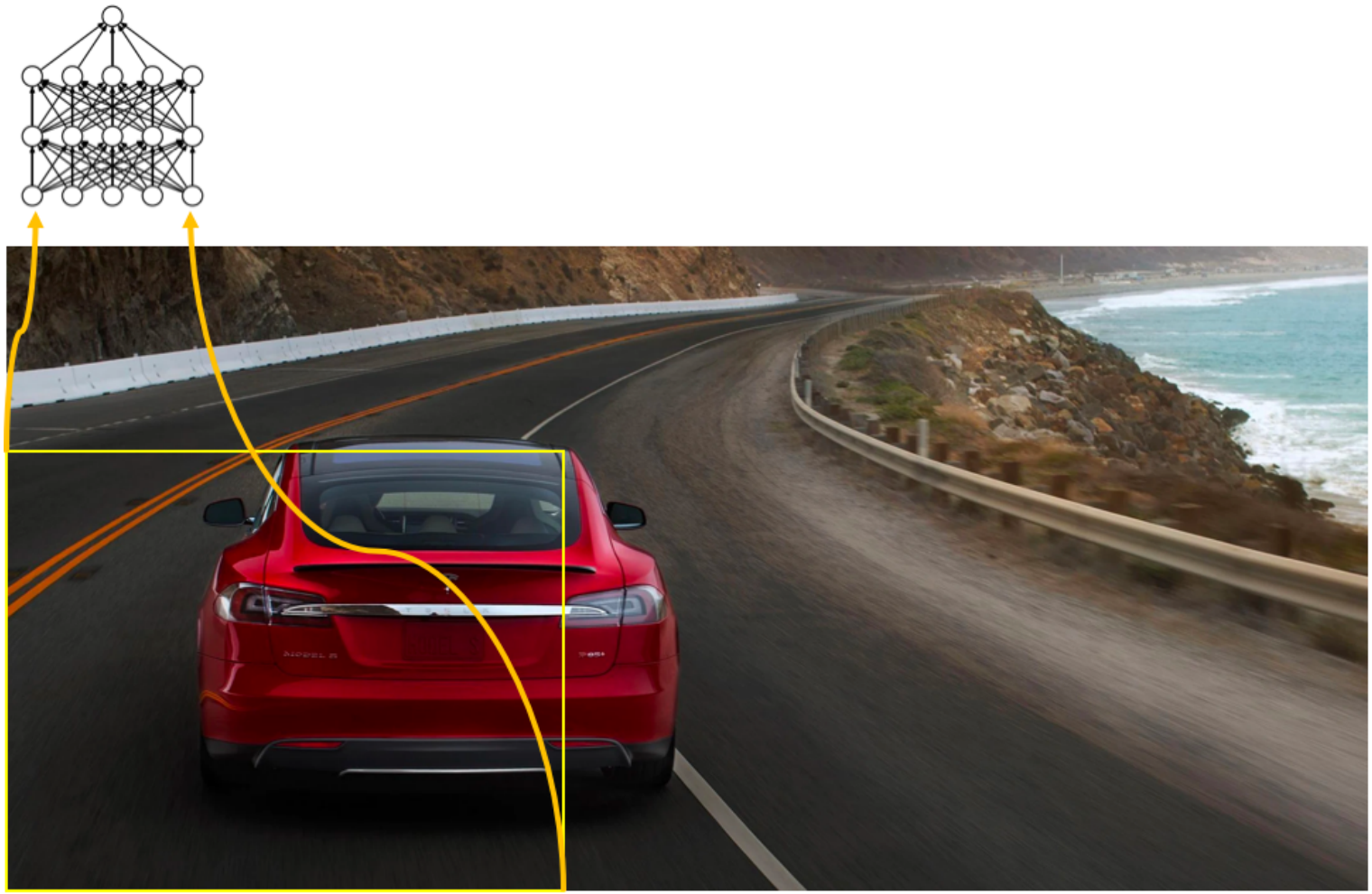
Solution: Scan the Image

- Scan for the target object at each location using a patch (kernel) of size $K \times K$.



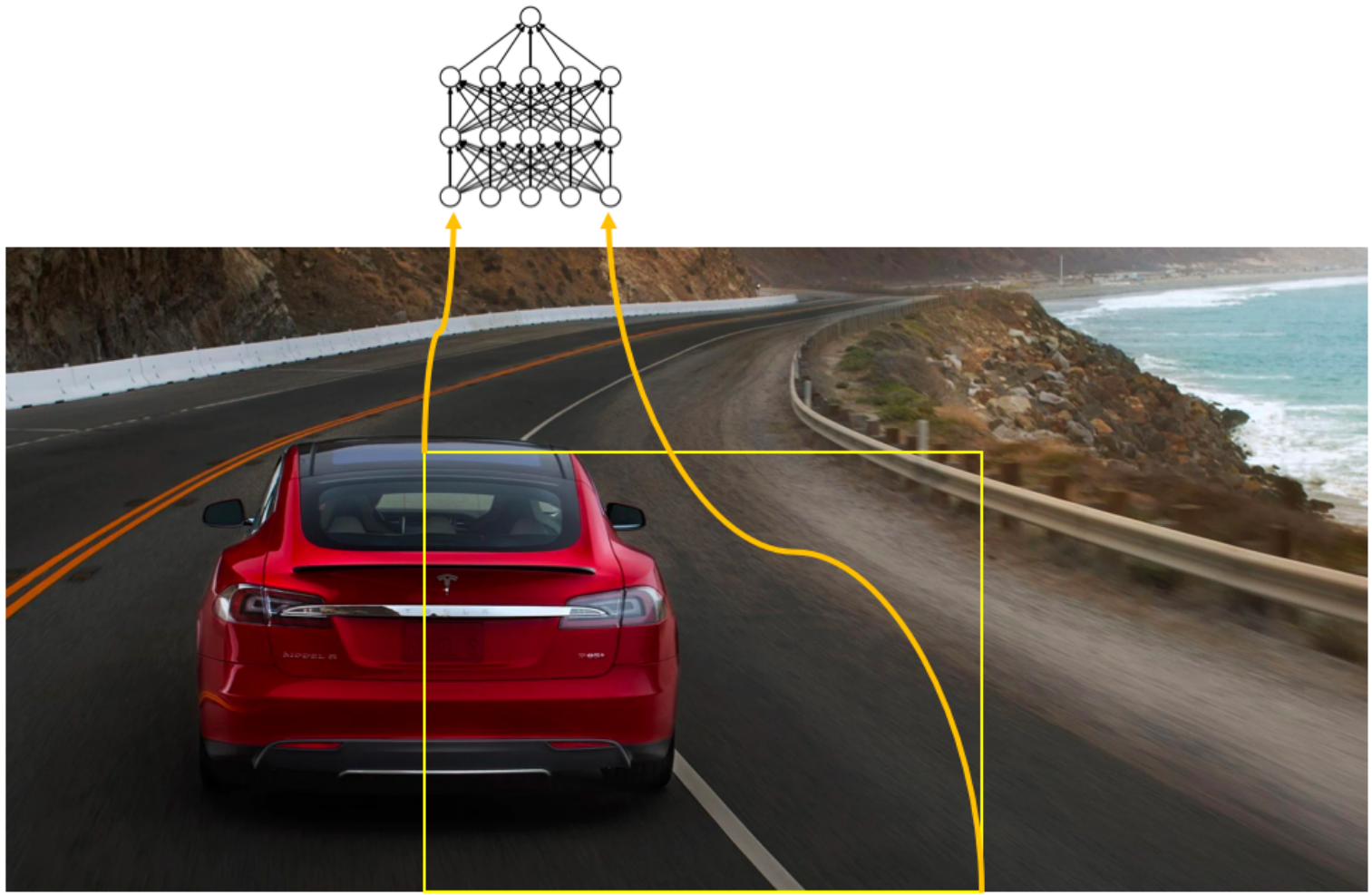
Solution: Scan the Image

- Scan for the target object at each location using a patch (kernel) of size $K \times K$.



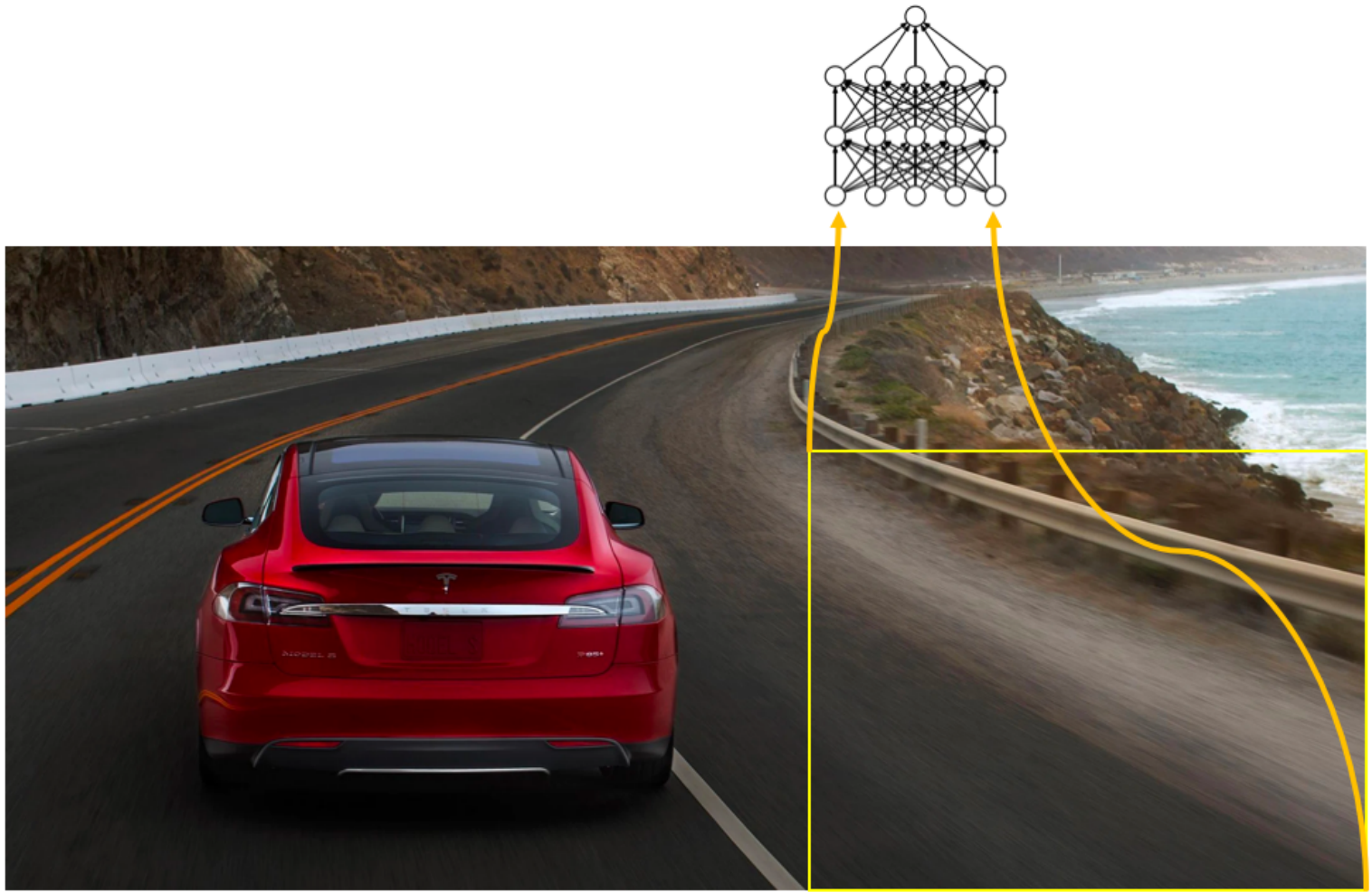
Solution: Scan the Image

- Scan for the target object at each location using a patch (kernel) of size $K \times K$.

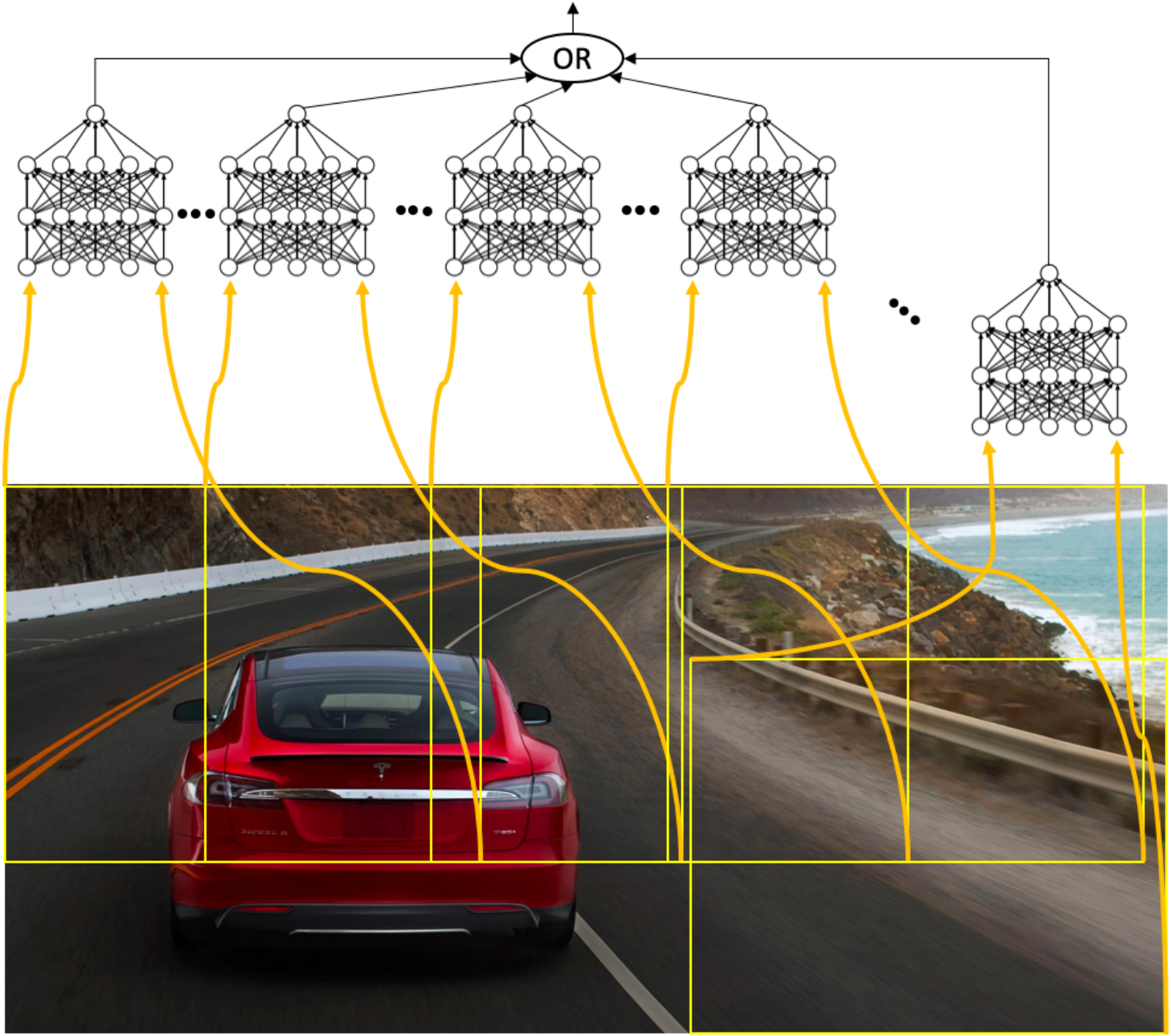


Solution: Scan the Image

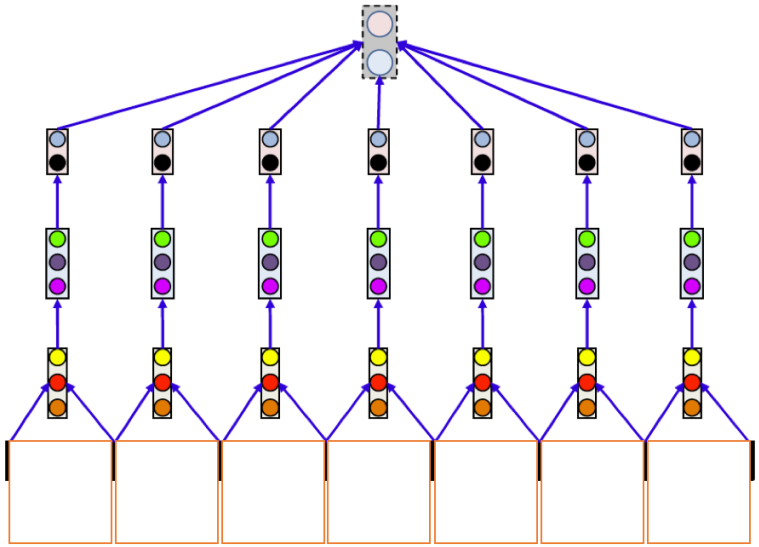
- Scan for the target object at each location using a patch (kernel) of size $K \times K$.



Combining all the scans, we get a Giant Neural Network!



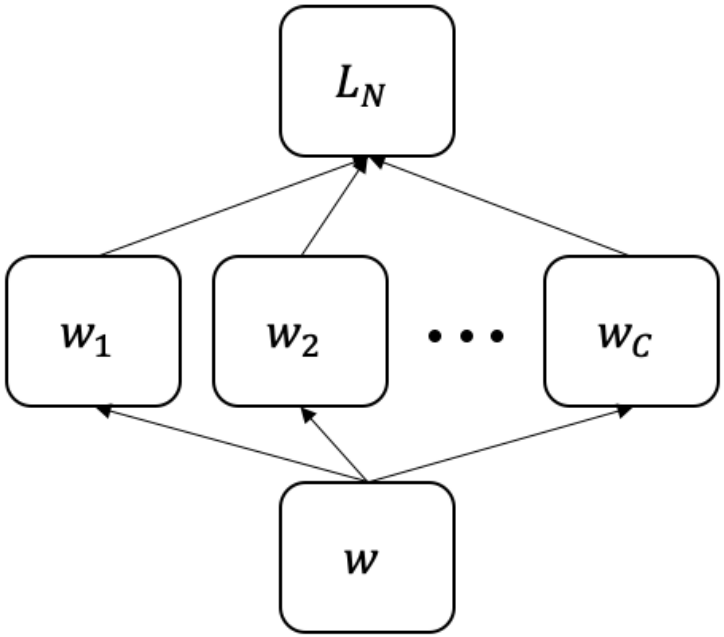
Note: This is a Shared Parameter Network!



```
def giant_nn(Img,K):
    W = width(Img)
    H = height(Img)
    C = num_channels(Img)
    for i = 1:W-K+1
        for j = 1:H-K+1
            ImgSegment = Img(1:C,i:i+K-1, j:j+K-1)
            y(i,j) = NN(ImgSegment)
    Y = OR(y(1,1), ..., y(W-K+1,H-K+1))
```

- Note that this is not the same as regular NN models.
- **Shared Parameter Networks**:
  - `NN()` uses the same weights for different locations
  - Block-structured weight matrix, with identical blocks for each kernel scan!
  - In other words, any parameter update of one scan must update equally for all scans.

Training Shared Parameter Networks



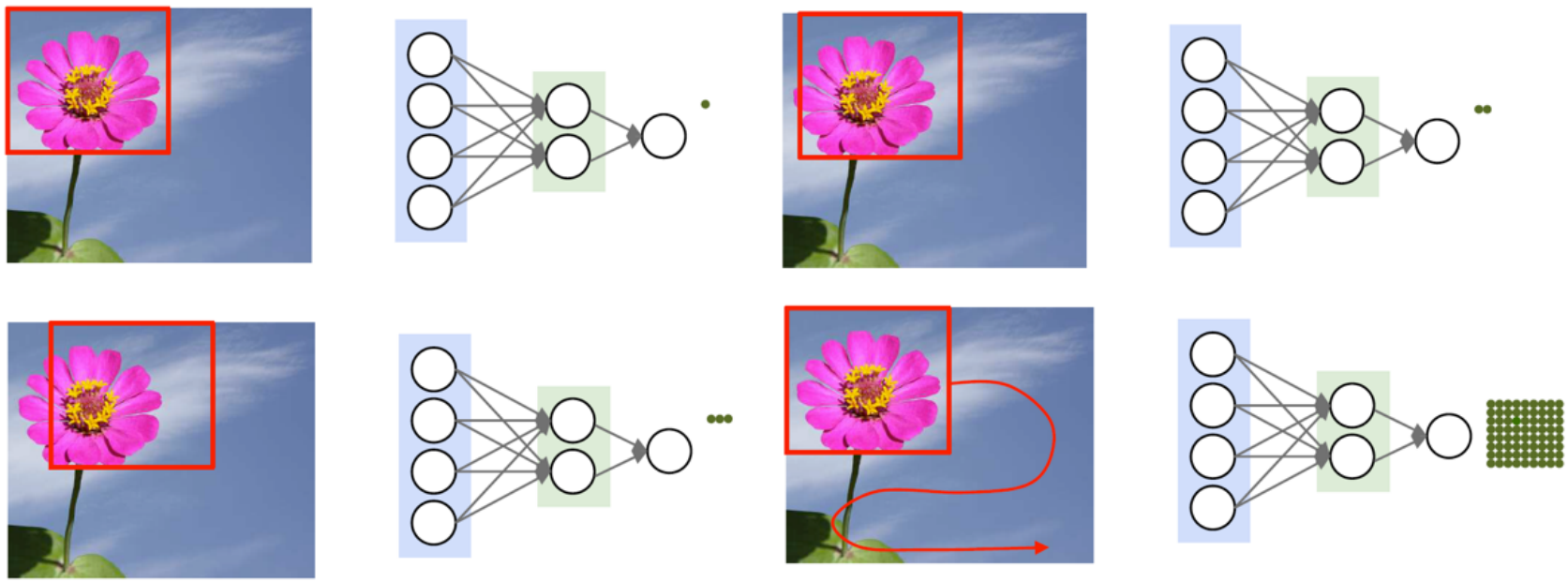
$$\frac{\partial L_N}{\partial w} = \sum_{c=1}^{C} \frac{\partial L_N}{\partial w_c} \cdot \frac{\partial w_c}{\partial w}$$

$$= \sum_{c=1}^{C} \frac{\partial L_N}{\partial w_c}.$$

- Say, we have L layers in `NN()` $\Rightarrow$ Let the weights be $W_1, \cdots, W_L$
- Let $\mathcal{S}$ denote the set of all edges with common value
- $\nabla_{w_{\mathcal{S}}} L_N = \dfrac{\partial L_N}{\partial w_{\mathcal{S}}} = \displaystyle\sum_{c \in \mathcal{S}} \frac{\partial L_N}{\partial w_c}.$
- Update step: $w_{\mathcal{S}} \leftarrow w_{\mathcal{S}} - \eta \nabla_{w_{\mathcal{S}}} L_N$,
- For every $(\ell, i, j) \in \mathcal{S}$, define $w^{\ell}_{i,j} = w_{\mathcal{S}}$.
- Continue until the stopping criterion is satisfied.

Take a Closer Look at Scanning...

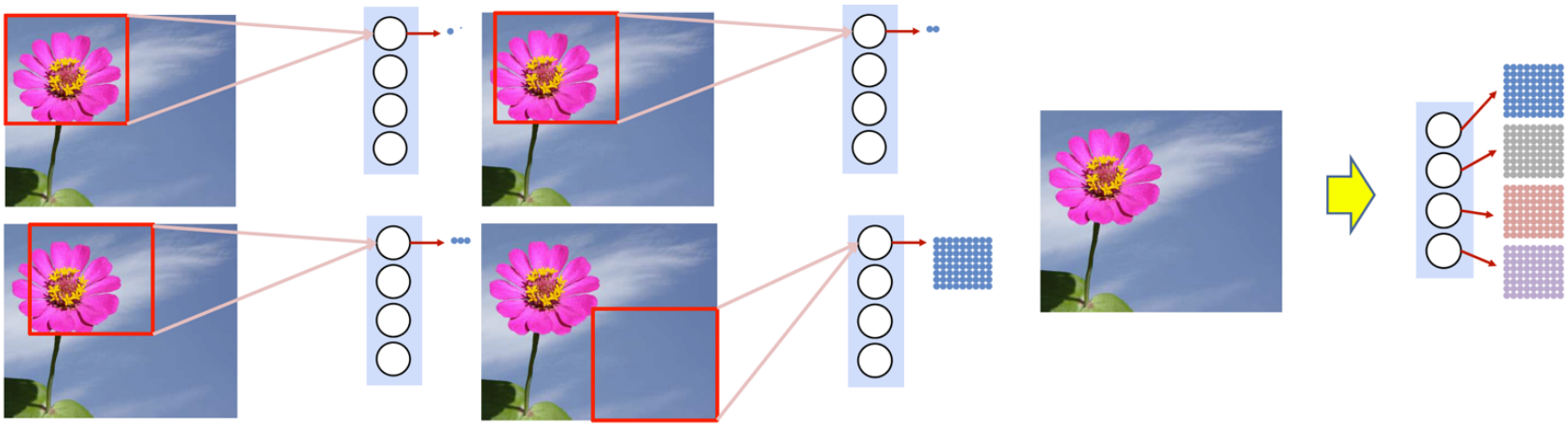
- Each scan passes the image segment through the entire network and produces an outcome
- Combining all the scans, we get a collection of outcomes, which are passed through OR (or softmax, or any other flat-NN) gate.



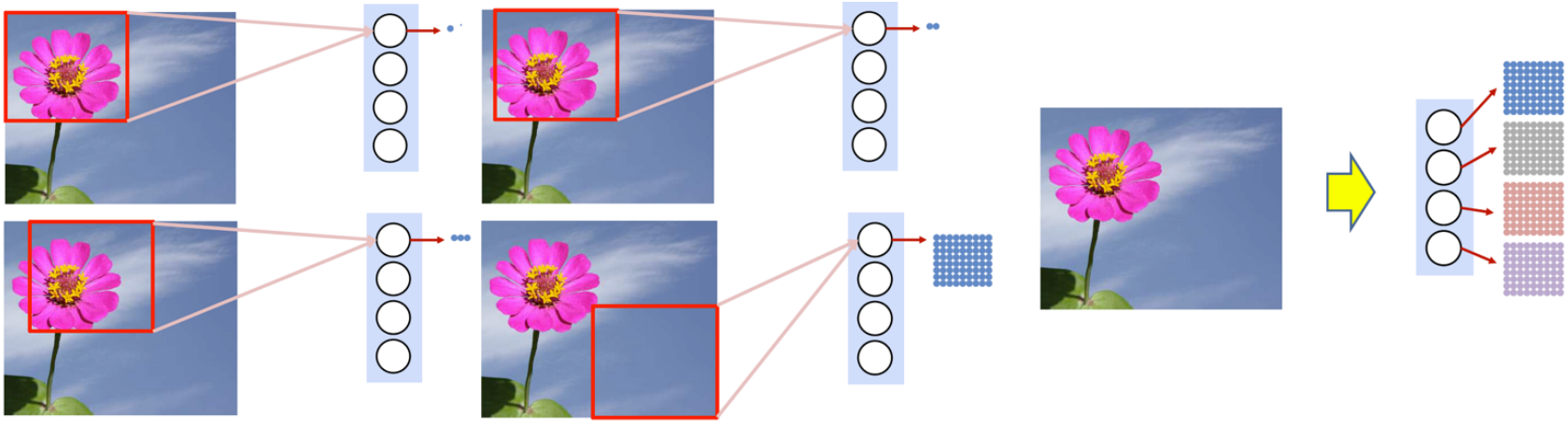
What if, we compute the outcome of one neuron for all scans?

A New Scanning Approach...

- Scan the entire image with one neuron
- Arrange the neuron's outputs from the scanned positions according to their positions in the original image

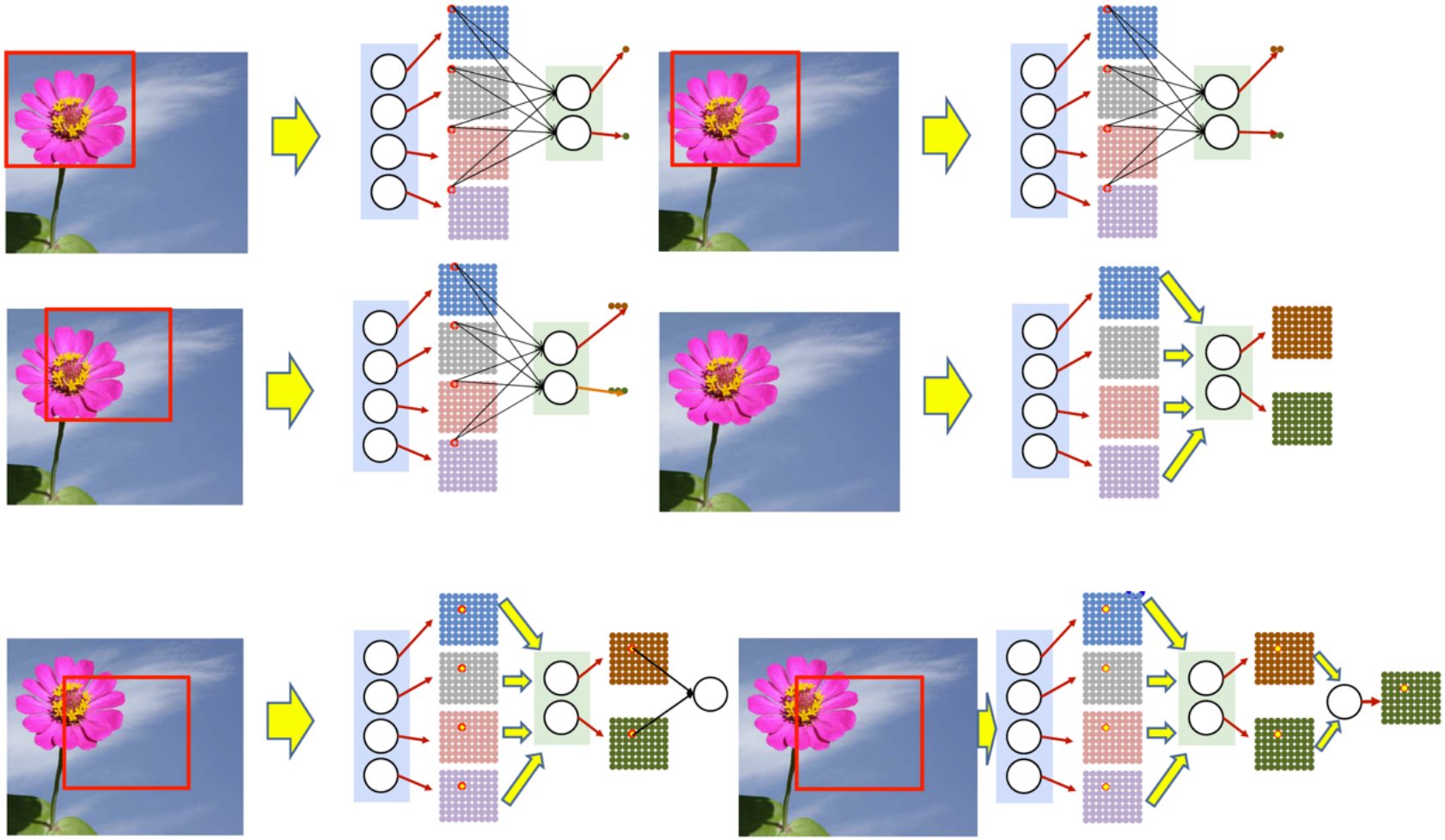


- Recurse the same logic across layers...



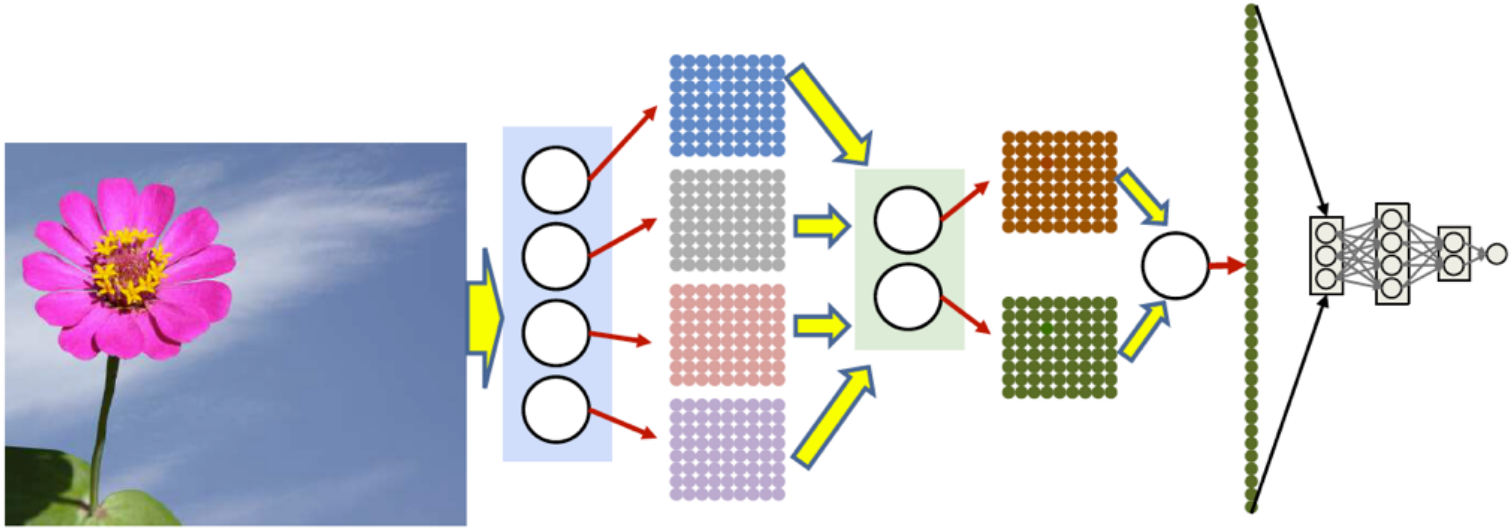
A New Scanning Approach (cont...)

- Subsequent layers jointly scan multiple maps of the previous layer.
- Final layer $\Rightarrow$ A map that detects flower at each location in the image.

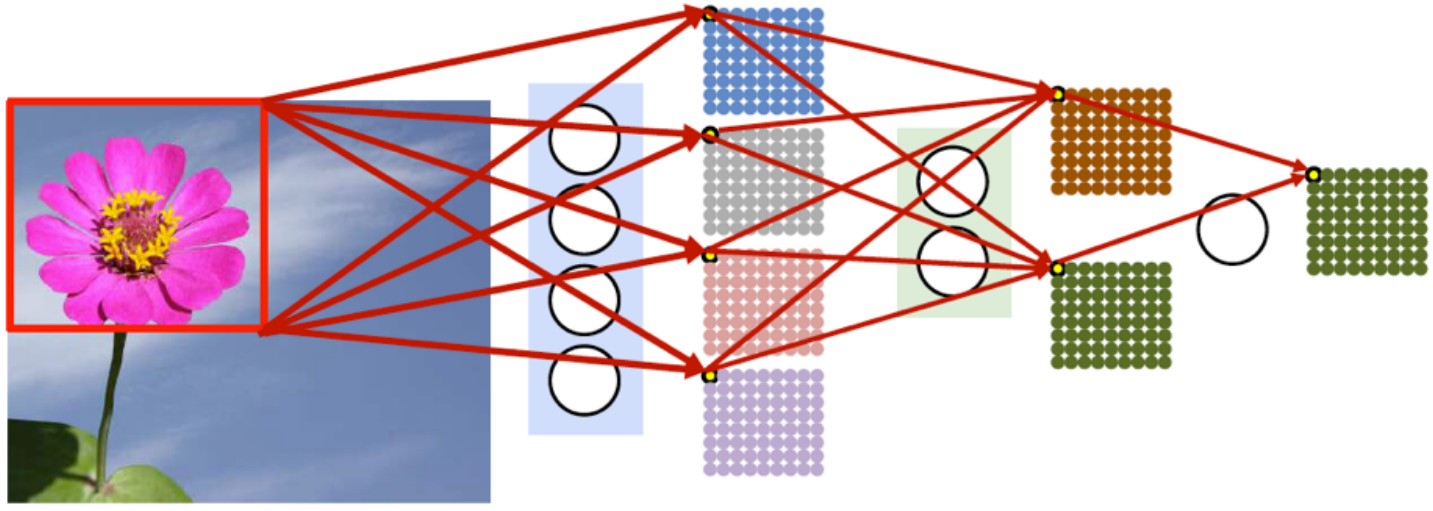


A New Scanning Approach (cont...)

- Goal: Detect the flower, not locate it!
- Flatten the final map and pass it into a softmax layer, or another NN.

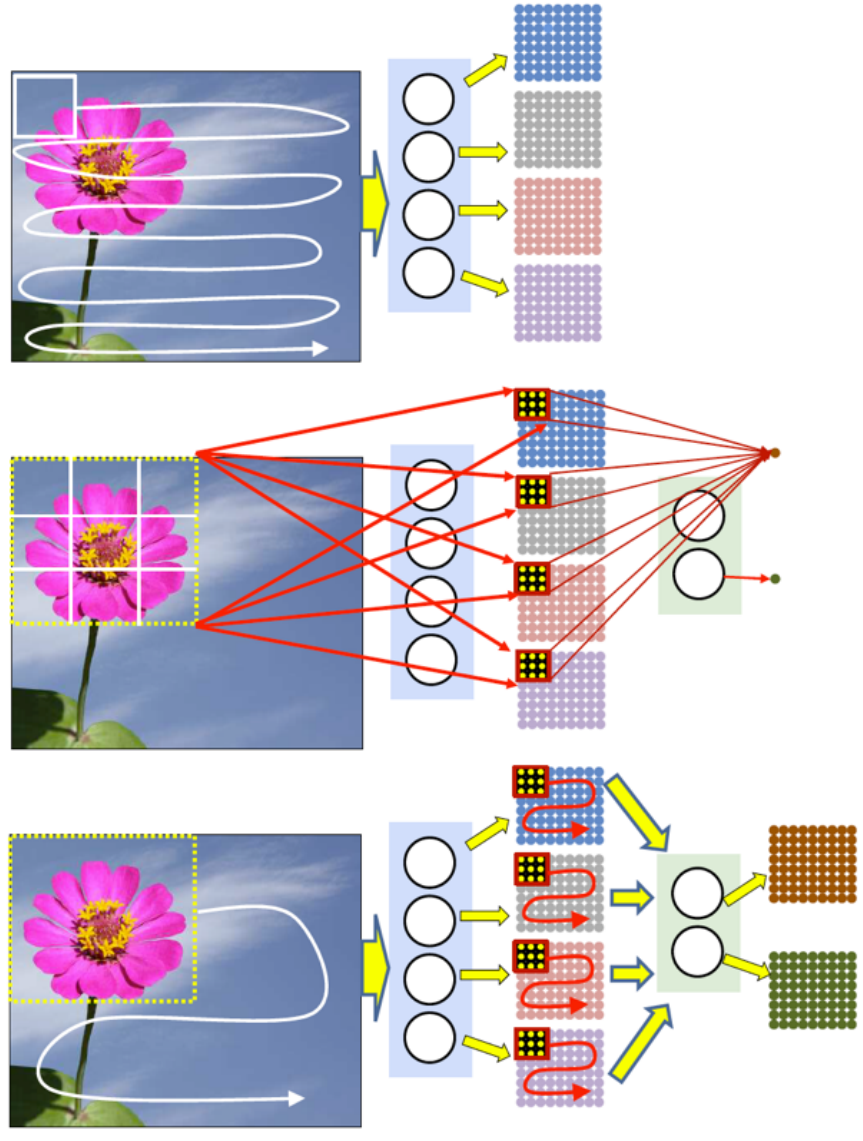


Is this scan sufficient?



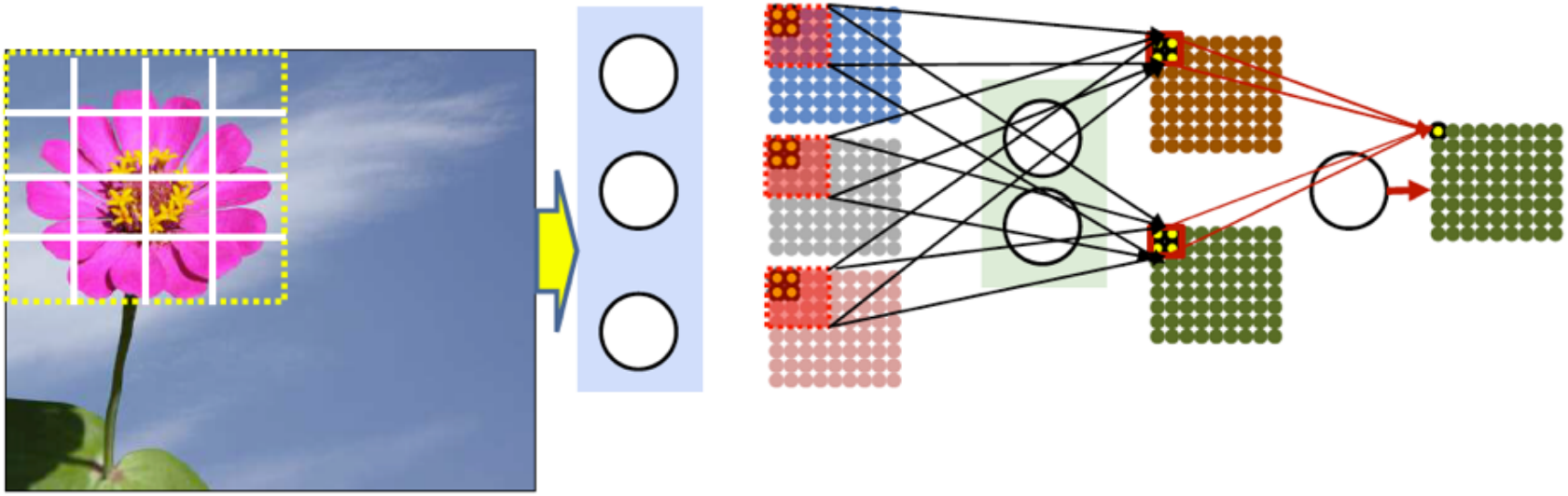
A New Scanning Approach (cont...)

- Distribute the scan $\Rightarrow$ Subsequent layers evaluate blocks of outputs from previous layers
- Now, we can evaluate larger windows
  - Windows distributed over layers $\Rightarrow$ Useful to learn larger patterns
- Jointly scan all first-layer maps $\Rightarrow$ Output of second layer represents a scan of full-sized input window



Convolutional Neural Network (CNNs)

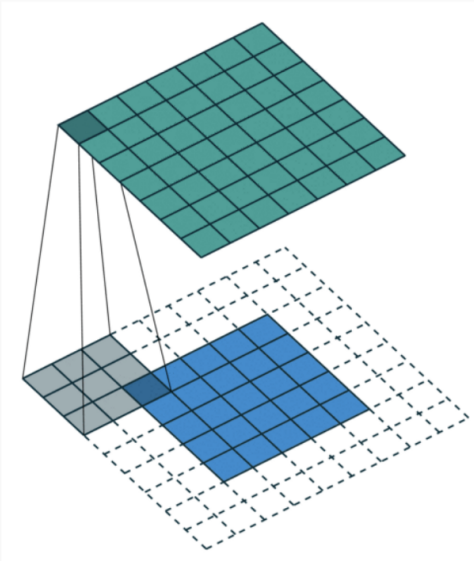
Use the same logic recursively across multiple layers, we obtain a convolutional neural network!



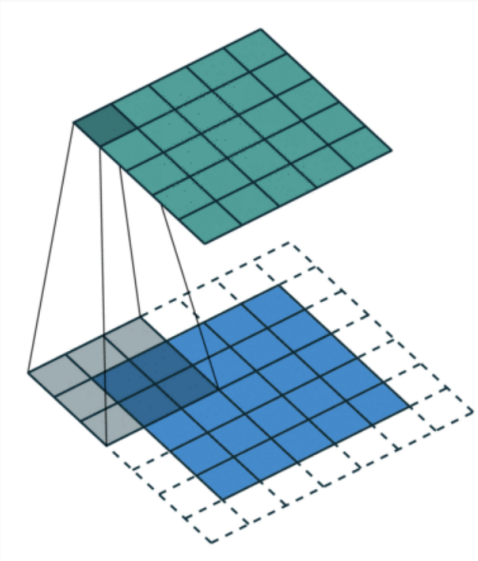
```
def cnn(Img,L,K):
    W = width(Img)
    H = height(Img)
    C = num_channels(Img)
    for l = 1:L
        Compute W_{l-1}, H_{l-1}, K_l, D_l, D_{l-1}
        for x = 1:W_{l-1} − K_l + 1
            for y = 1:H_{l-1} − K_l + 1
                Segment = Y(l-1, 1:D_{l-1}, x:x+K_l-1, y:y+K_l-1)
                for j = 1:D_l
                    Compute z(l,j,x,y) from Segment
                    Y(l,j,x,y) = activation(z(l,j,x,y))
    Y = softmax(Y(L,:,1,1), ···, Y(L,:,W+K-1,H-K+1))
```

Padding

- Sometimes, vital information is present on the corners of previous layers' maps.
- Large kernels will ignore information present in the corners of the activation maps.
- Consequence: Severe drop in accuracy in deeper networks
- **Solution:** Padding
  - *Zero Padding:* Append zeros
  - *Replication Padding:* Replicate the pixel value in the closest neighboring position.
  - *Reflective Padding:* Replicate the pixel value in the position at the same distance from closest neighboring position, but on the opposite side.



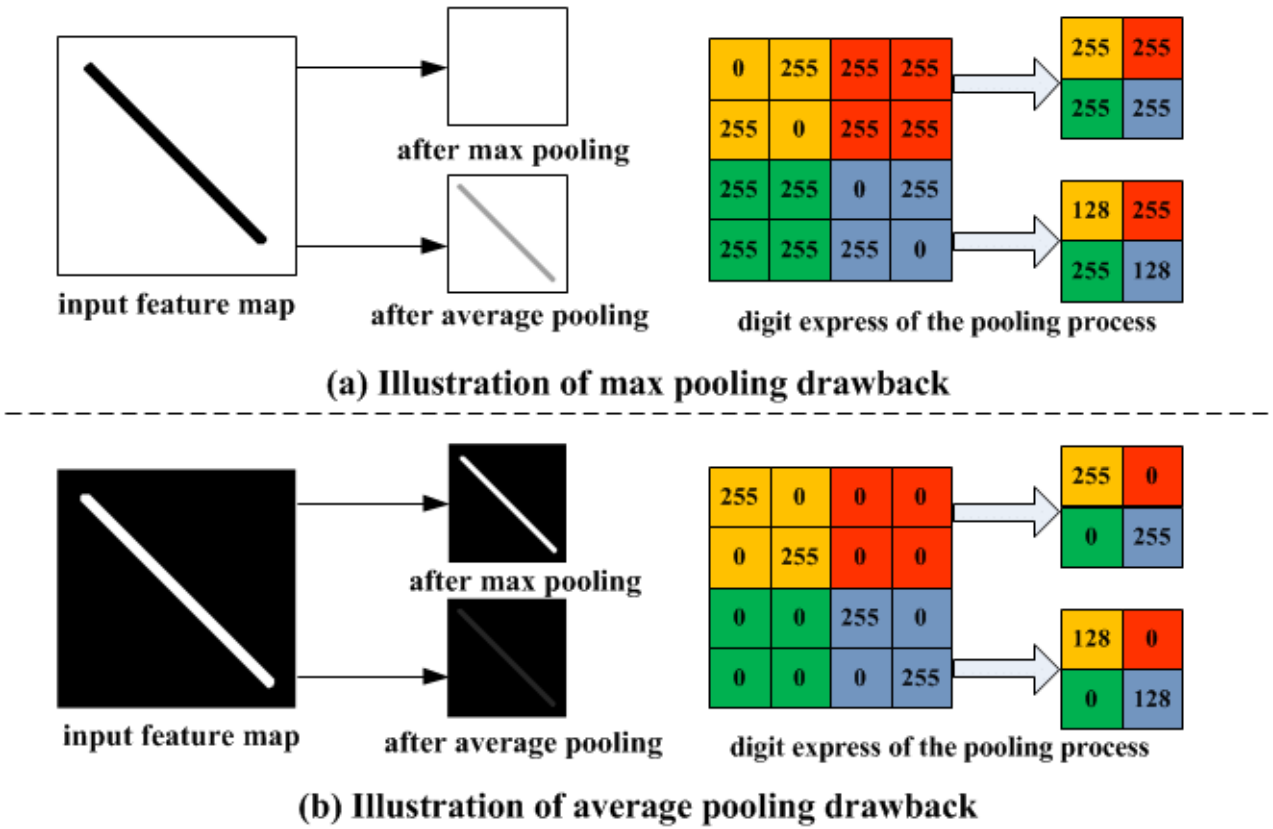
Full padding. Introduces zeros such that all pixels are visited the same amount of times by the filter. Increases size of output.



Same padding. Ensures that the output has the same size as the input.

Robustness to Jitter: Pooling

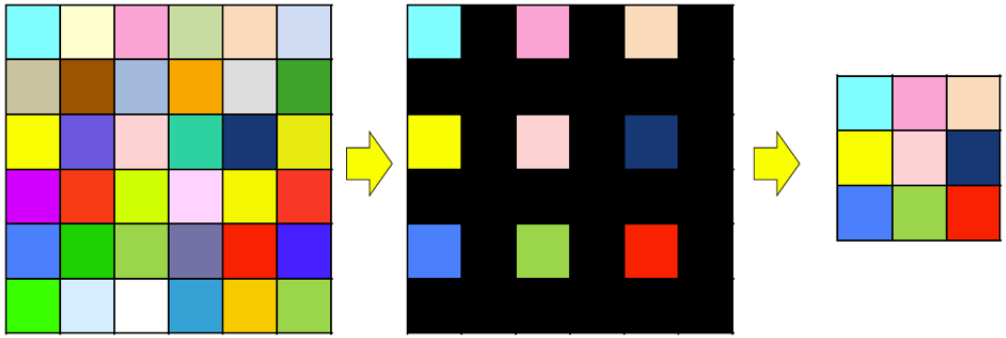
- ***Average Pooling:*** Compute the average of all entries in the kernel
- ***Max Pooling:*** Compute the maximum of all entries in the kernel
- Strong push towards completely ignoring pooling operation[1]



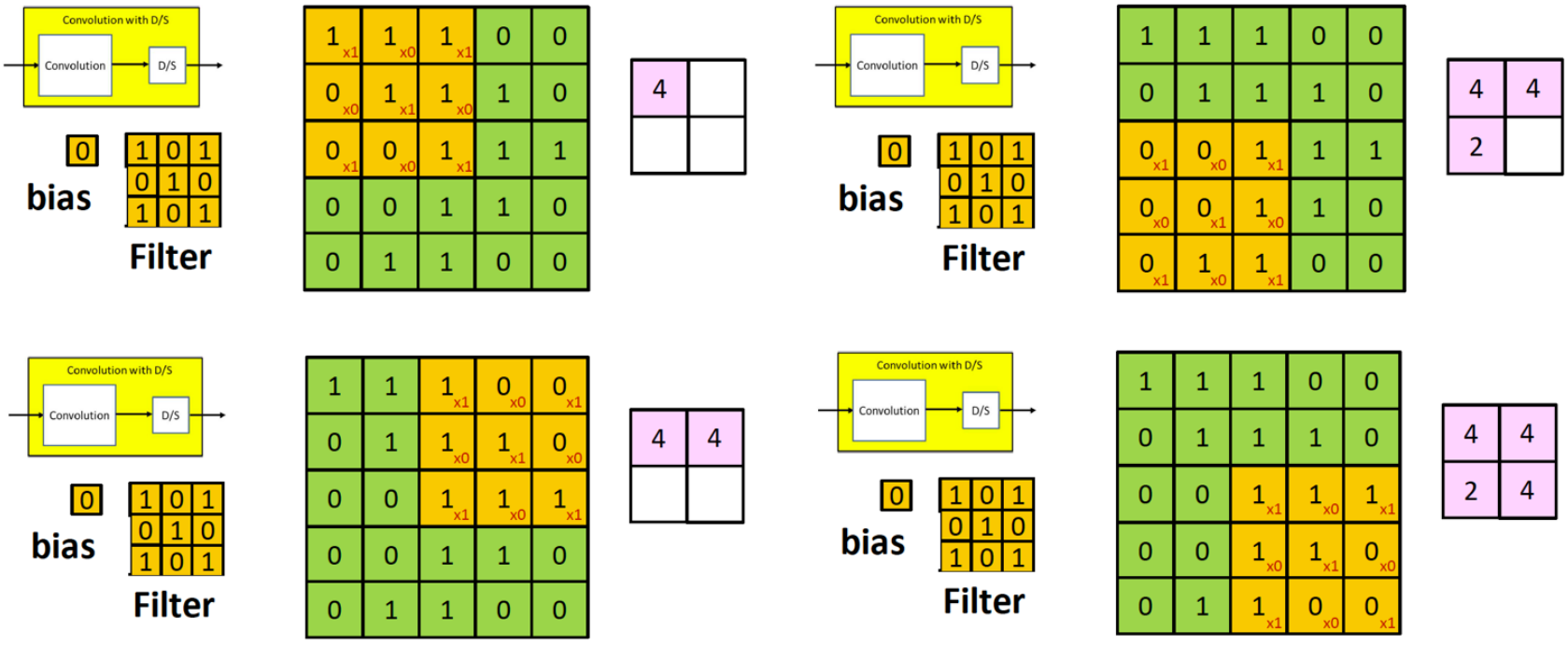
[1] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller, "Striving for Simplicity: The All Convolutional Net," arXiv preprint:1412.6806, 2014.

Need for Downsampling: Striding

- Downsampling: Reduces the size of map by a factor of s in every direction.

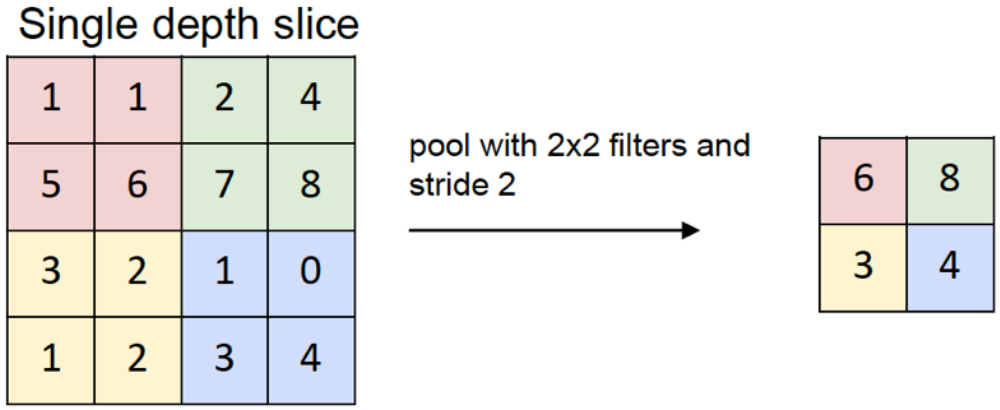


- Convolution + Downsampling $\Rightarrow$ Strides (More efficient method!)

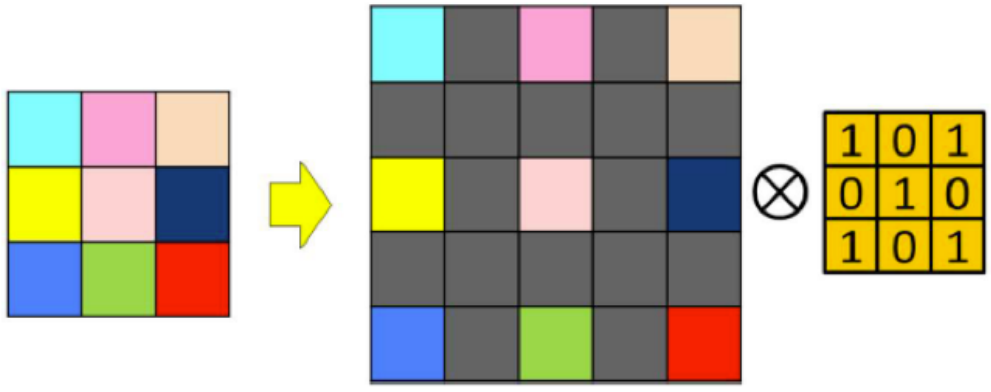


Striding (cont...)

- Pooling + Downsampling $\Rightarrow$ Strides (More efficient method!)

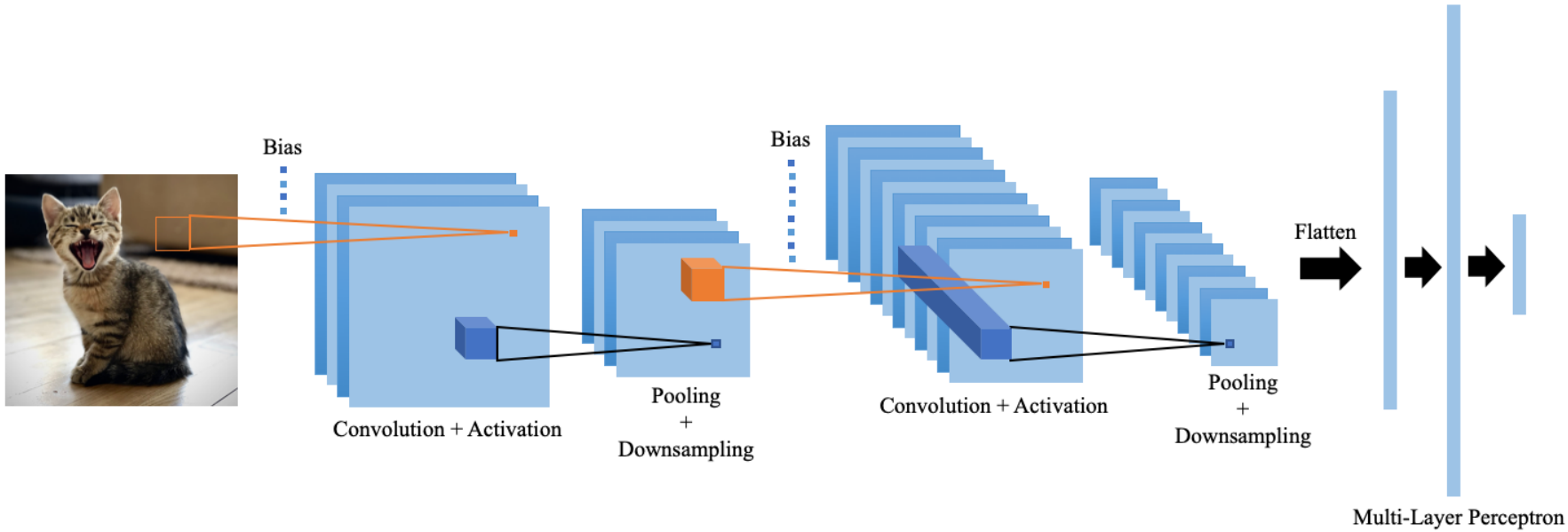


- Upsampling: Increases the size of map by a factor of s in every direction.
  - Typically used after a conv layer
  - Never in conjunction with pooling
  - Not in the final layer...



A General CNN Architecture

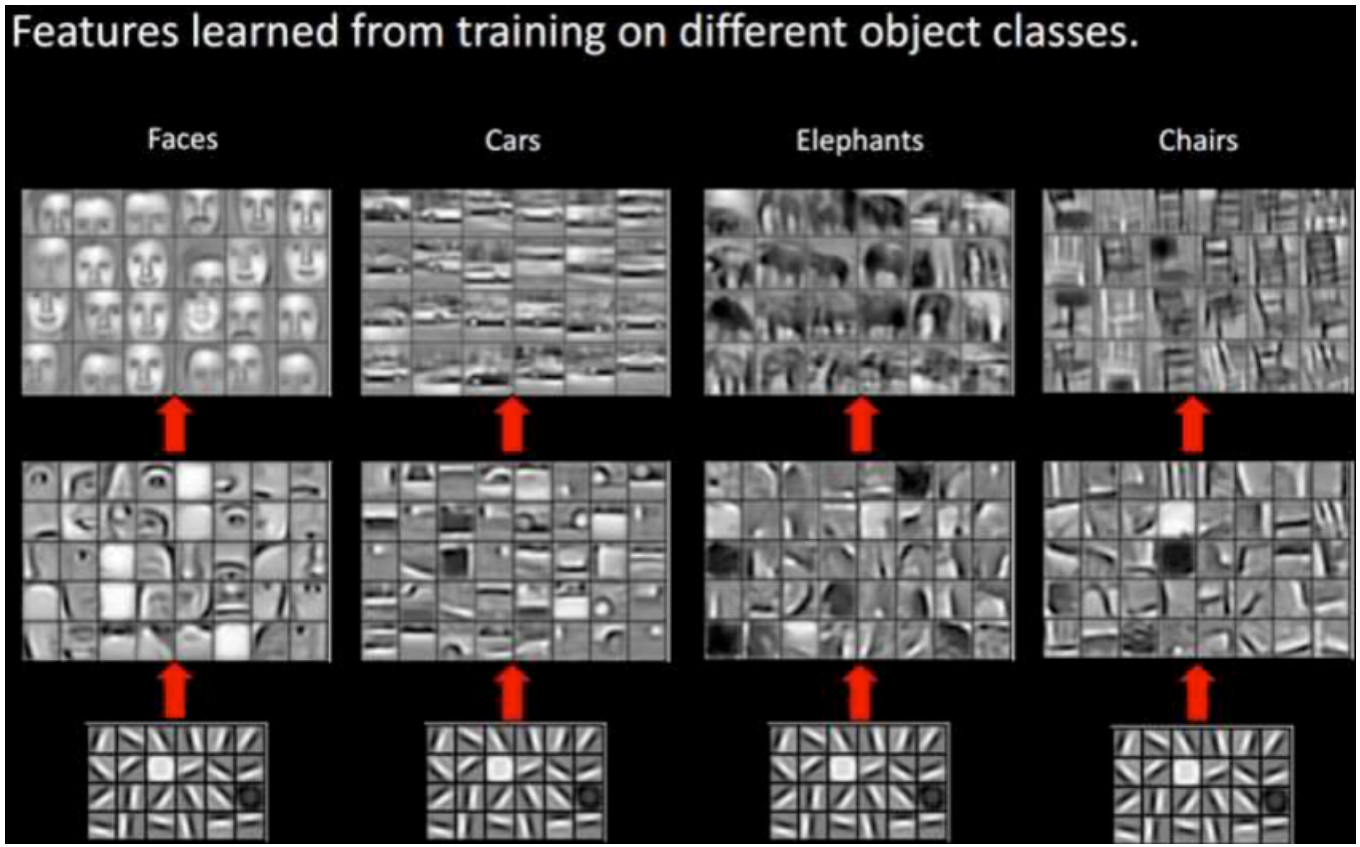
Combining all the feature maps, we obtain...



- **Model parameters updated during training:** Filter weights, bias variables and perceptron weights.
- **Layer Dimensions:** $N \times N$ input + $K \times K$ kernel + s stride = $\left(1 + \left\lfloor \frac{N-K}{s} \right\rfloor\right) \times \left(1 + \left\lfloor \frac{N-K}{s} \right\rfloor\right)$ output – no padding case
- Vectorizing convolution operation demands inner product of *tensors*.
- **Note:** No specific order between convolution layers and downsampling layers.
- For any CONV layer, there is an FC layer that implements the same forward function. The converse is also true. (Ref. https://cs231n.github.io/convolutional-networks/)

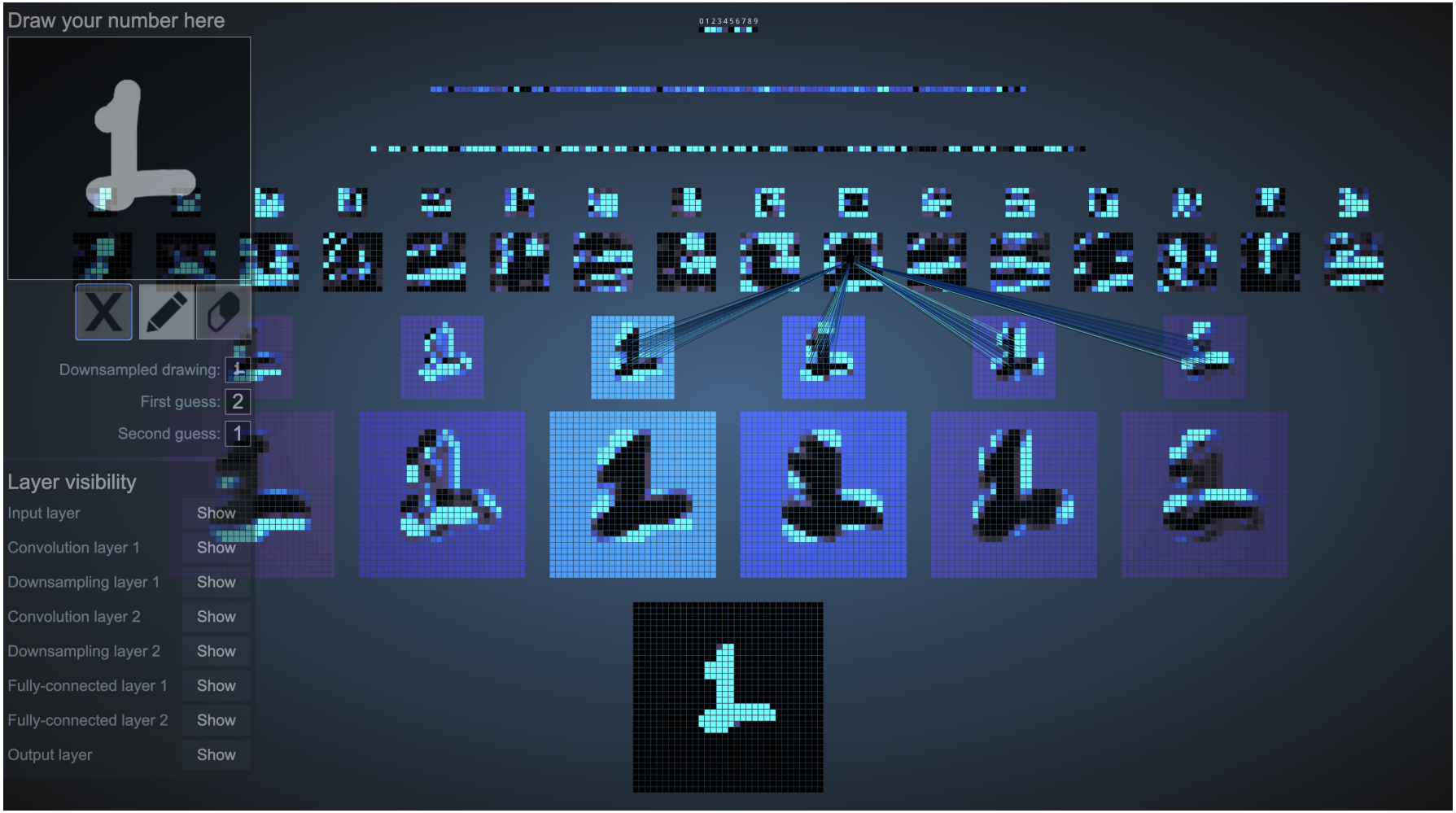
Receptive Fields of CNNs

- Receptive field is the pattern of input image, as seen by each neuron (filter).
- Layer 1: Simple arrangement of weights
- Higher Layers: Non-trivial exercise
  - What patterns in the input does a given neuron respond to?
  - Set the output of neuron to 1 and learn the input via backpropagation.



Illustrative Example: Visualizing Receptive Fields of CNNs[2]

- Layer 1: Convolutional layer (6 filters) + Downsampler
- Layer 2: Convolutional layer (16 filters) + Downsampler
- Layer 3: Flatten + Neural Network + Softmax Output

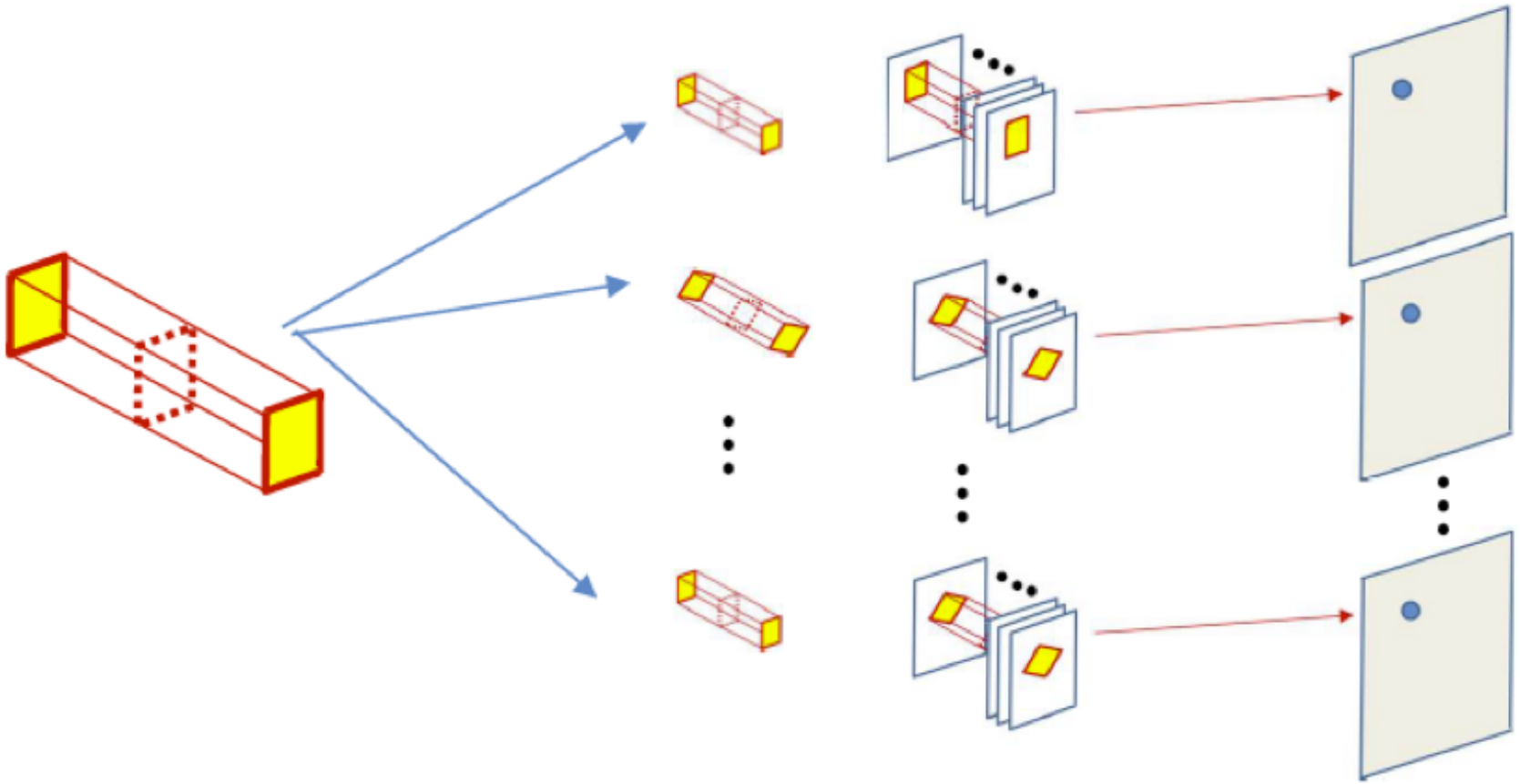


[2] Source: A. W. Harley, "An Interactive Node-Link Visualization of Convolutional Neural Networks," in ISVC, pages 867-877, 2015, Available: http://www.cs.cmu.edu/ aharley/vis/

Beyond Shift Invariance

CNNs are shift-invariant!
But, are they rotation, scale, or reflection invariant?

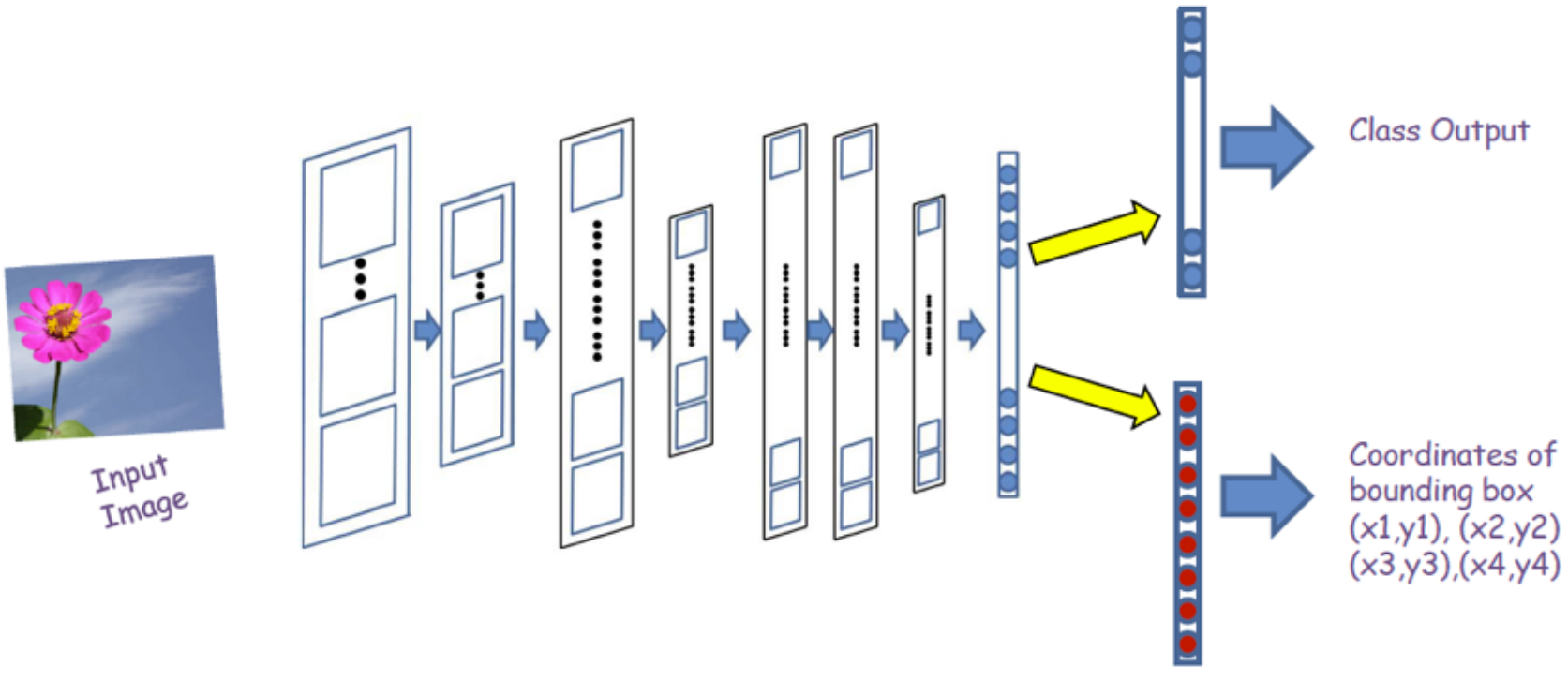
- Transform invariance can be achieved by introducing a finite set of neurons based on transformed kernels.
- Gradients flow back through transformations to update filter weights.



Bounding Boxes

This is a multi-task learning problem – joint detection + localization!

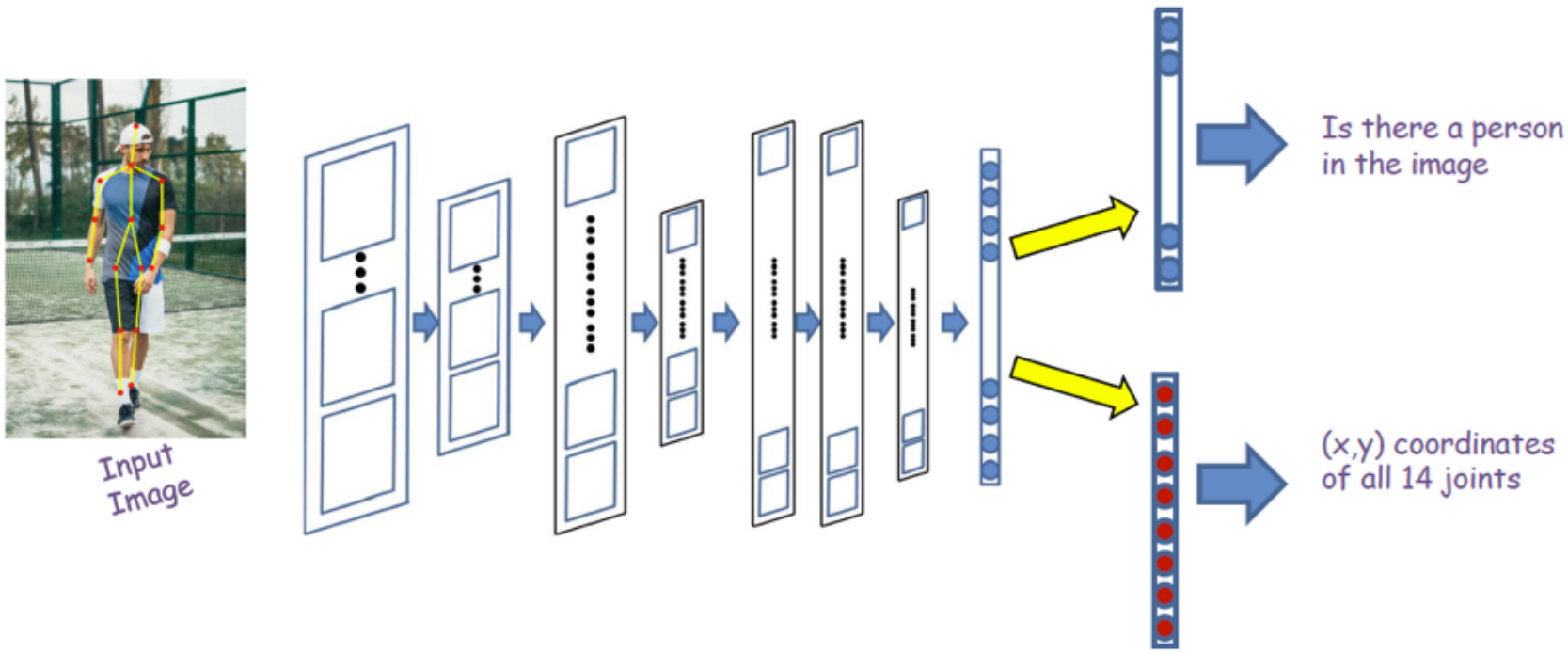
- Same architecture with two separate output layers.
  - One for predicting the class label
  - The other predicts the corners of the bounding box.
  - Loss function: Sum of cross-entropy loss of detector and L2-loss for bounding box predictor.
- Weights used to combine the two losses is a vrey difficult hyper-parameter problem – may need a different success metric for making a good choice!



Pose Estimation

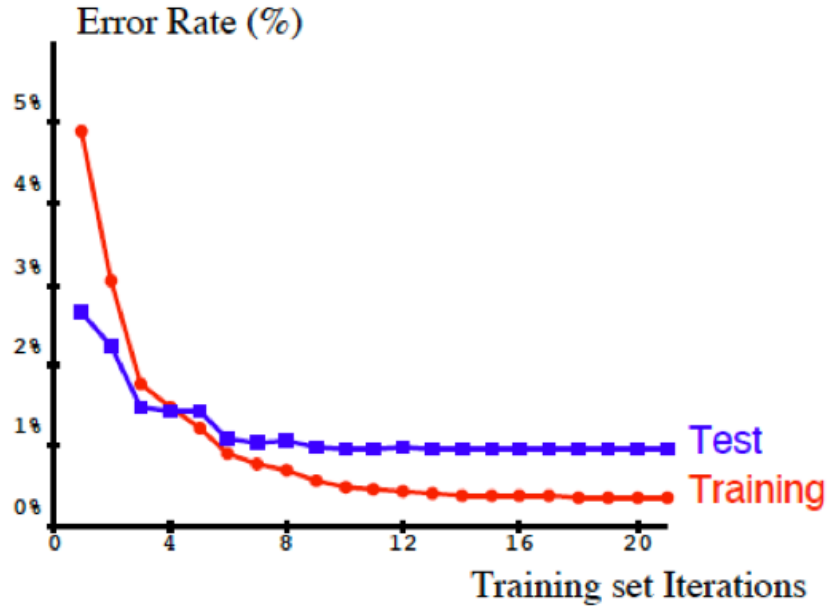
Another multi-task learning problem – joint detection + pose estimation!

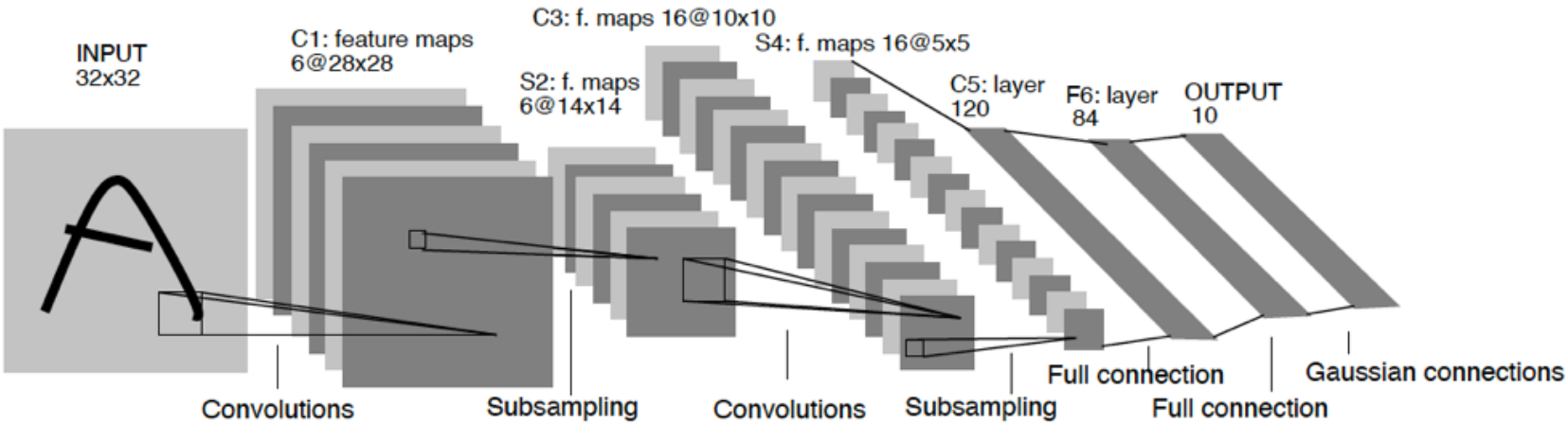
- Same architecture with two separate output layers.
  - One for predicting the class label
  - The other predicts the location of nodes in the stick model.
  - Loss function: Sum of cross-entropy loss of detector and L2-loss for stick-model predictor.
- Same problem with choosing weights for combining losses!



Evolution of CNN Architectures: LeNeT[3]

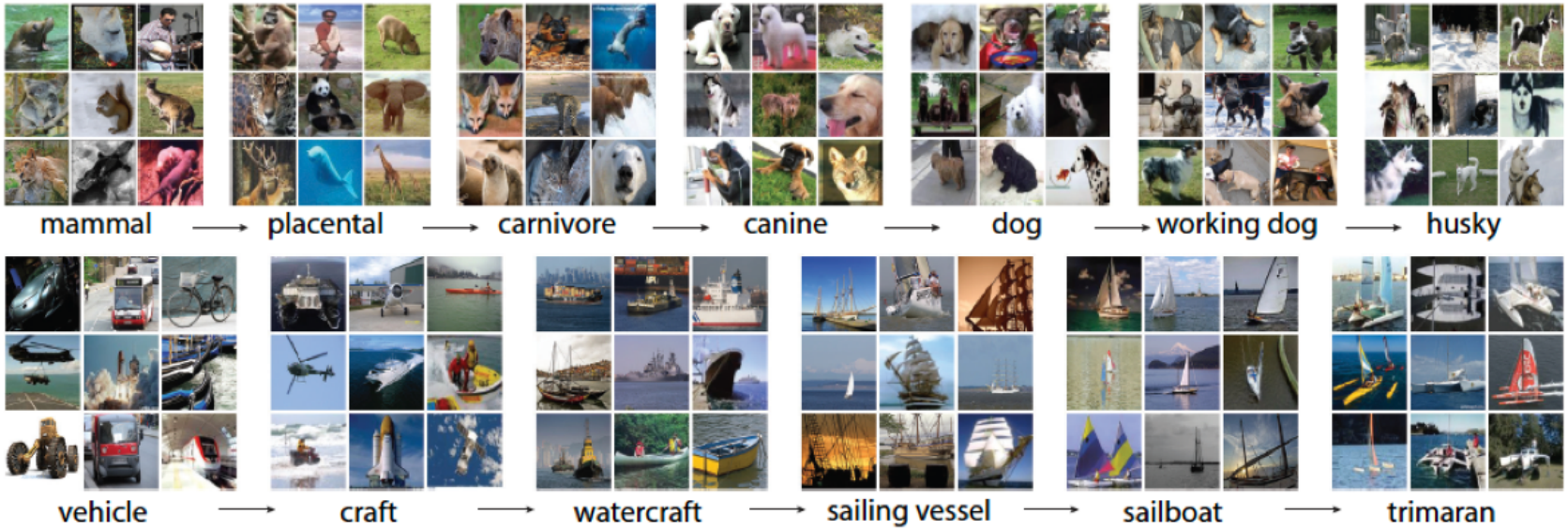
- Yann LeCun's architectures from 1990s
- **LeNet:** Best known in that era.
- 7 layers $\Rightarrow$ 4 conv layers + 3 fully connected (FC) layers
- Error rate: 1% on MNIST data





[3] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, 1998.

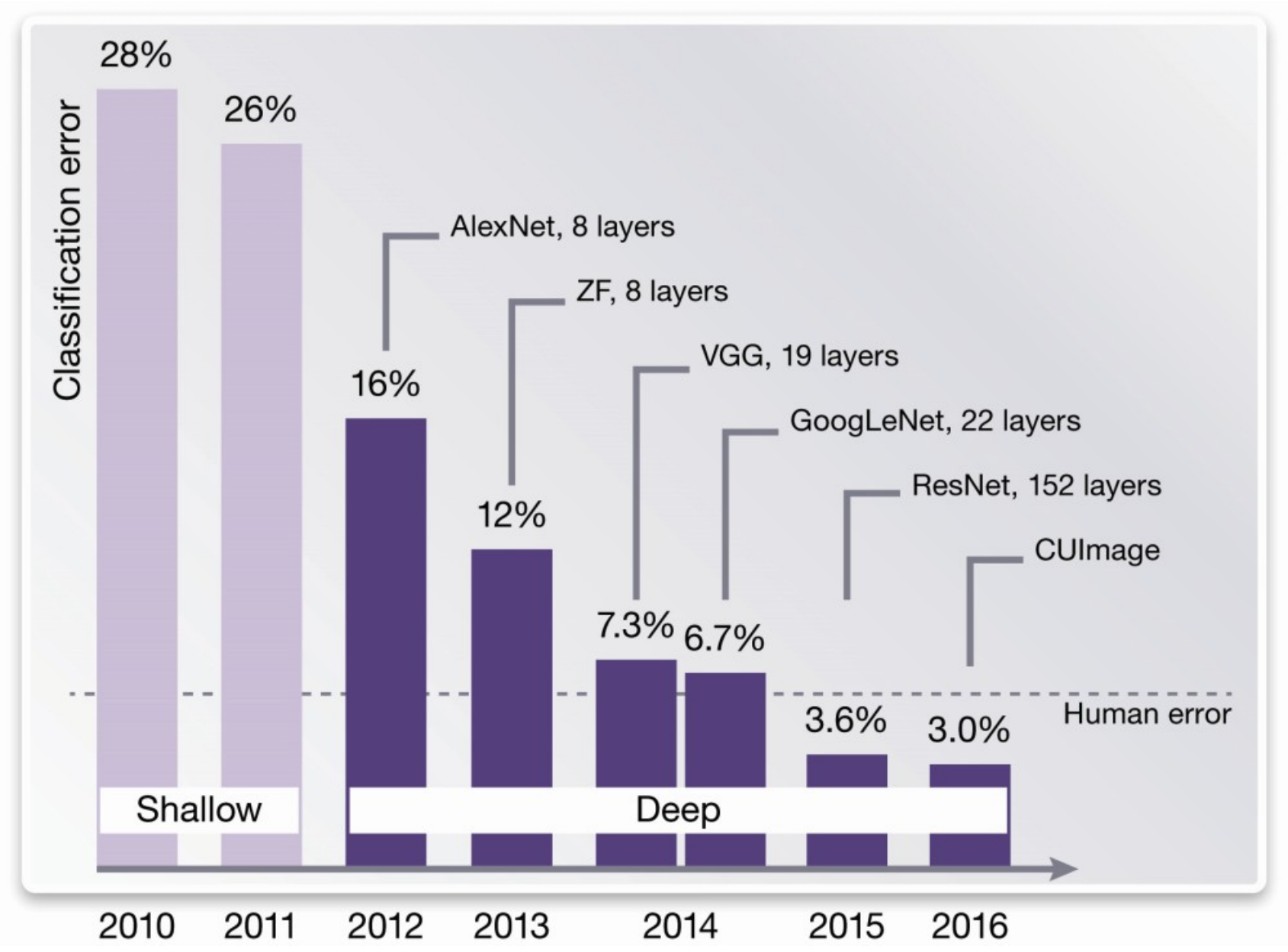
ImageNet[4]



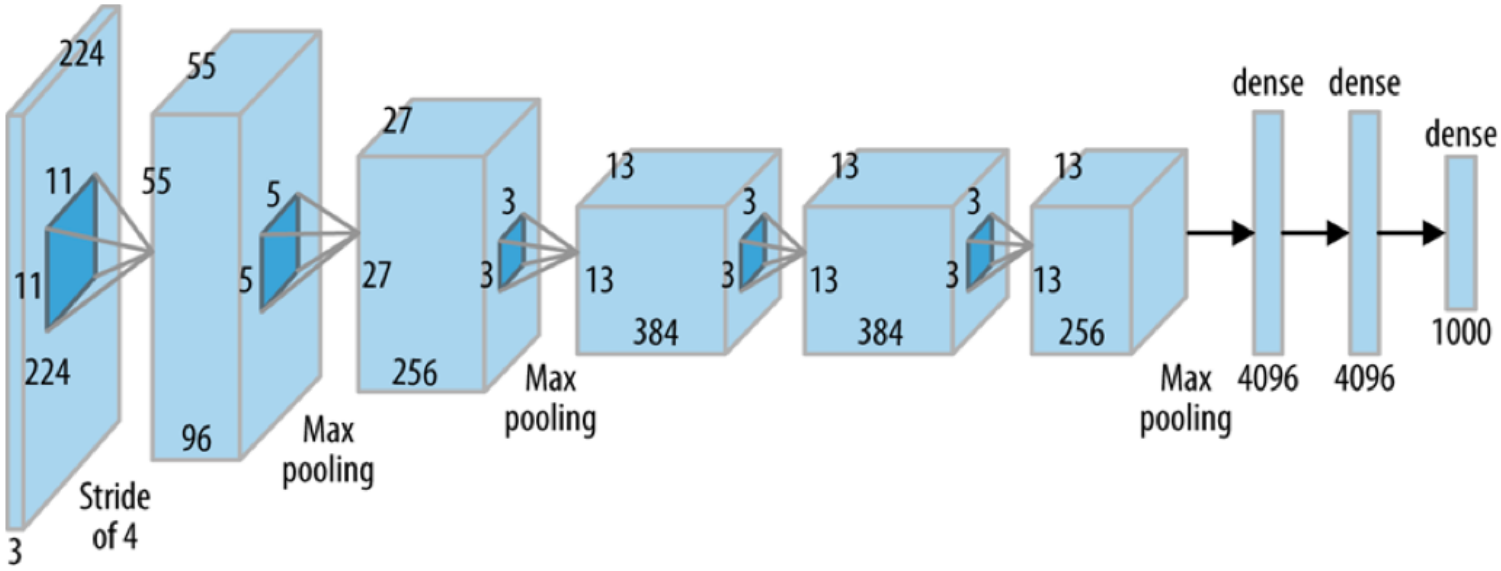
- Images collected from the web (no copyrights on images)
- Labeling performed on Amazon's Mechanical Turk
- Data organized in WordNet hierarchy
- 1.2 million high-resolution images with 1000 classes

[4] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2009.

ImageNet Large Scale Visual Recognition Challenge



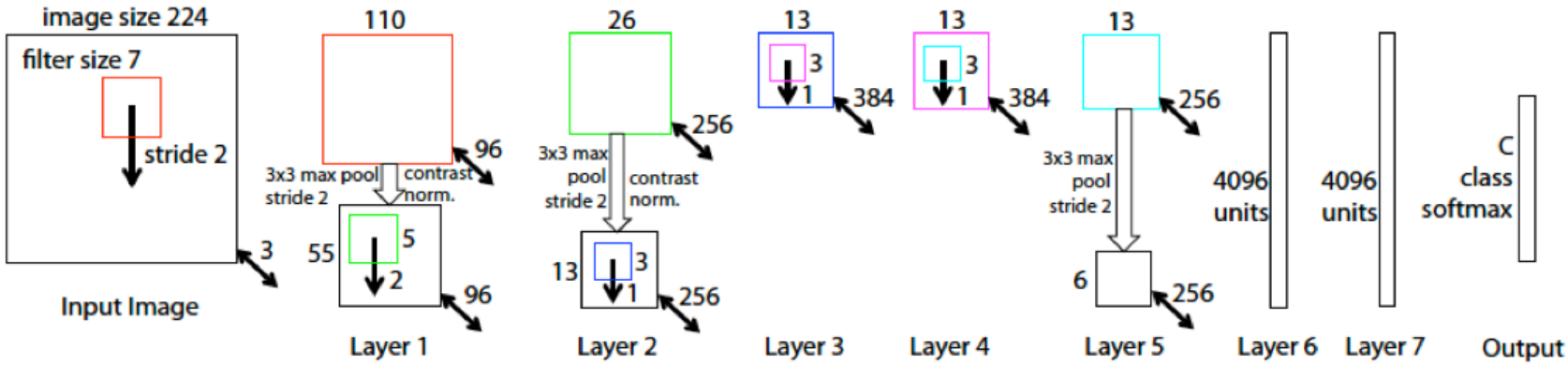
AlexNet[5]



- Proposed by Alex Krizhevsky, Ilya Sutskeve and Geoffrey Hinton in 2012, hence its eponymous title
- **Main Idea:** Deep CNN with model parallelism, dropout and data augmentation
- 9 layers $\Rightarrow$ 6 conv./pool. layers + 3 FC layers
- Kernel size of 11×11 in the first conv. layer
- Model parallelism using two different GPU cards.
- **Error rate:** 12% (25% improvement from AlexNet)

[5] Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "Imagenet Classification with Deep Convolutional Neural Networks." in *Advances in Neural Information Processing Systems*, vol. 25, 2012.

ZFNet[6]

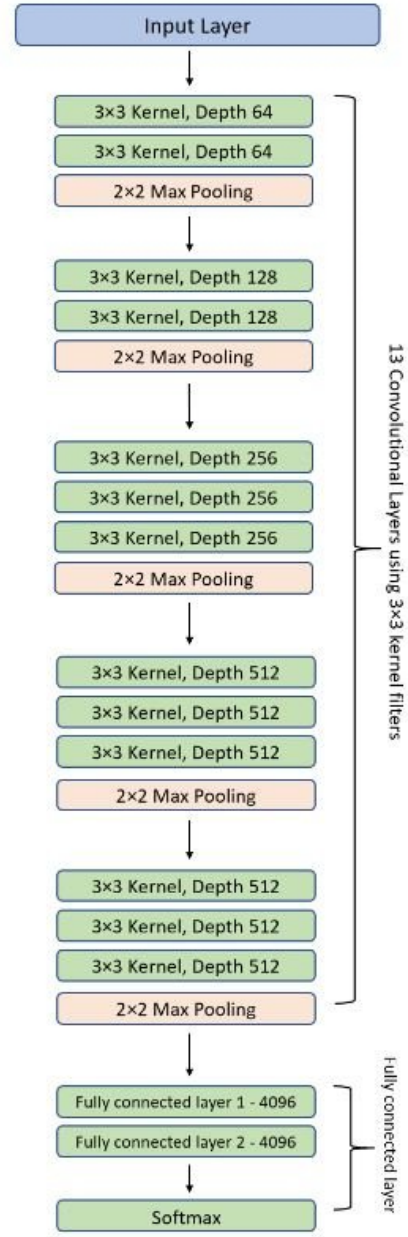


- Proposed by Zeiler and Fergus in 2013, hence their eponymous title
- **Main Idea:** Used receptive fields to improve AlexNet's architecture
  - Issues found in receptive fields of Layers 1 and 2.
- 9 layers $\Rightarrow$ 6 conv./pool. layers + 3 FC layers (similar to AlexNet)
- First layer: Kernel size reduced to 7×7, Stride reduced to 2
- ReLU activations + Minibatch SGD
- **Error rate:** 12% (25% improvement from AlexNet)

[6] Matthew D. Zeiler, and Rob Fergus, "Visualizing and Understanding Convolutional Networks," In *European Conference on Computer Vision*, pp. 818-833, Springer, Cham, 2014.

VGG[7]

- Proposed by Visual Geometry Group (VGG) at Univ. Oxford, hence their eponymous title
- **Main Idea:** Use of identical blocks repeatedly across the network.
  - Each block consists of a sequence of convolutional layers with ReLU activation, followed by a maximum pooling layer for spatial downsampling.
- 16 layers in total
  - 13 convolutional layers + 3 fully-connected (FC) layers
- Kernel size reduced to 3×3 in conv layers
- Kernel size of 2×2 with a stride of 2 in pooling layers
- **Error rate:** 7.3% (39.2% improvement from ZF-Net)

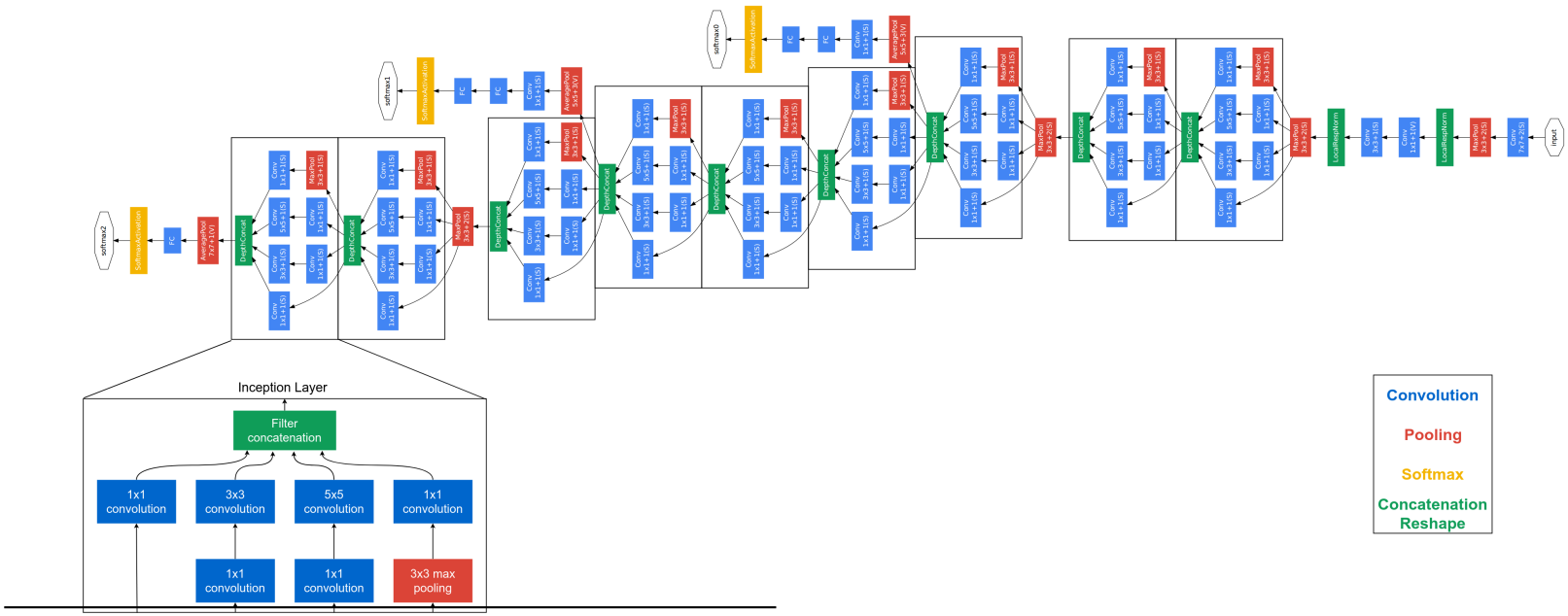


[7] Karen Simonyan Andrew Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *International Conference on Learning Representations (ICLR)*, 2015.

GoogLeNet[8]

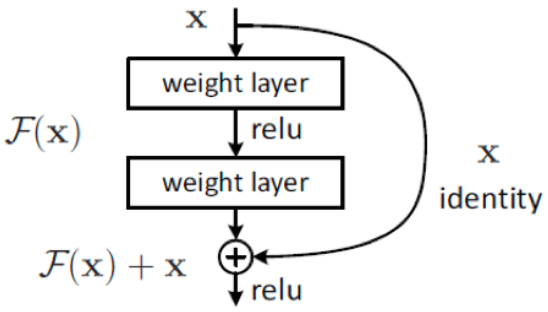
What happens if we have FC layers mixed along with Conv layers in each block?

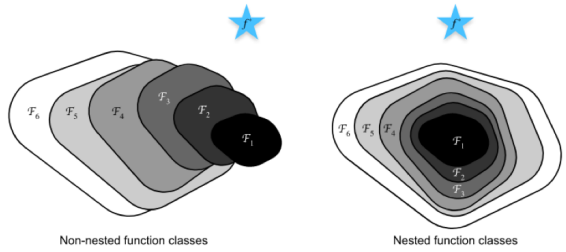
- Most of the authors work for Google Inc., hence its name
- Main Idea: Inception block with both Conv and FC layers (i.e. 1×1 Conv layer)
- 27 layers in total $\Rightarrow$ 9 inception blocks
- Auxiliary classifiers in intermediate layers to mitigate vanishing gradients problem
- Trained on Google's DistBelief platform.
- Error Rate: 6.7% (8.2% improvement from VGG)

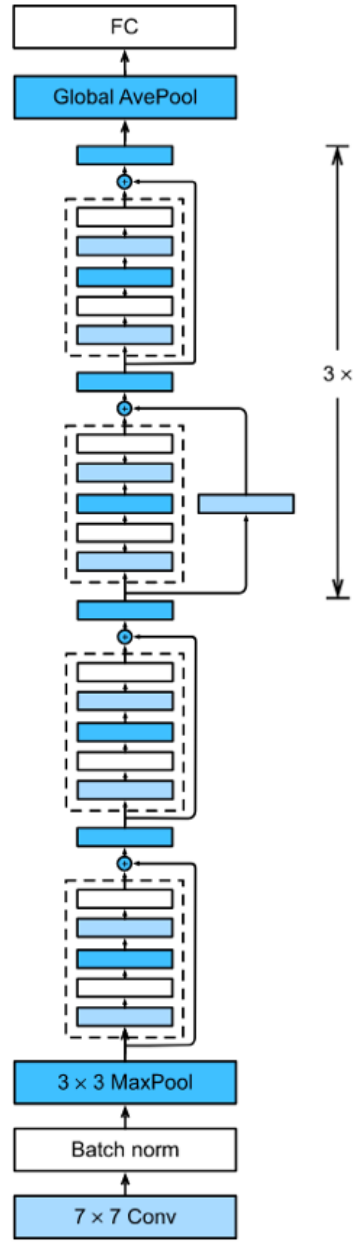


[8] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. "Going deeper with convolutions." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1-9. 2015.

Residual Networks (ResNet)[9]







- Proposed by He *et al.* in CVPR 2016.
- Similar to VGG, but with skip connections
- Skip connections $\Rightarrow$ nested function classes $\Rightarrow$ Fast training
- Two types of skip connections: direct, one with FC layer.
- Enables construction of very deep neural networks, e.g. ResNet-18, ResNet-101, ResNet-152
- ResNet-152 won every competition in 2015
- Error Rate: 3.6% (46.3% improvement over GoogLeNet)

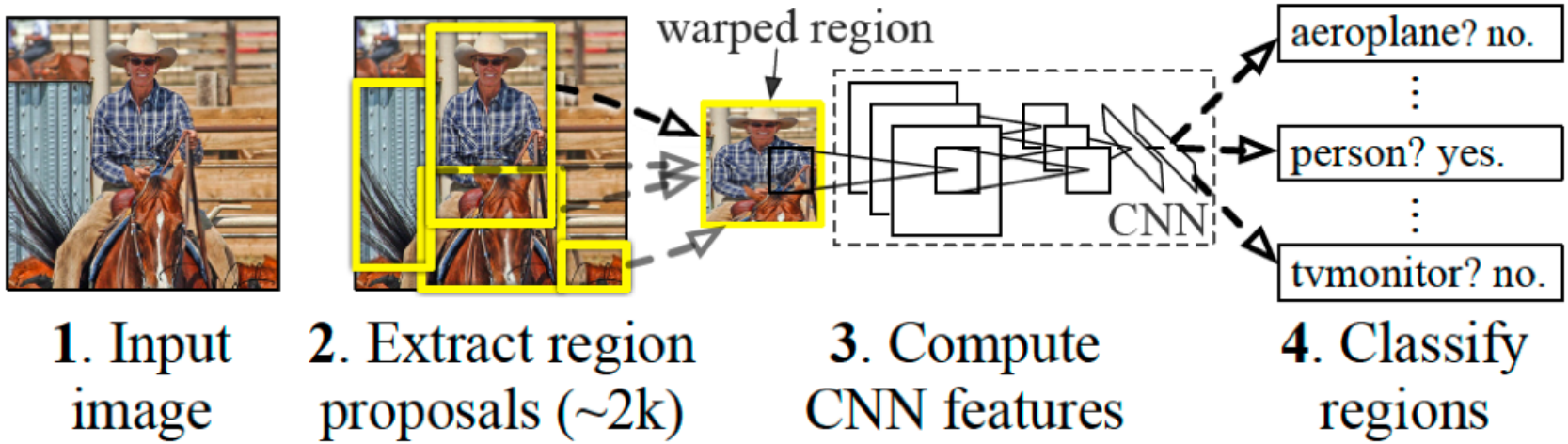
[9] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770-778. 2016.

Object Detection: Joint Classification and Localization

Challenges:

- Single object detection $\Rightarrow (x, y, w, h)$
- Unknown number/size of multiple objects
- Potential obstructions
- Quick forward pass $\Rightarrow$ At least support 30fps for video inputs.

R-CNN[11]: Regions with CNN Features



- Extract $\sim$ 2000 region proposals using selective search[10]
- Reshape each region proposal into a fixed size image (224×224)
- Pass each warped region through AlexNet
- Use SVM to perform classification in each region.
- Reject duplicate regions based on IoU and NMS.
- **Result:** 13s/image on GPU, 53s/image on CPU

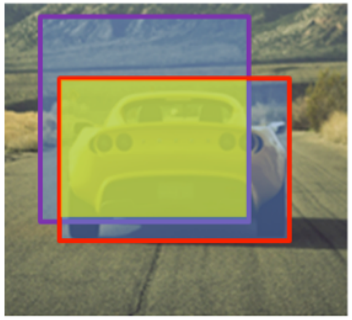
---

[10] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders, "Selective Search for Object Recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp: 154-171, 2013.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 580-587, 2014.

Deduplication of Overlapping Region Proposals

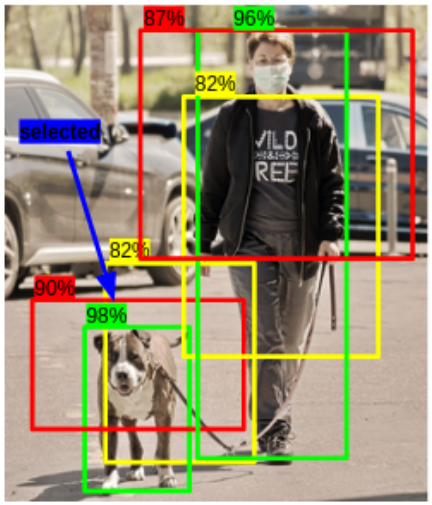
Intersection over Union (IoU): Also called Jaccard's index (or, Jaccard's similarity)



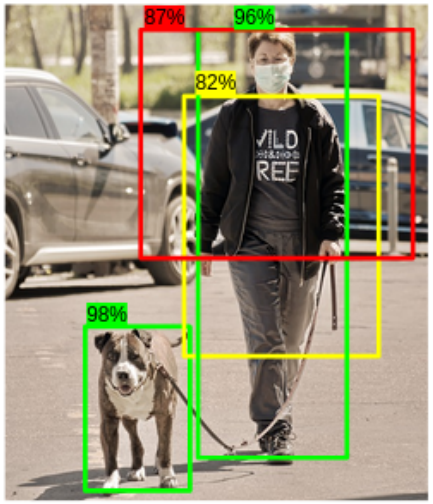
Intersection over union (IoU)

$$= \frac{\text{size of}}{\text{size of}}$$

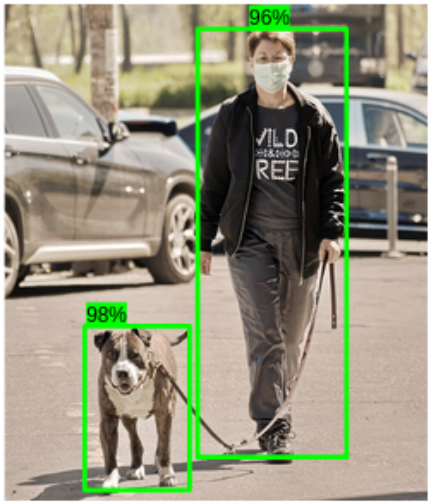
Non-Max Suppression (NMS): Sometimes eliminates "good" BBs when objects are highly overlapping.



Step 1: Selecting Bounding box with highest score



Step 3: Delete Bounding box with high overlap



Step 5: Final Output

Evaluating Object Detectors: Confusion Matrix for a Given Bounding Box

Confusion Matrix based on IoU:

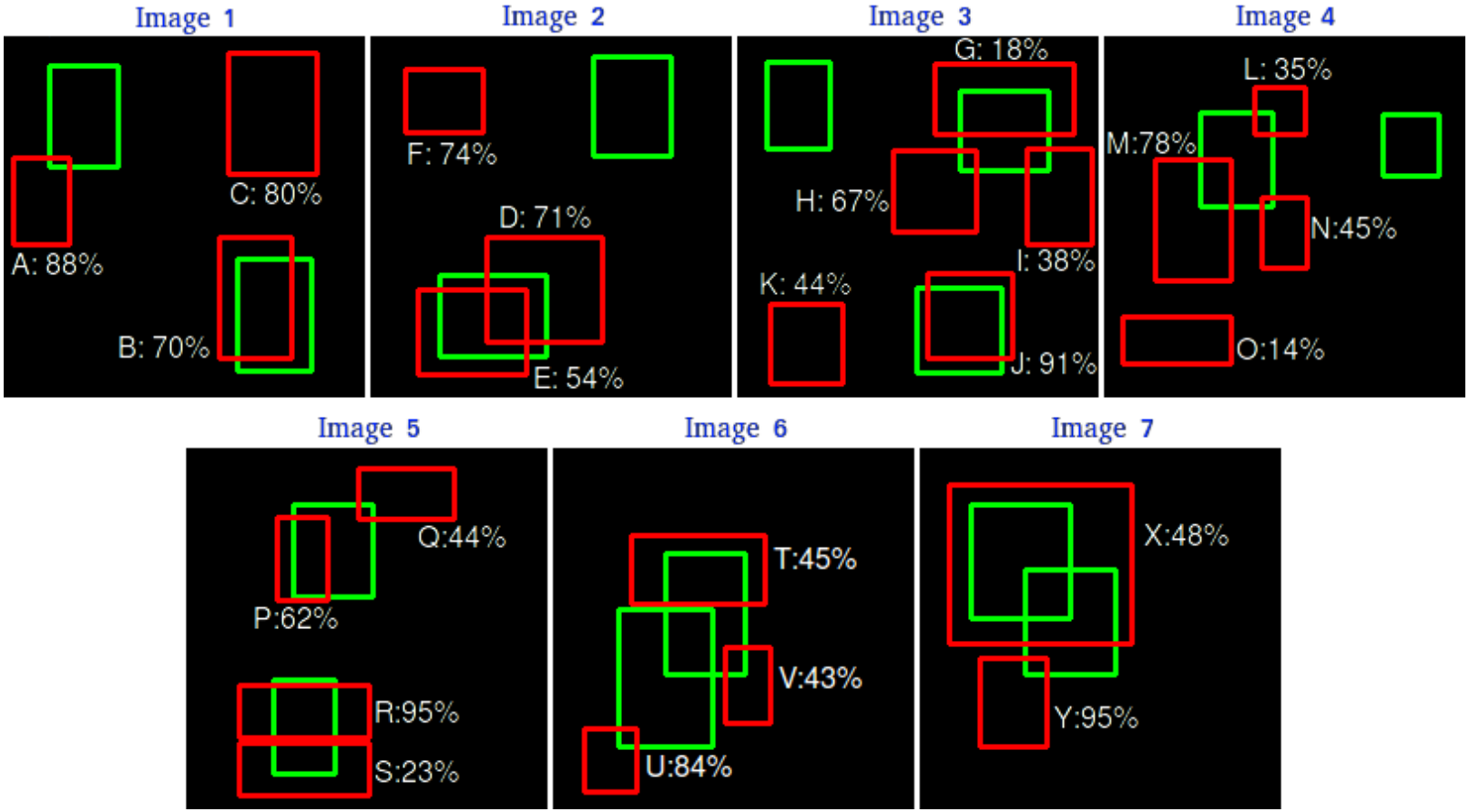
- ***True Positive (TP):*** IoU $\geq$ threshold
- ***False Positive (FP):***
  - No significant overlap: IoU $<$ threshold
  - Bounding boxes overlap, but predicted label does not match with true label $\Rightarrow$ IoU = 0
  - Region proposals with lower confidence scores of same objects are counted under FP.
- ***False Negative (FN):*** Ground truth undetected, although object present in image.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{\#\, predictions}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{\#\, ground\, truths}$$

Evaluating Object Detectors: Precision-Recall Curve[12]

Image Samples from a Dataset:



- 7 images with 15 ground truth objects (green BBs) and 24 detected objects (red BBs, labeled A-Y) of same class.
- Sort all bounding boxes according to their confidence scores

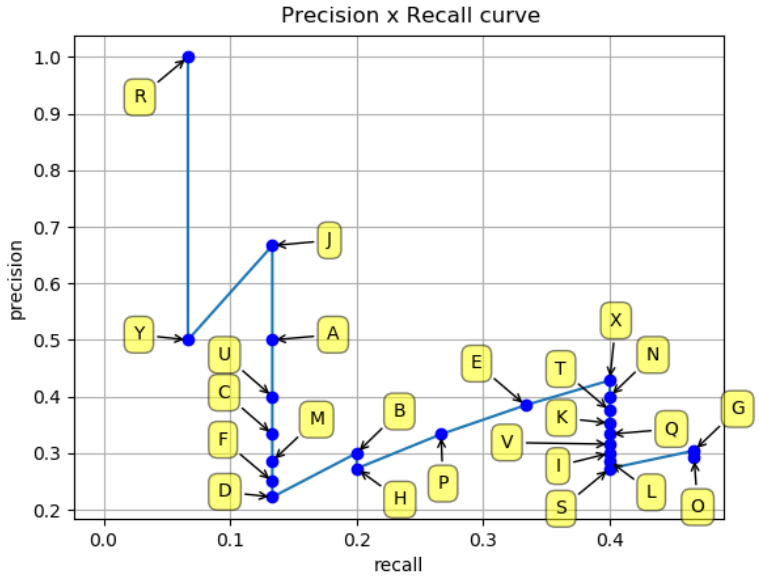
[12] R. Padilla, S. L. Netto, E. A. B. da Silva, "A Survey on Performance Metrics for Object-Detection Algorithms," In *International Conference on Systems, Signals and Image Processing (IWSSIP)*, p. 237–242., 2020.

Evaluating Object Detectors: Precision-Recall Curve (cont...)

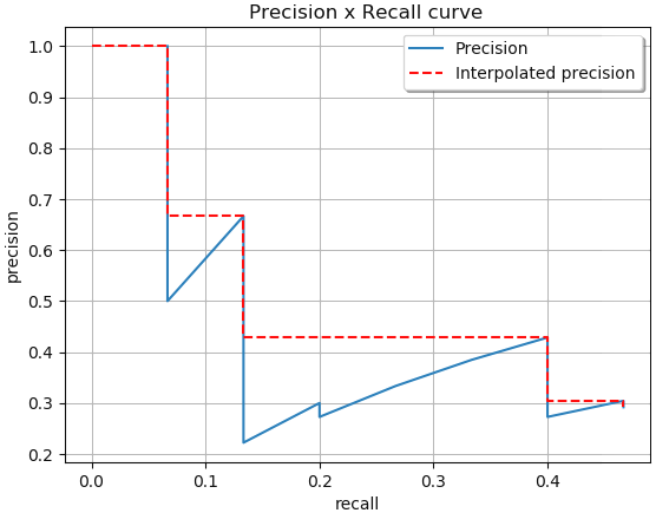
Images	Detections	Confidences	TP	FP	Acc TP	Acc FP	Precision	Recall
Image 5	R	95%	1	0	1	0	1	0.0666
Image 7	Y	95%	0	1	1	1	0.5	0.0666
Image 3	J	91%	1	0	2	1	0.6666	0.1333
Image 1	A	88%	0	1	2	2	0.5	0.1333
Image 6	U	84%	0	1	2	3	0.4	0.1333
Image 1	C	80%	0	1	2	4	0.3333	0.1333
Image 4	M	78%	0	1	2	5	0.2857	0.1333
Image 2	F	74%	0	1	2	6	0.25	0.1333
Image 2	D	71%	0	1	2	7	0.2222	0.1333
Image 1	B	70%	1	0	3	7	0.3	0.2
Image 3	H	67%	0	1	3	8	0.2727	0.2
Image 5	P	62%	1	0	4	8	0.3333	0.2666
Image 2	E	54%	1	0	5	8	0.3846	0.3333
Image 7	X	48%	1	0	6	8	0.4285	0.4
Image 4	N	45%	0	1	6	9	0.4	0.4
Image 6	T	45%	0	1	6	10	0.375	0.4
Image 3	K	44%	0	1	6	11	0.3529	0.4
Image 5	Q	44%	0	1	6	12	0.3333	0.4
Image 6	V	43%	0	1	6	13	0.3157	0.4
Image 3	I	38%	0	1	6	14	0.3	0.4
Image 4	L	35%	0	1	6	15	0.2857	0.4
Image 5	S	23%	0	1	6	16	0.2727	0.4
Image 3	G	18%	1	0	7	16	0.3043	0.4666
Image 4	O	14%	0	1	7	17	0.2916	0.4666

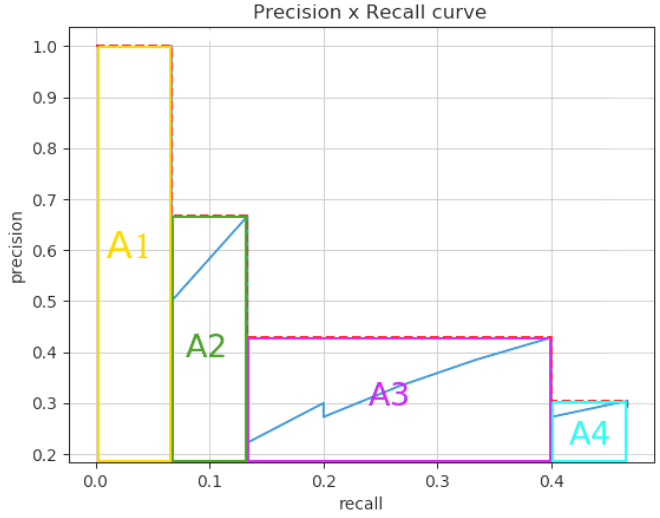
Evaluating Object Detectors: Mean Average Precision (mAP)

Position them on a precision-recall plot:



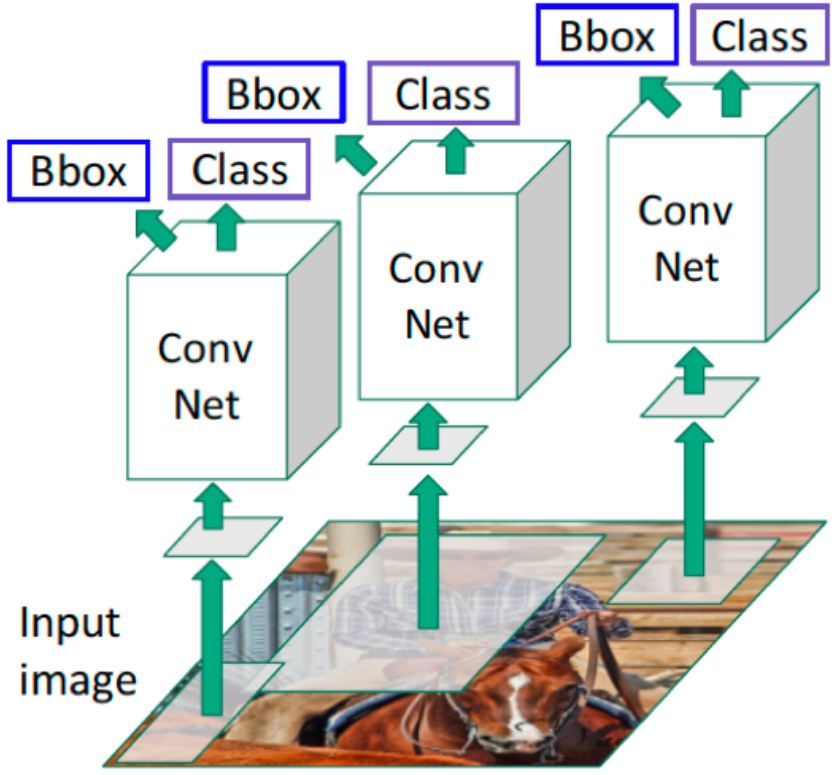
Average Precision: Interpolate the Precision-Recall Curve and find AUC





- **mAP:** Mean of APs across different classes.

R-CNN with Bounding Box Regression

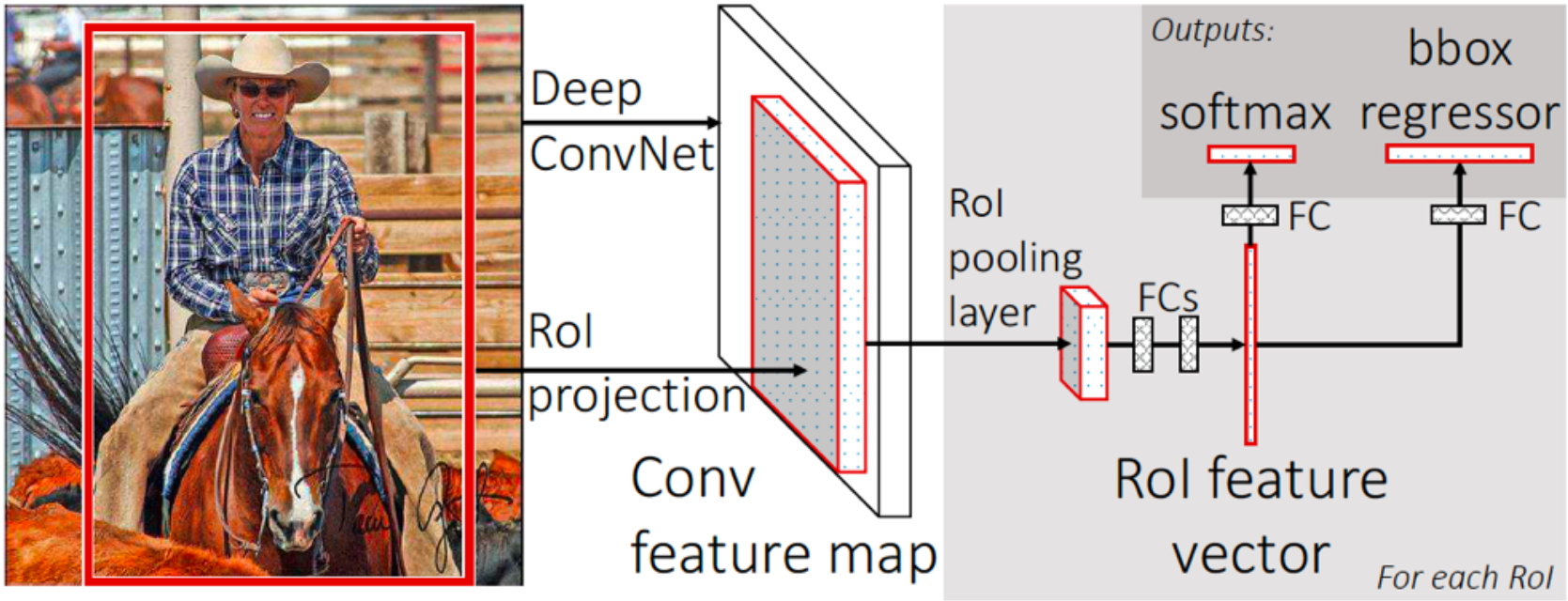


- **Bounding Box Regressor:** Predict a transform to correct the region proposal
- Four numbers: $(\delta_x, \delta_y, \delta_w, \delta_h)$

Concerns:

- $\sim$ 2000 region proposals, 1 forward pass per region $\Rightarrow$ Significant bottleneck!

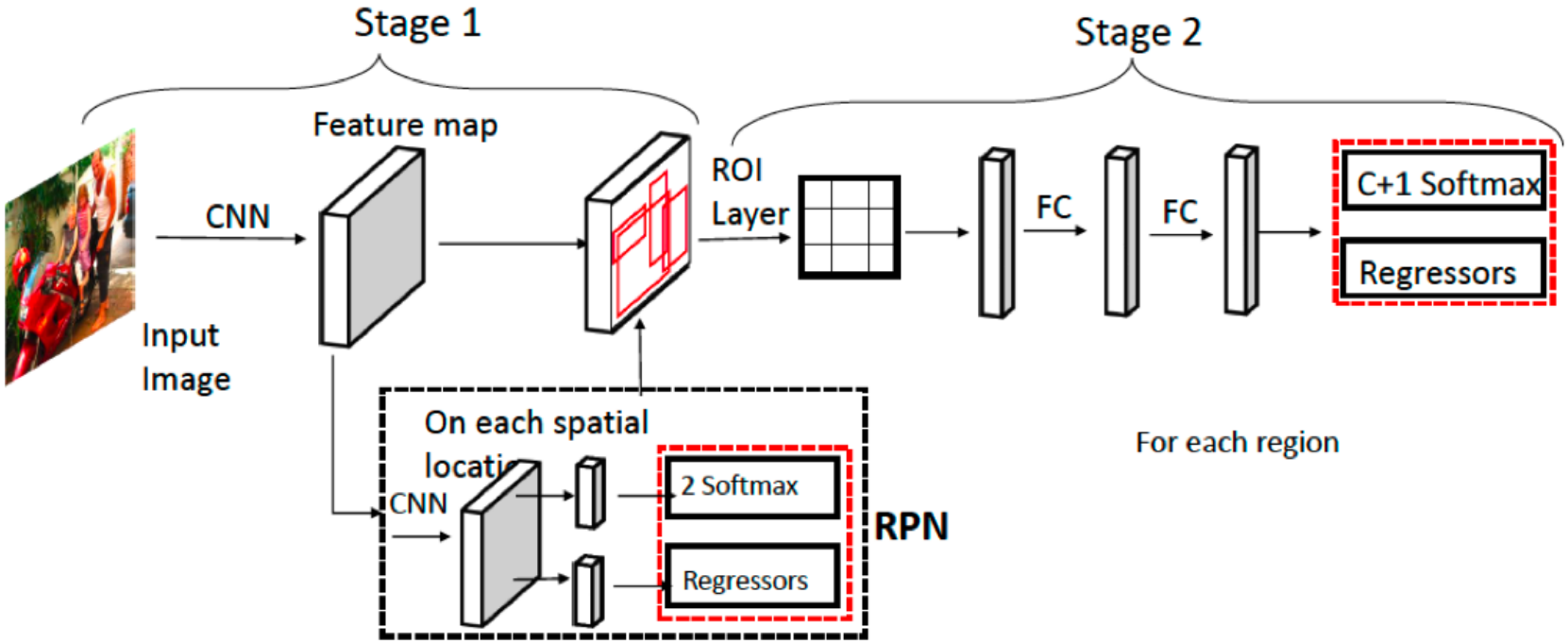
Fast R-CNN[13]



- Run CNN directly on the image once.
- Construct region proposals on CNN features
- Resize/downsample regions of interest (RoI), and pass them through a classifier
- Include BB regressor to fix region proposals
- **Result:** 2.3s/image on CPU (still dominated by selective search)

[13] Ross Girshick, "Fast R-CNN," In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1440-1448, 2015.

Faster R-CNN[14]

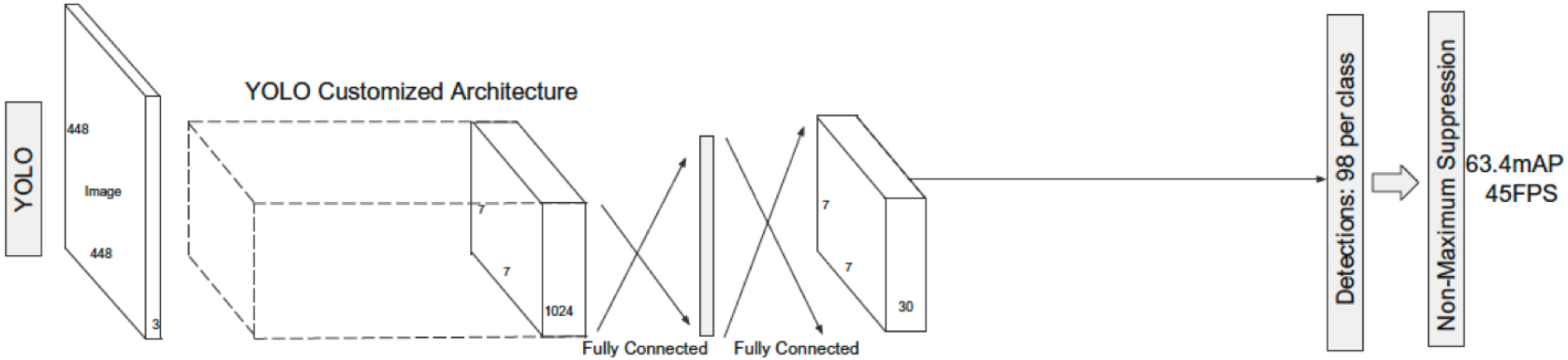


- Compute region proposals using a CNN $\Rightarrow$ Region Proposal Network (RPN)
- Four learning tasks:
  - RPN classification
  - RPN regression
  - Object classification
  - Object regression
- **Result:** 0.2s/image on CPU

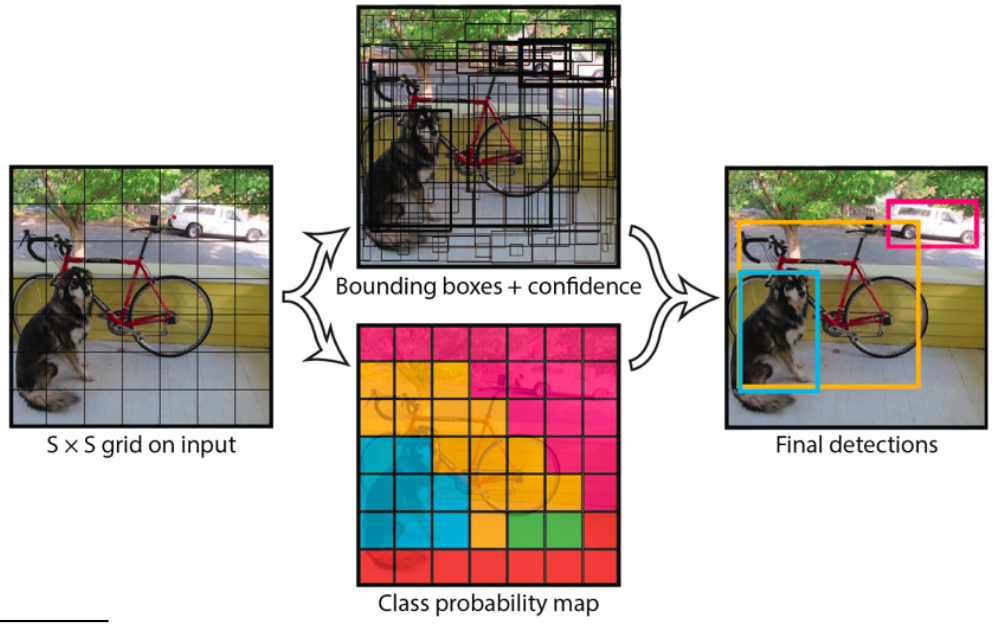
[14] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 28, 2015.

Single-Stage Detectors: YOLO[15] (You Only Look Once)

Do we really need two-stage detectors?

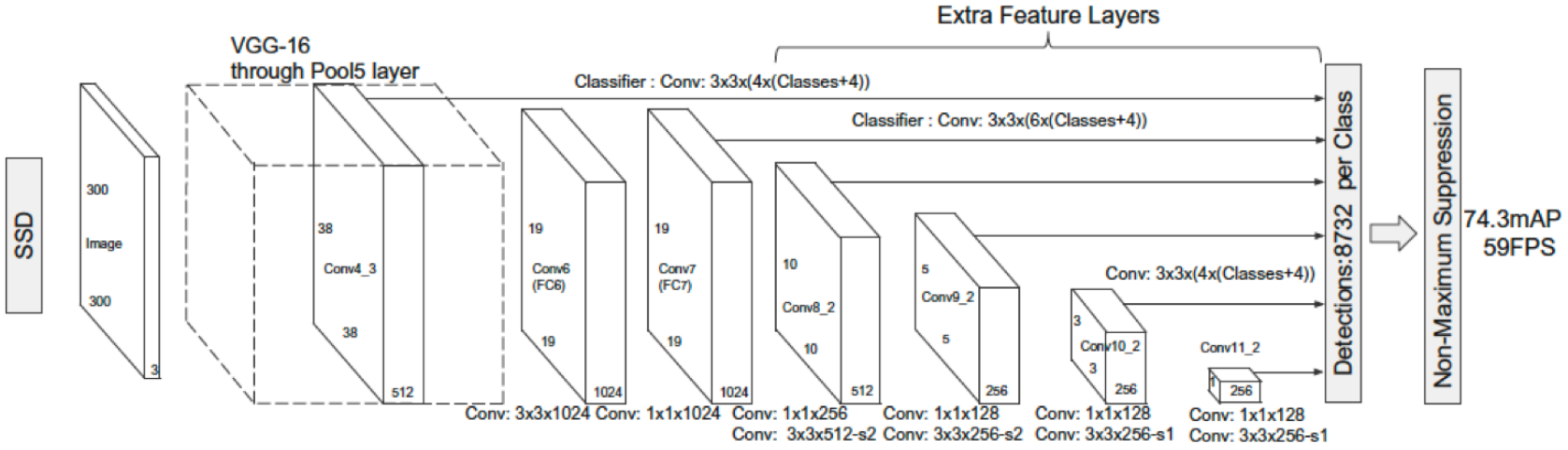


- Reframe object detection as a single regression problem!
- Divide image into $S \times S$ grid
- Each grid cell predicts B BBs, their confidences and C class probabilities
- Architecture inspired from GoogLeNet (inception layers replaced with 1×1 conv + 3×3 conv layers)
- **Result:** 45-150 fps

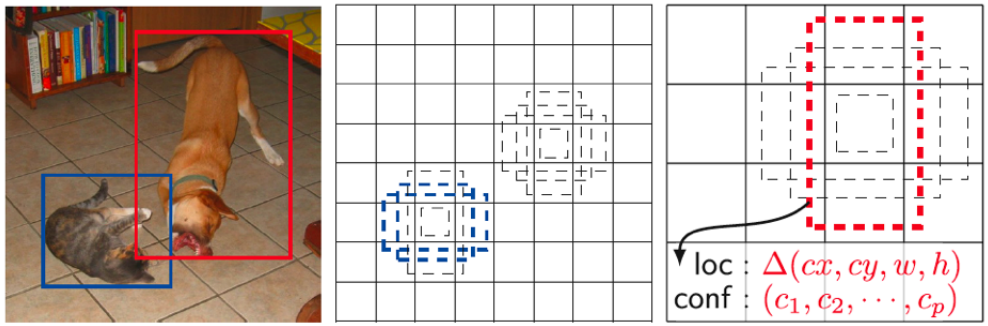


[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 779-788. 2016.

Single-Stage Detectors: SSD[16] (Single-Shot Detector)



- Concern: YOLO detects only two objects in one location
- In each grid cell, set of anchors with multiple scales and aspect ratios (unlike predicting with fixed grid cells as in YOLO)
- VGG-16 as base architecture



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

Result: 59 fps, but accuracy comparable to Faster R-CNN.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "SSD: Single Shot Multibox Detector," In *European Conference on Computer Vision*, pp. 21-37. Springer, Cham, 2016.

Comparison of Object Detectors[17]

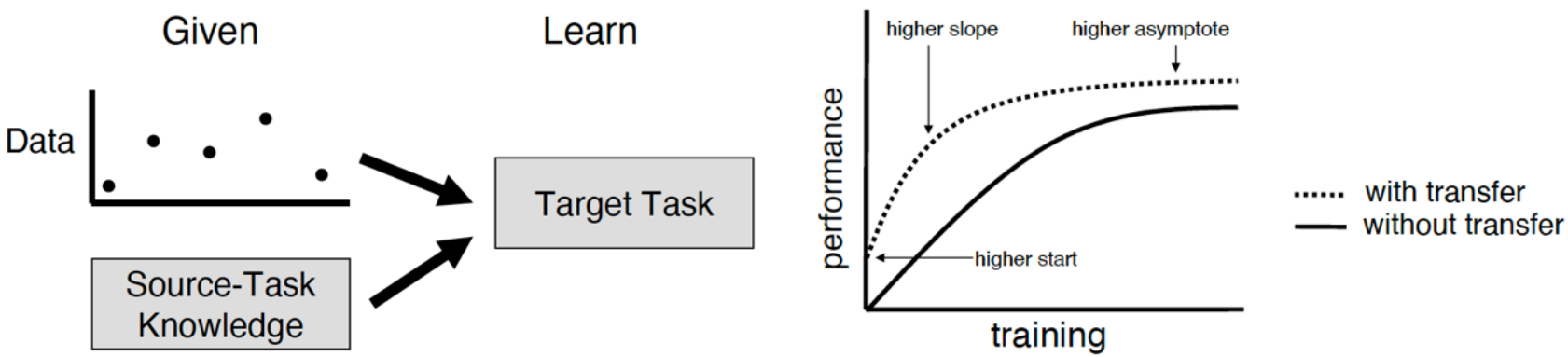
Model	Year	Backbone	Size	$AP_{[0.5:0.95]}$	$AP_{0.5}$	FPS
R-CNN*	2014	AlexNet	224	-	58.50%	~0.02
SPP-Net*	2015	ZF-5	Variable	-	59.20%	~0.23
Fast R-CNN*	2015	VGG-16	Variable	-	65.70%	~0.43
Faster R-CNN*	2016	VGG-16	600	-	67.00%	5
R-FCN	2016	ResNet-101	600	31.50%	53.20%	~3
FPN	2017	ResNet-101	800	36.20%	59.10%	5
Mask R-CNN	2018	ResNeXt-101-FPN	800	39.80%	62.30%	5
DetectoRS	2020	ResNeXt-101	1333	53.30%	71.60%	~4
YOLO*	2015	(Modified) GoogLeNet	448	-	57.90%	45
SSD	2016	VGG-16	300	23.20%	41.20%	46
YOLOv2	2016	DarkNet-19	352	21.60%	44.00%	81
RetinaNet	2018	ResNet-101-FPN	400	31.90%	49.50%	12
YOLOv3	2018	DarkNet-53	320	28.20%	51.50%	45
CenterNet	2019	Hourglass-104	512	42.10%	61.10%	7.8
EfficientDet-D2	2020	Efficient-B2	768	43.00%	62.30%	41.7
YOLOv4	2020	CSPDarkNet-53	512	43.00%	64.90%	31
Swin-L	2021	HTC++	-	57.70%	-	-

[a]Models marked with * are compared on PASCAL VOC 2012, while others on MS COCO.Rows colored gray are real-time detectors (>30 FPS).

[17] Syed Sahil Abbas Zaidi, Mohammad Samar Ansari, Asra Aslam, Nadia Kanwal, Mamoona Asghar, and Brian Lee, "A Survey of Modern Deep Learning based Object Detection Models," *Digital Signal Processing*, pp: 103514, 2022.

Transfer Learning[18]: Vision and Its Potential

- **Goal:** Leverage knowledge from a source task to improve training on target task
- Three potential means to improve learning:
  - Initial performance always better than any ignorant agent.
  - Quick learning with less data
  - Final performance higher than that without transfer.
- Particularly useful in computer vision applications
  - Training on large datasets
  - Most preliminary features needed to learn are the same across tasks



[18] Ref. Lisa Torrey and Jude Shavlik, "Transfer learning," In *Handbook of Research on Machine Learning Applications and Trends: Algorithms, Methods, and Techniques*, pp. 242-264, IGI Global, 2010.

Transfer Learning in Neural Networks: An Experiment[19]

Questions:

- Can we quantify the degree to which a particular layer has general/specific features?
- Does the transition occur suddenly at a single layer, or is it spread out over several layers?
- Where does this transition take place: near the first, middle, or last layer of the network?

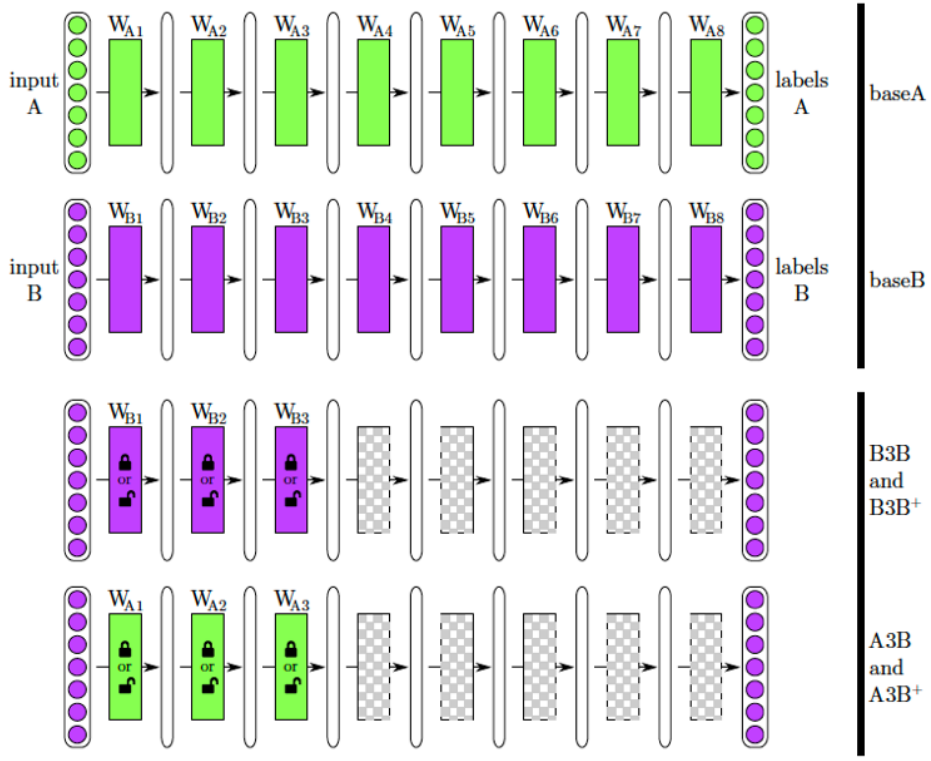
Experiment Design: Consider ImageNet (1000-classes, each class with approx. equal number of images).

- *Task A:* Data corresponding to random 500 classes
- *Task B:* Data corresponding to remaining 500 classes
- *BaseA:* An 8-layer CNN, trained on Task A
- *BaseB:* An 8-layer CNN, trained on Task B

[19] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How Transferable are Features in Deep Neural Networks?," In *Advances in Neural Information Processing Systems(NeurIPS)*, vol. 27, 2014.

Transfer Learning: Experiment Design (cont...)

- ***Selfer Network (B3B):*** Fix layers 1:n in BaseB, initialize layers (n+1):8 randomly, train layers (n+1):8 on Task B
- ***Selfer Network (B3B+):*** Retain layers 1:n in BaseB, initialize layers (n+1):8 randomly, train all layers on Task B
- ***Transfer Network (A3B):*** Fix layers 1:n in BaseA, initialize layers (n+1):8 randomly, train layers (n+1):8 on Task B
- ***Transfer Network (A3B+):*** Retain layers 1:n in BaseA, initialize layers (n+1):8 randomly, train all layers on Task B



Transfer Learning: Experiment Results