Review

- **MobileNets [2017]**: A lightweight DNN using depthwise separable conv (depthwise conv + 1x1 conv), width multiplier and resolution multiplier
- **MobileNets v2 [2018]**: A lightweight DNN with skip connection by using inverted residual block (skip connection between bottlenecks), depthwise conv for expansion layer, and linear bottleneck
- **MnasNet [2019]**: Neural architecture search with a multi-target objective function (including latency) and block-wise search space
- **EfficientNet [2019]**: NAS-based baseline and compound scaling

Transfer Learning

Lecture 4-1

Hyung-Sin Kim

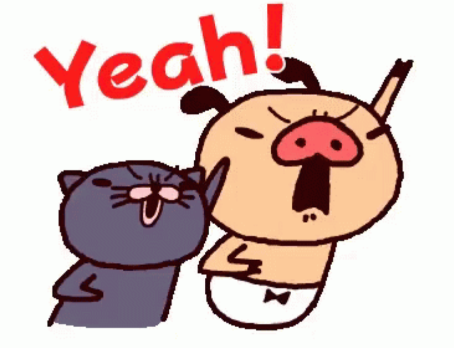
SNU Graduate School of Data Science

Motivation

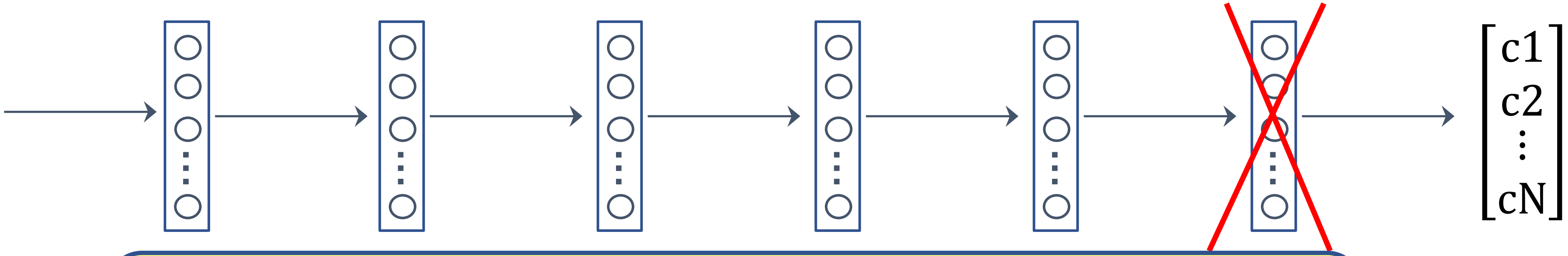
- I want to make a DNN for my application, but don't have a data set big enough to train the DNN (don't have resource to do that, don't want to do that... whatever)

- There is a well-trained DNN out there, which takes the same input type and works for a similar task

- Can I transfer this existing DNN for my own task, with a small data set?

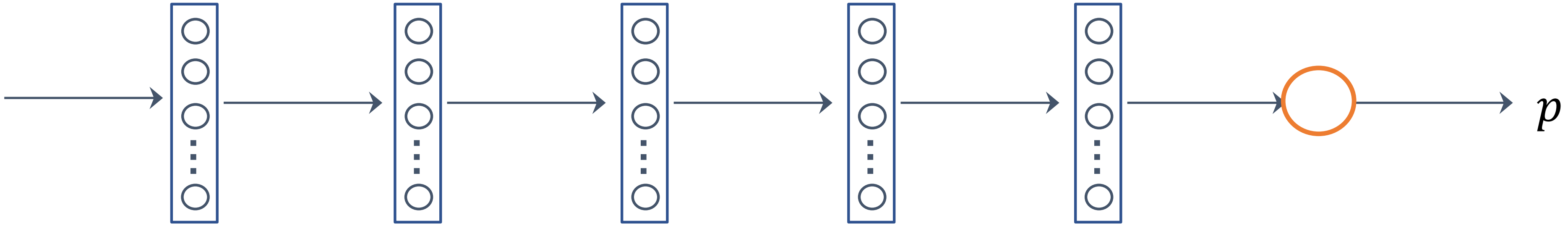


Transfer Learning – How?



Dog?
Fish?
Cat?

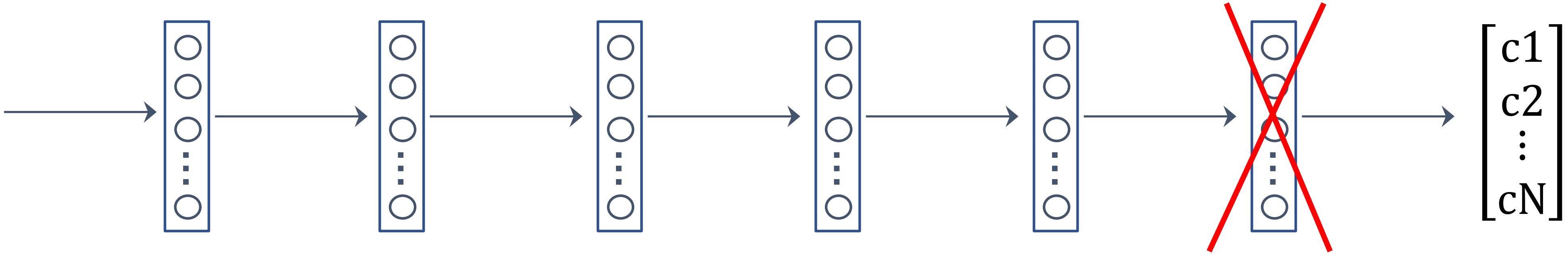
1. Remove the last layer
2. Replace it with another layer for my application
3. Weight initialization of the last layer



COVID or not?

4. Train the last (and more) layers while freezing other layers depending on how much data you have

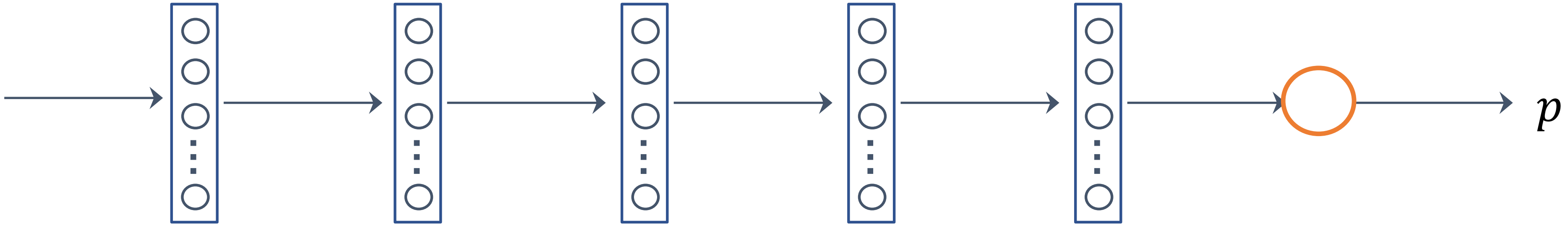
Transfer Learning – Why?



Dog?
Fish?

Cat?

Low level features (edges, corners…) extracted by the pre-trained DNN could be helpful for a new task



COVID
or not?

Two-stage 2D Object Detectors

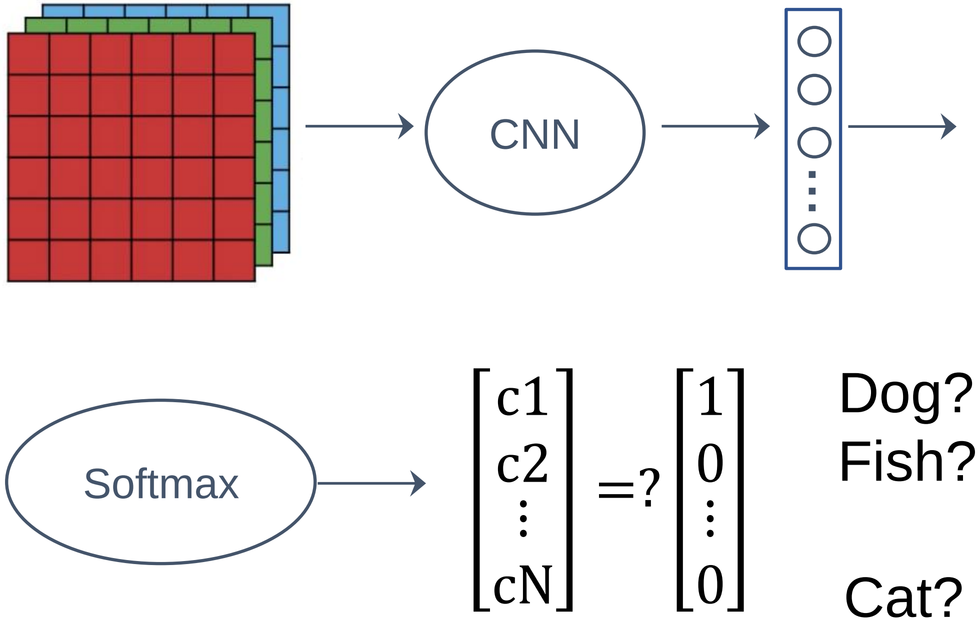
Lecture 4-2

Hyung-Sin Kim

SNU Graduate School of Data Science

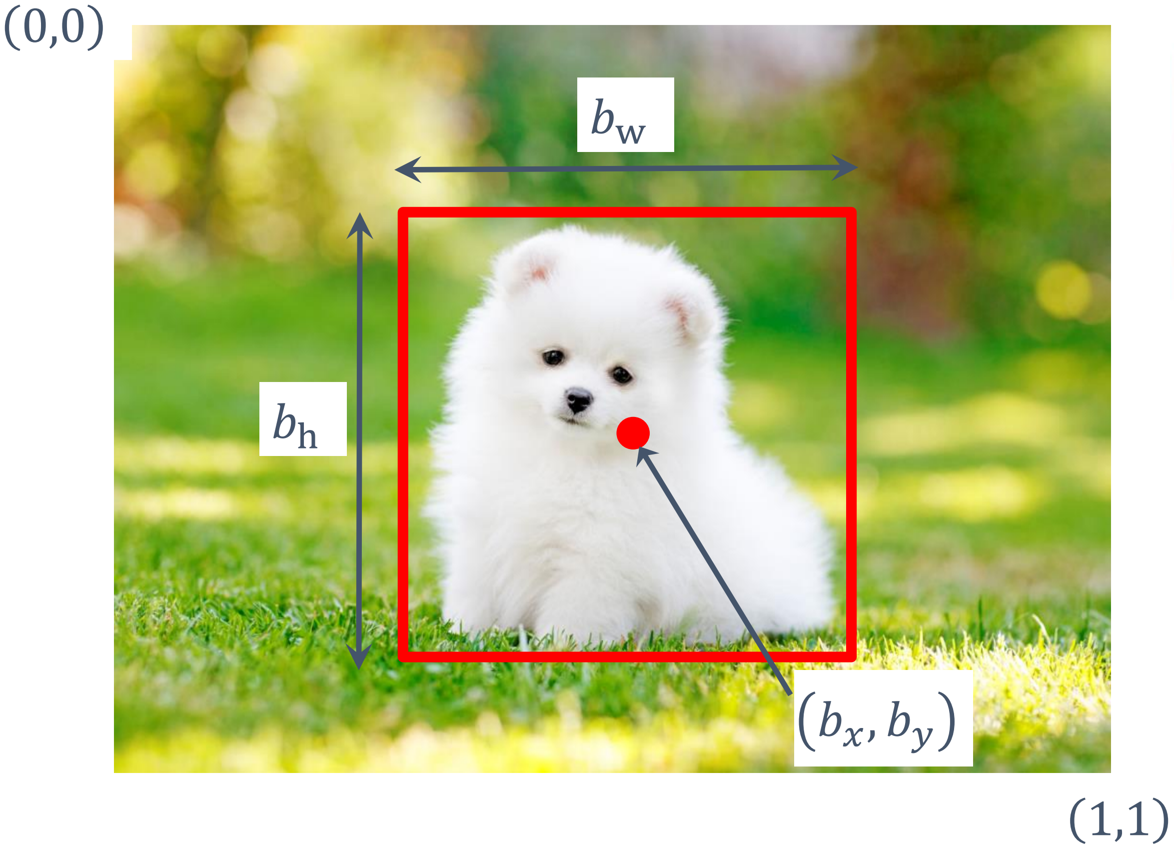
Object Classification

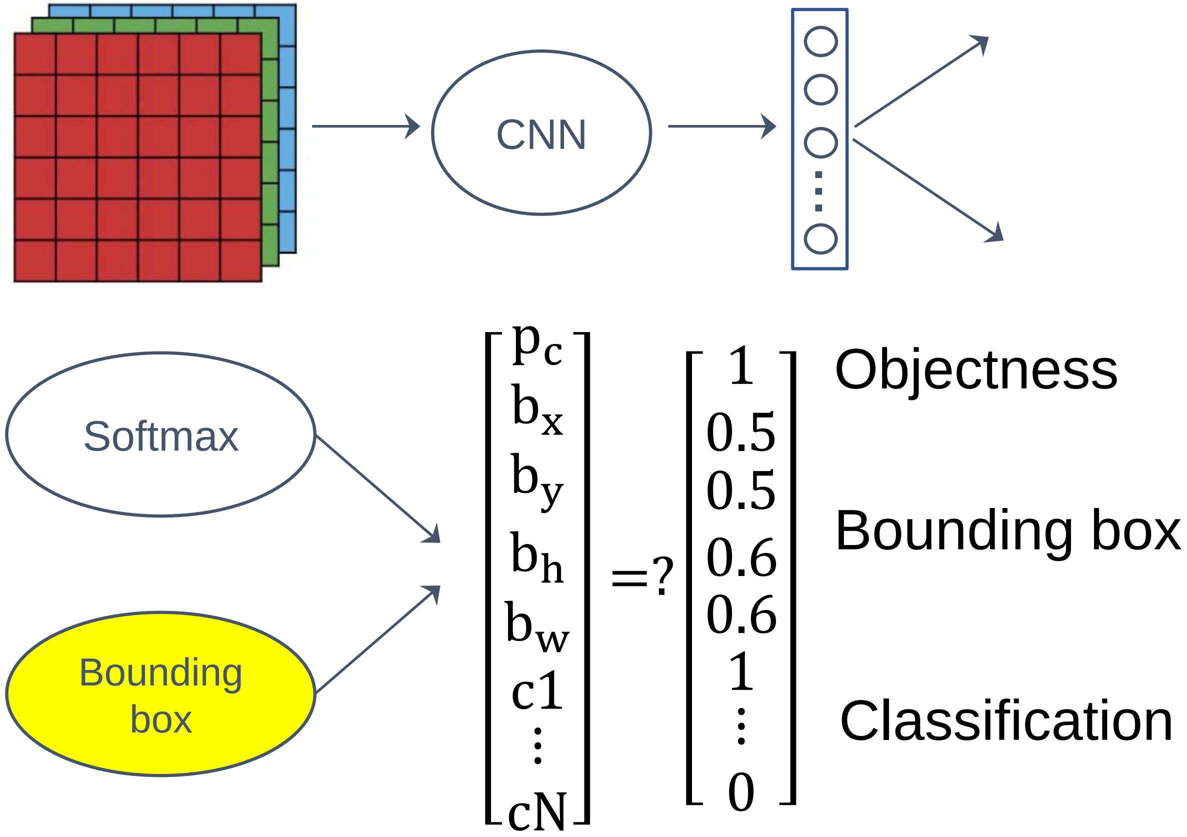
- Assumption: This image has a **single** object.
- Question: What is the object?



Object Classification with Localization

- Assumption: This image **might** have a **single** object.
- Question: What is the object and where is it?

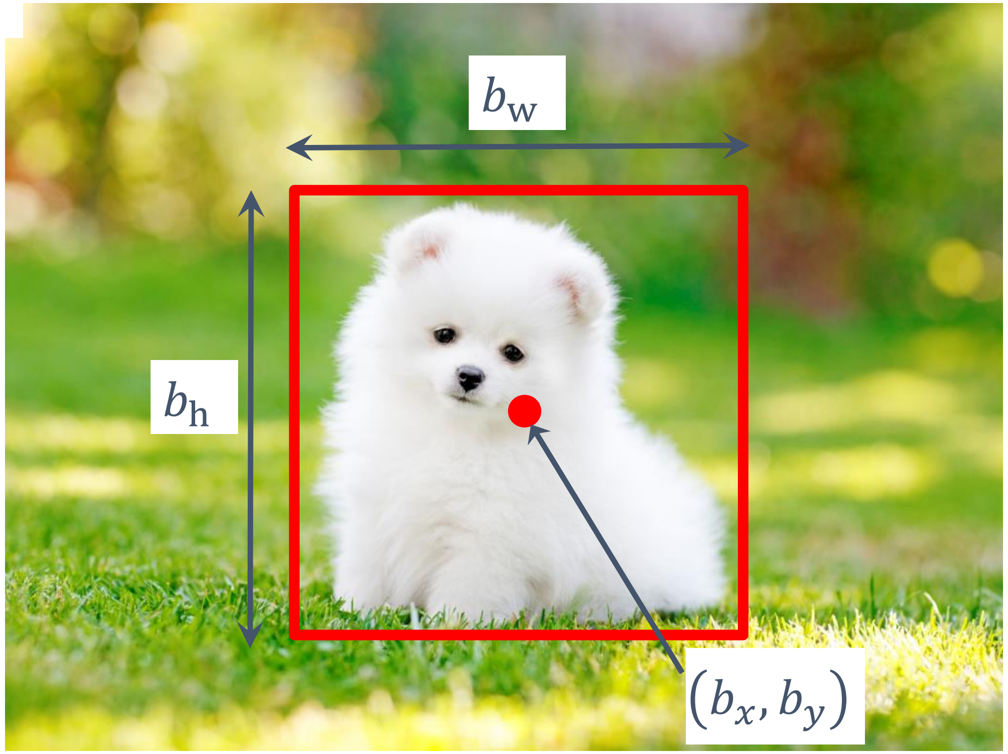




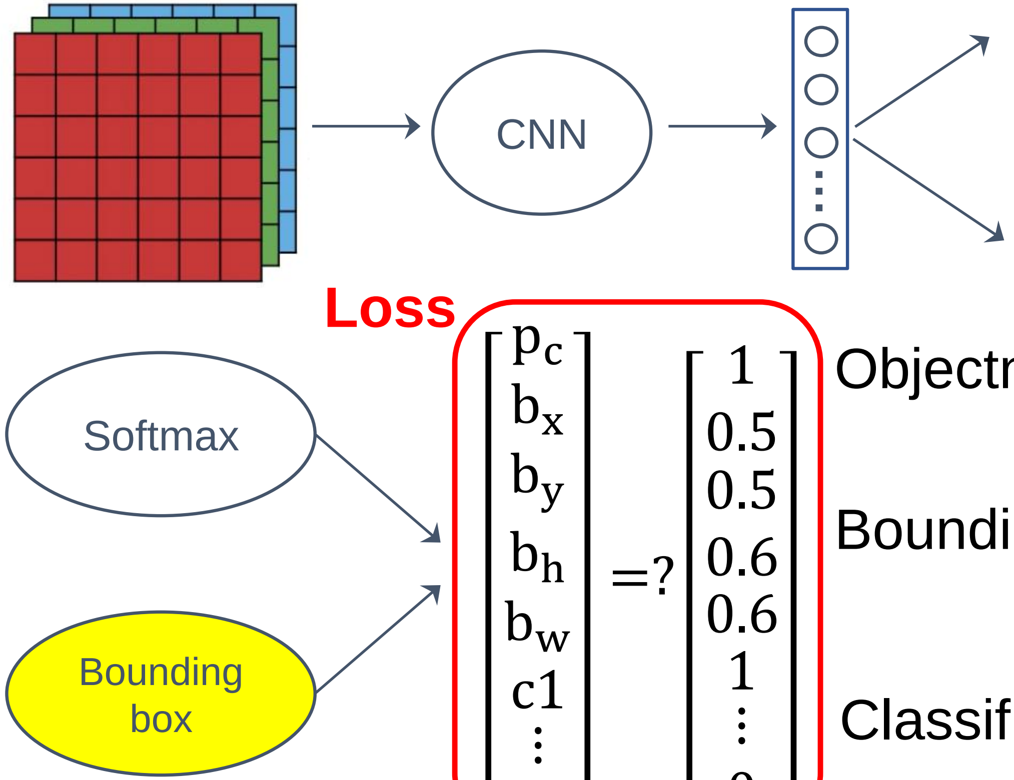
Object Classification with Localization

- Assumption: This image **might** have a **single** object.
- Question: What is the object and where is it, if it exists?

(0,0)



(1,1)



Loss

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c1 \\ \vdots \\ cN \end{bmatrix} =? \begin{bmatrix} 1 \\ 0.5 \\ 0.5 \\ 0.6 \\ 0.6 \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

Objectness

Bounding box

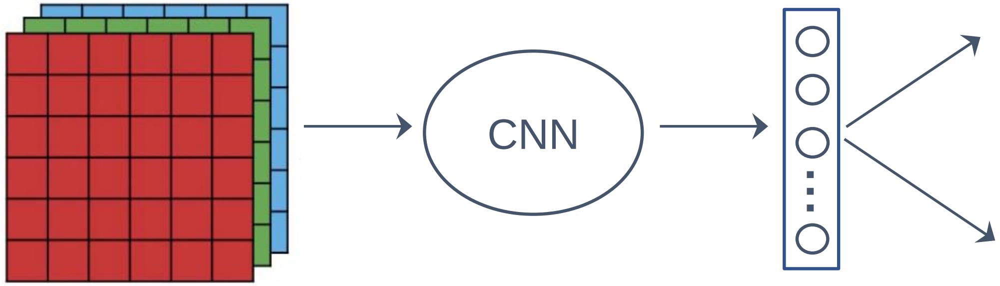
Classification

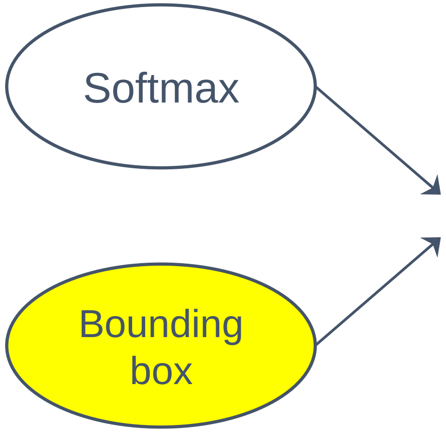
Object Classification with Localization

- Assumption: This image **might** have a **single** object.
- Question: What is the object and where is it, if it exists?

(0,0)

(1,1)





$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c1 \\ \vdots \\ cN \end{bmatrix} =? \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ \vdots \\ ? \end{bmatrix}$$

No object

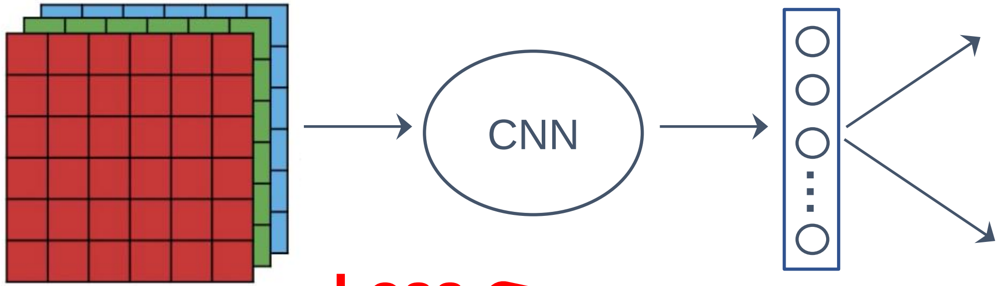
Don't care…

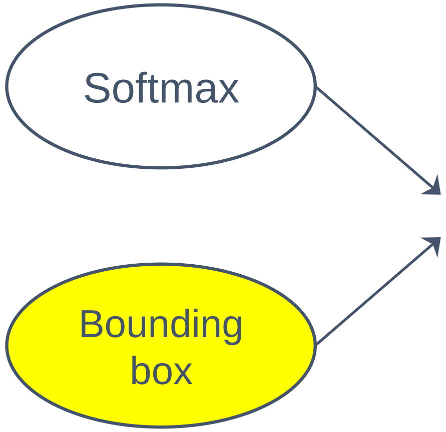
Object Classification with Localization

- Assumption: This image **might** have a **single** object.
- Question: What is the object and where is it, if it exists?

(0,0)

(1,1)





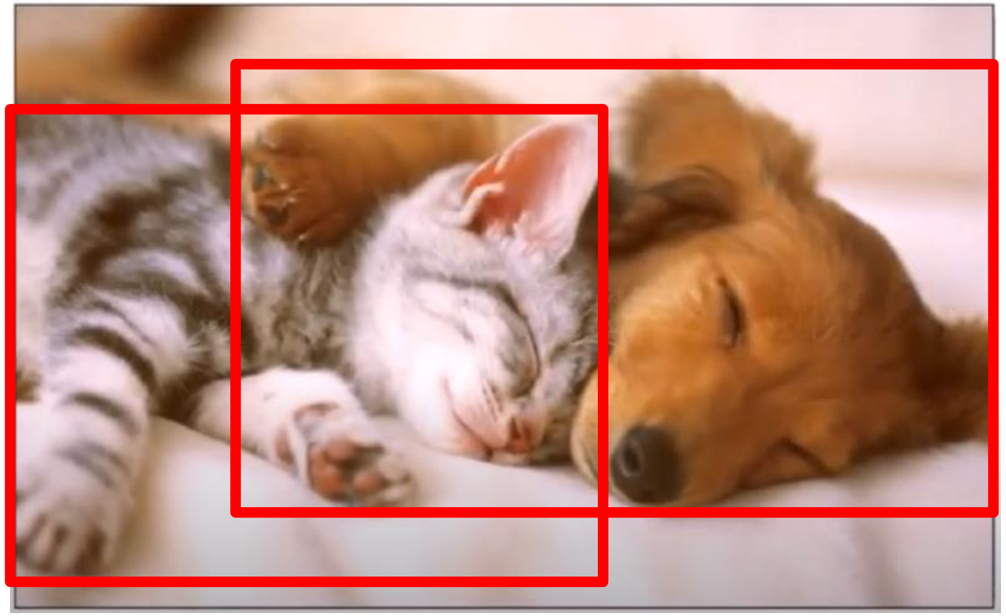
Loss

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c1 \\ \vdots \\ cN \end{bmatrix} =? \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ \vdots \\ ? \end{bmatrix}$$

No object

Don’t care…

Object Detection – Doing it for Multiple Objects!



How to propose bounding boxes?

Sliding Windows Detection

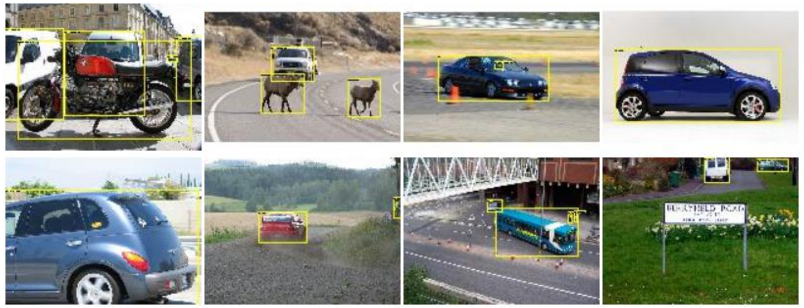
- Step 1) Train a CNN for object detection
  - Training set = bounding boxes **full** of an object
  - Classify an object in a bounding box image

bird

bus

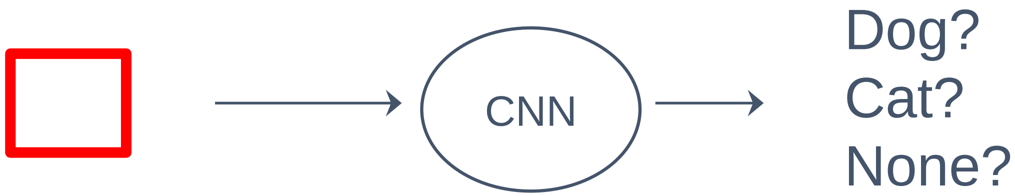
airplane

car



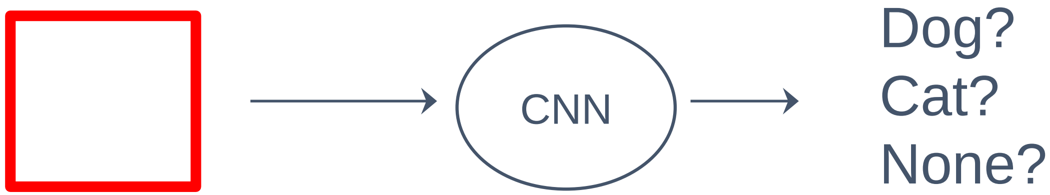
Sliding Windows Detection

- Step 2) Propose a bounding box and feed it to the CNN
  - For every location and every size …



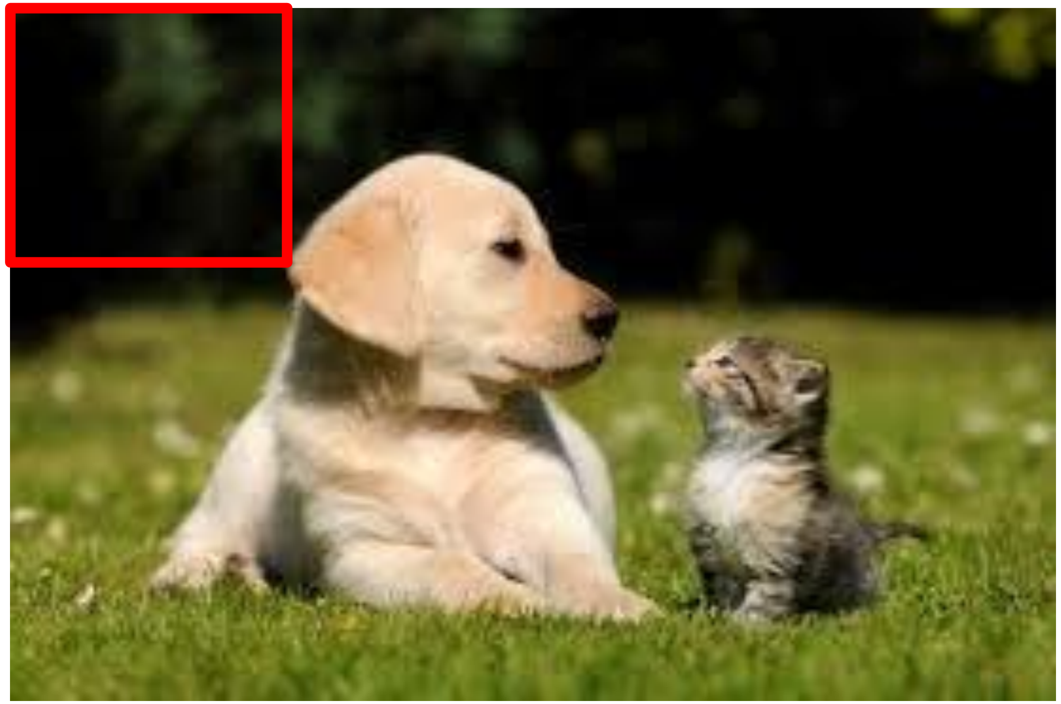
Sliding Windows Detection

- Step 2) Propose a bounding box and feed it to the CNN
  - For every location and every size …



Sliding Windows Detection

- Step 2) Propose a bounding box and feed it to the CNN
  - For every location and every size …



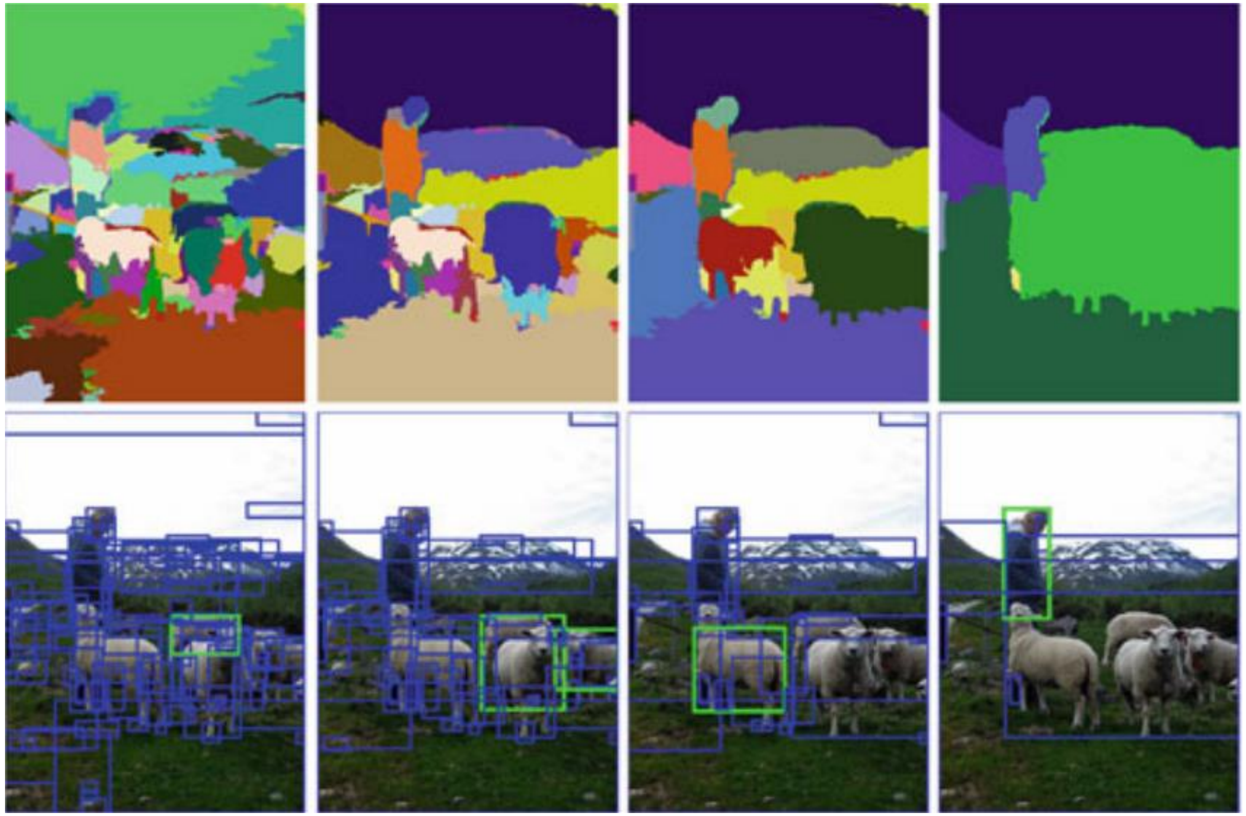
→ CNN → Dog? Cat? None?

You run CNN so….many times…
Huge computation overhead!

How can we propose fewer bounding boxes without losing detection performance?

Selective Search [IJCV'13]

- Let's propose bounding boxes a bit smarter by analyzing relationship between pixels

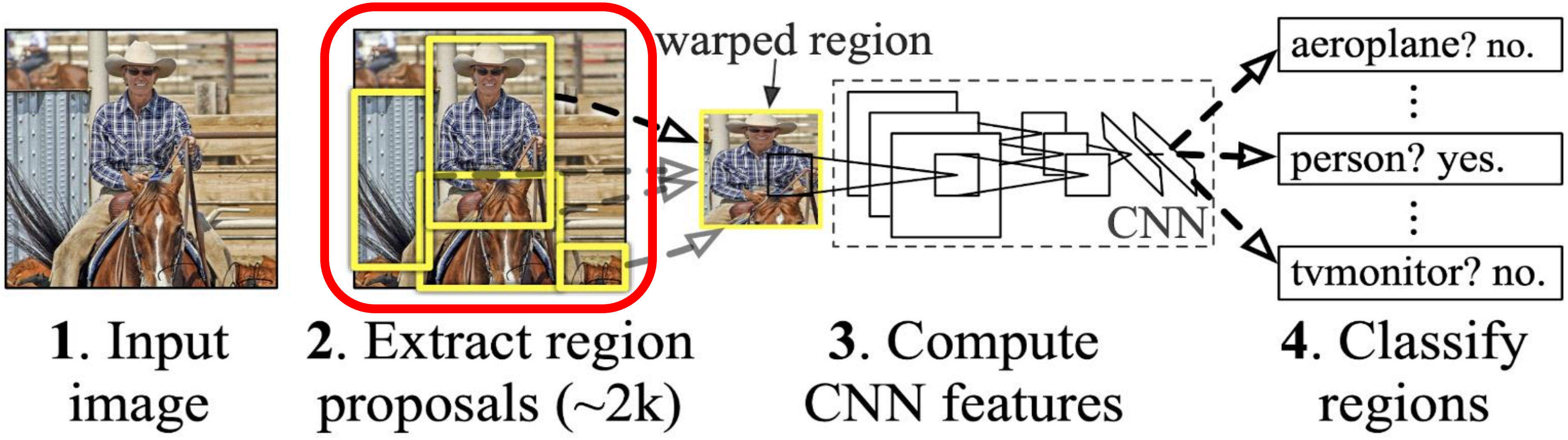


Fewer bounding boxes!

[The figures are from JRR. Uijlings et al., "Selective search for object recognition."]

R-CNN [CVPR'14] – Detection Flow

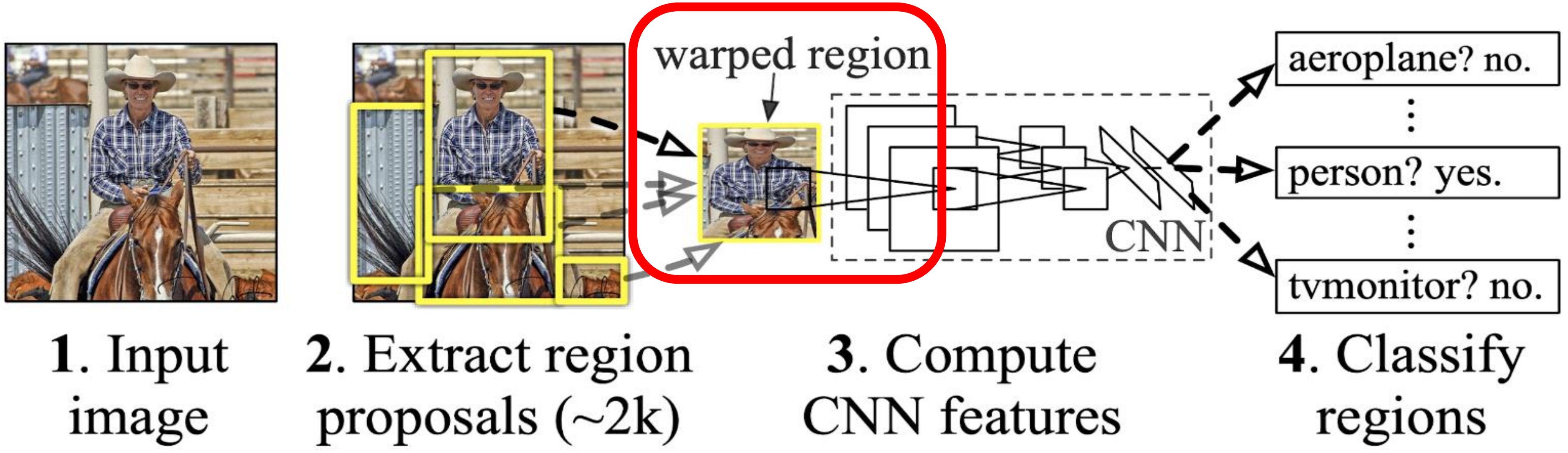
- Step 1) Region proposal: Use Selective search technique to propose **only 2,000 bounding boxes**



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

R-CNN [CVPR'14] – Detection Flow

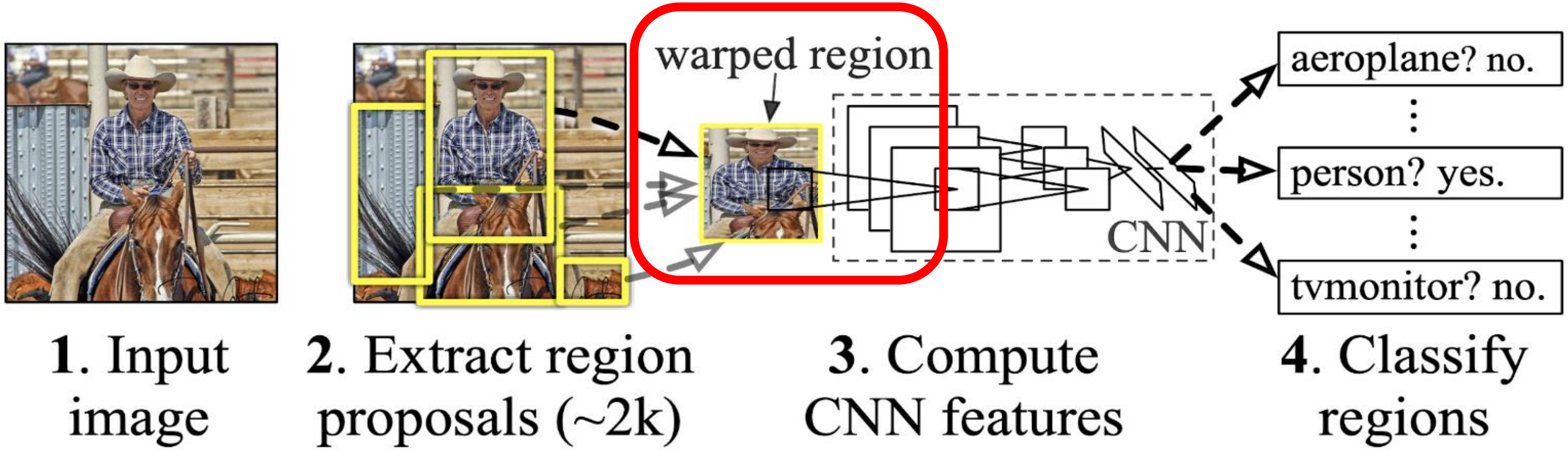
- Step 1) Region proposal: Use Selective search technique to propose **only 2,000 bounding boxes**
- Step 2) Warping: Re-size each box (227x227) to make it an input for one CNN



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

R-CNN [CVPR’14] – Resizing… Why?

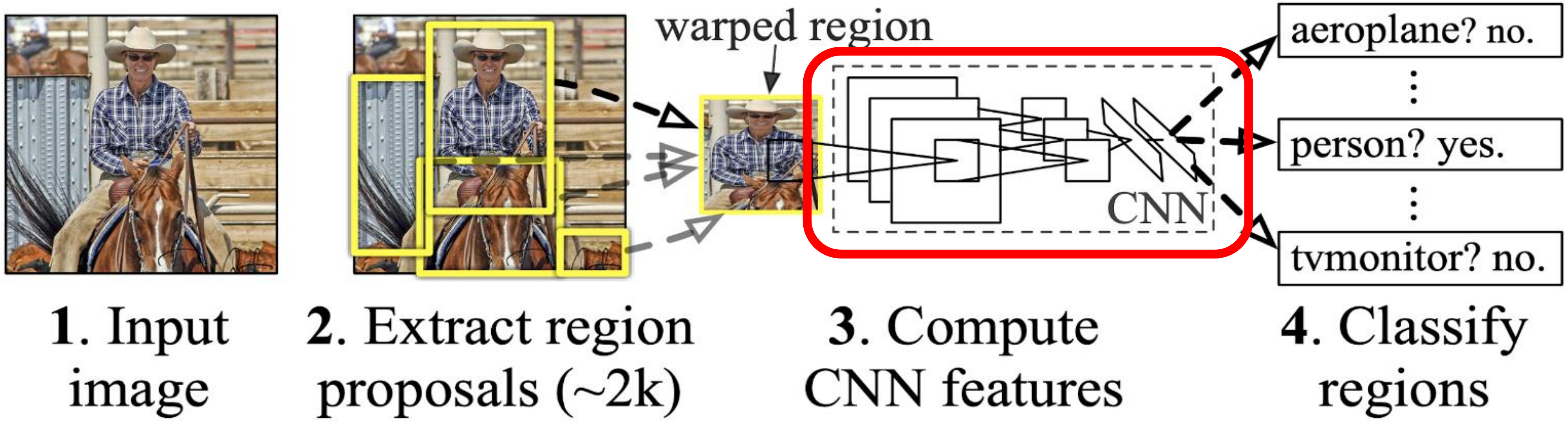
- Convolutional layers operate regardless of input size
  - filter size, stride, and padding, all of which have nothing to do with input size
  - Output size is proportional to input size
- However, **fully connected layers** have fixed input and output sizes
  - Therefore, CNN input size should be set considering its FC layers at the end



[The figures are from R. Girshick et al., “Rich feature hierarchies for accurate object detection and semantic segmentation.”]

R-CNN [CVPR'14] – Detection Flow

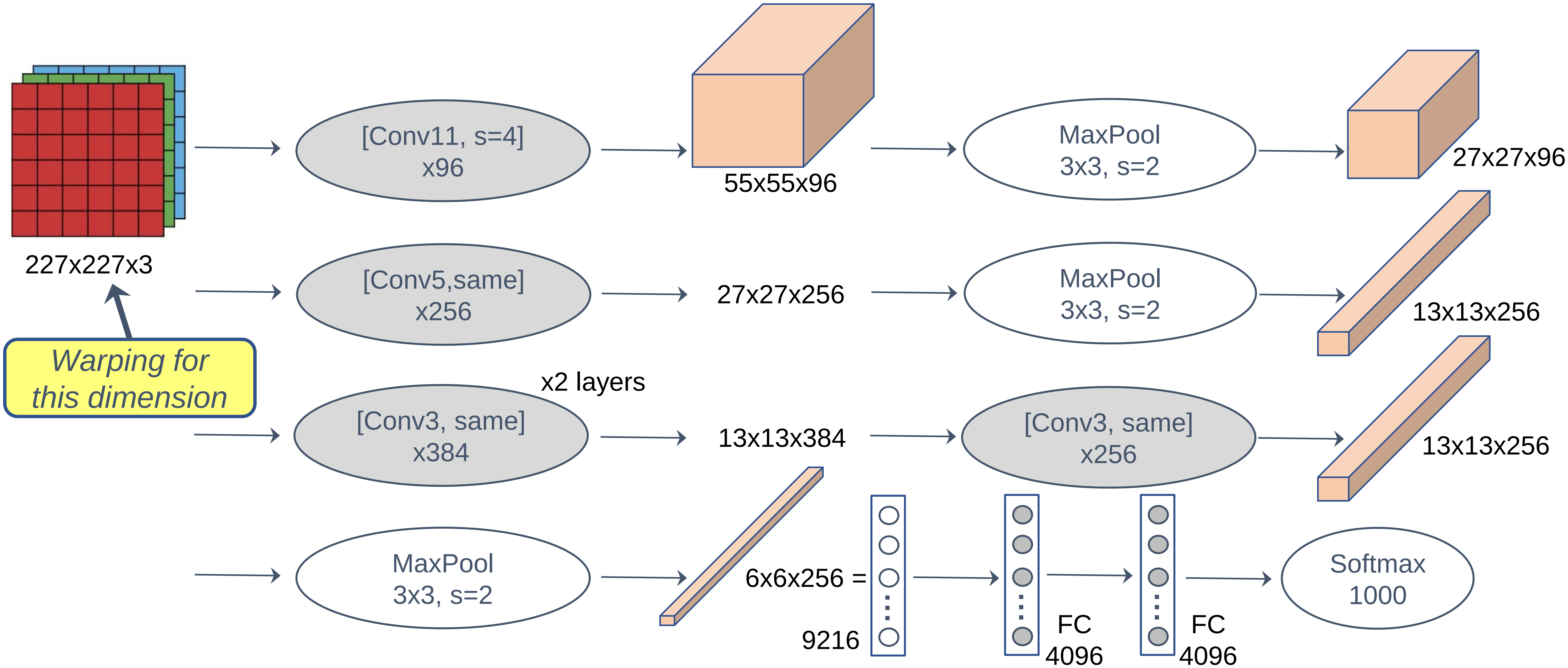
- Step 1) Region proposal: Use Selective search technique to propose only **2,000 bounding boxes**
- Step 2) Warping: Re-size each box (227x227) to make it an input for one CNN
- Step 3) Feature extraction: By using a CNN



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

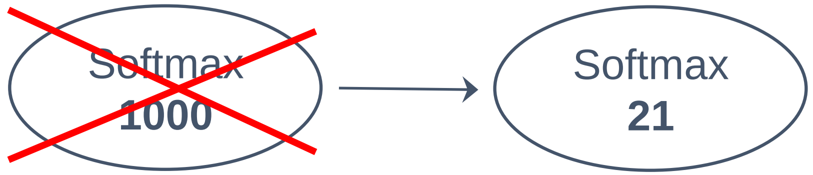
What CNN architecture does R-CNN use,
*given that the paper was published in **2014**?*

R-CNN [CVPR'14] – AlexNet [2012] for CNN



R-CNN [CVPR'14] – Transfer Learning for CNN

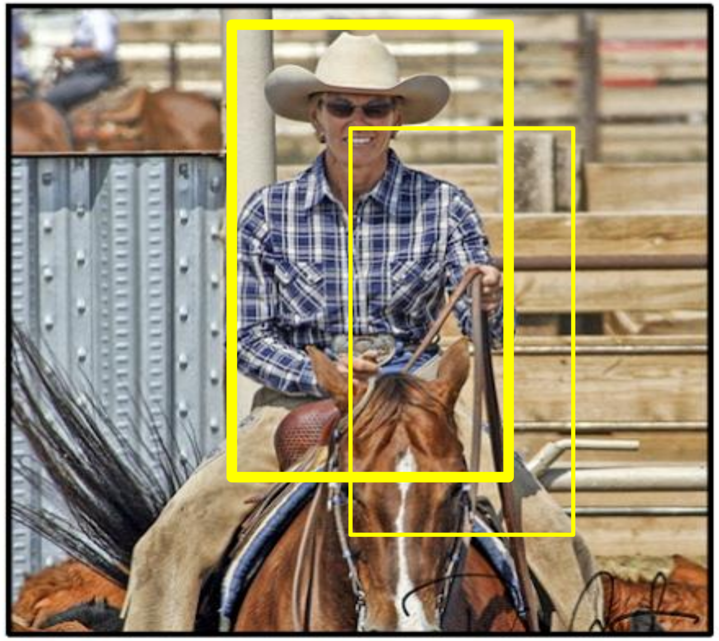
- (Pre)train AlexNet on ImageNet (1000-way classification)
- Detection dataset (PASCAL) has only 20 classes
- Replace the 1000-way Softmax classifier with 21-way Softmax (1 for background)



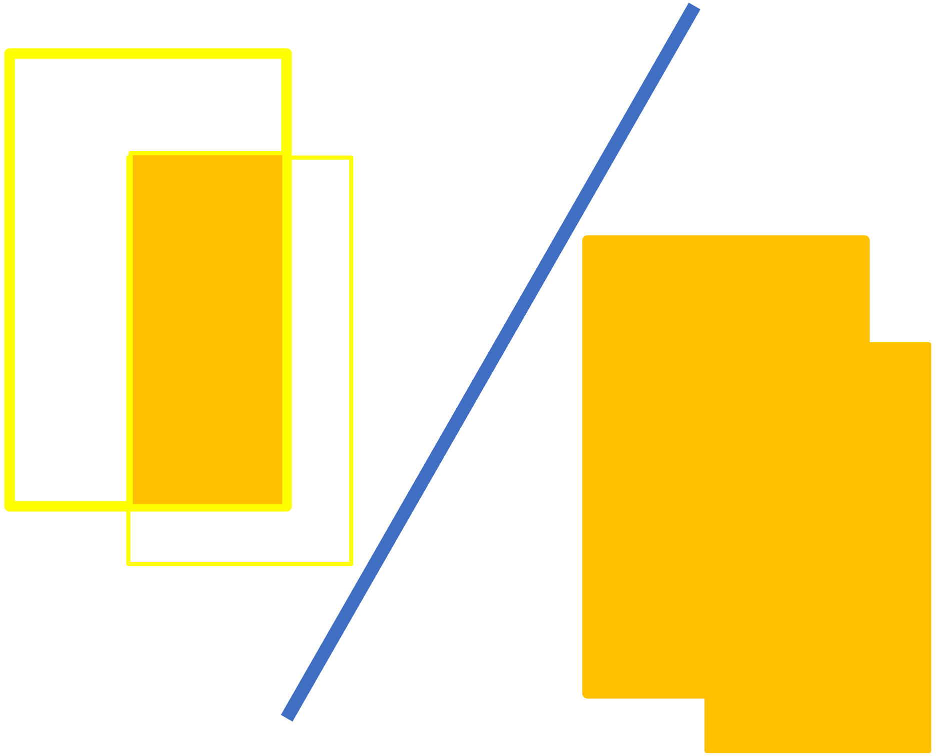
- Randomly initialize weights toward the 21 Softmax
- Fine tune by using VoC data set

R-CNN [CVPR’14] – Fine Tuning Methodology

- Labeling proposed bounding boxes
  - If a bounding box has **>0.5 IoU** with class A’s ground truth bounding box, the bounding box’s label for class A is positive (1).
  - Otherwise, its label for class A is negative (0).



IoU
(Intersection over Union)



[The figures are from R. Girshick et al., “Rich feature hierarchies for accurate object detection and semantic segmentation.”]

R-CNN [CVPR'14] – Fine Tuning Methodology

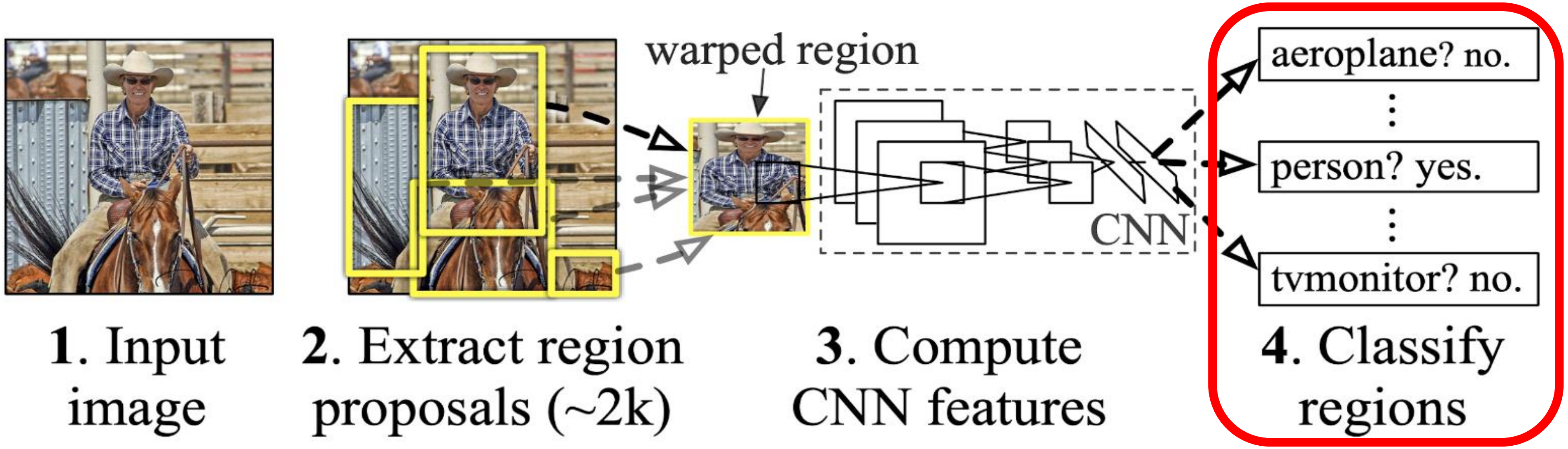
- Labeling proposed bounding boxes
  - If a bounding box has **>0.5 IoU** with class A's ground truth bounding box, the bounding box's label for class A is positive (1).
  - Otherwise, its label for class A is negative (0).
  - By doing this, we can have 30x more positive samples, which mitigates overfitting

- Learning rate: 0.001 (1/10 of that used for pre-training)
  - Fine tune can make progress without clobbering pre-trained weights

- Mini-batch gradient decent
  - 128 bounding boxes: 32 with a valid object and 96 with only background
  - Why sampling fewer positive cases than backgrounds? There are much more background boxes than those with an object

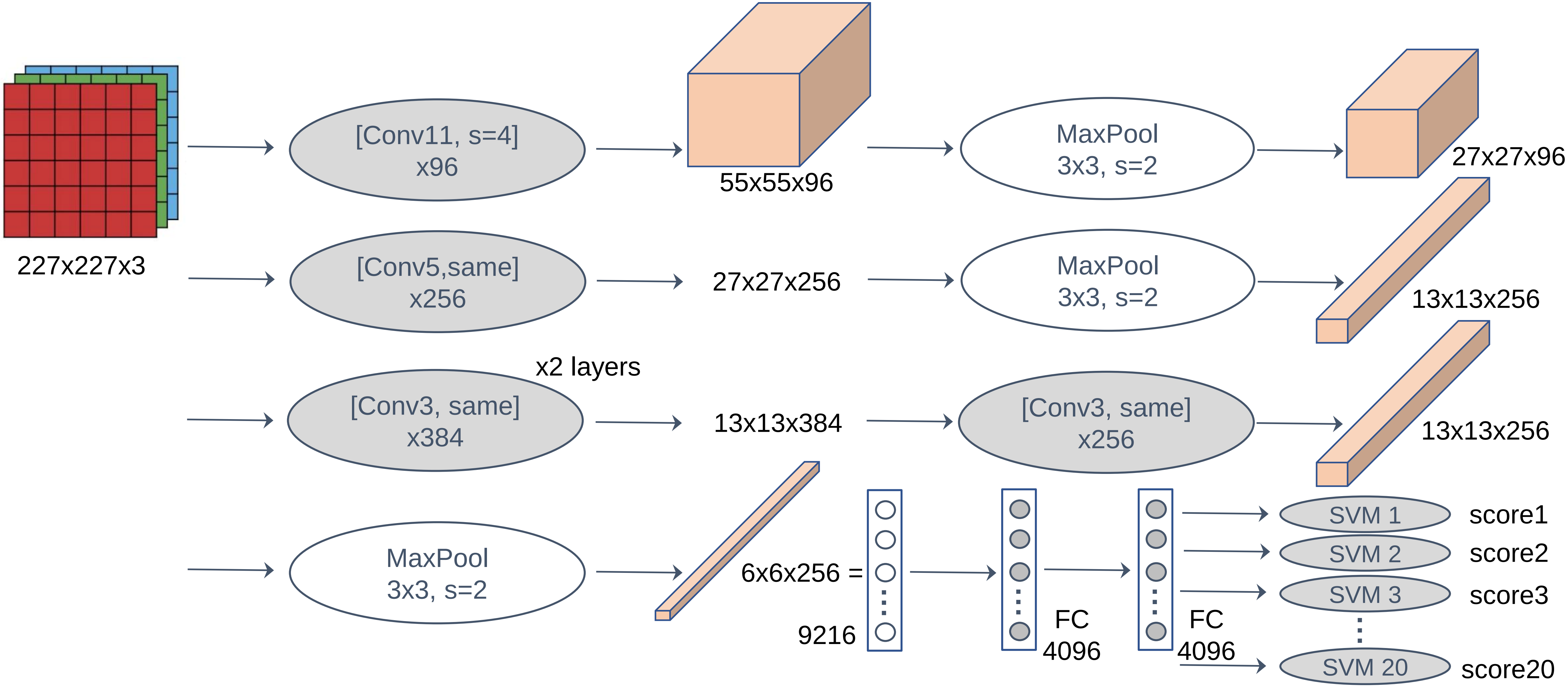
R-CNN [CVPR'14] – Detection Flow

- Step 4) Classification: a class-specific **SVM** scores the feature vector in the box for that class
  - Wait… the CNN already has 21-way Softmax classifier!
  - Yes, we will remove it and add 20 SVMs



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

R-CNN [CVPR'14] – Class-specific SVMs



R-CNN [CVPR'14] – SVM Training

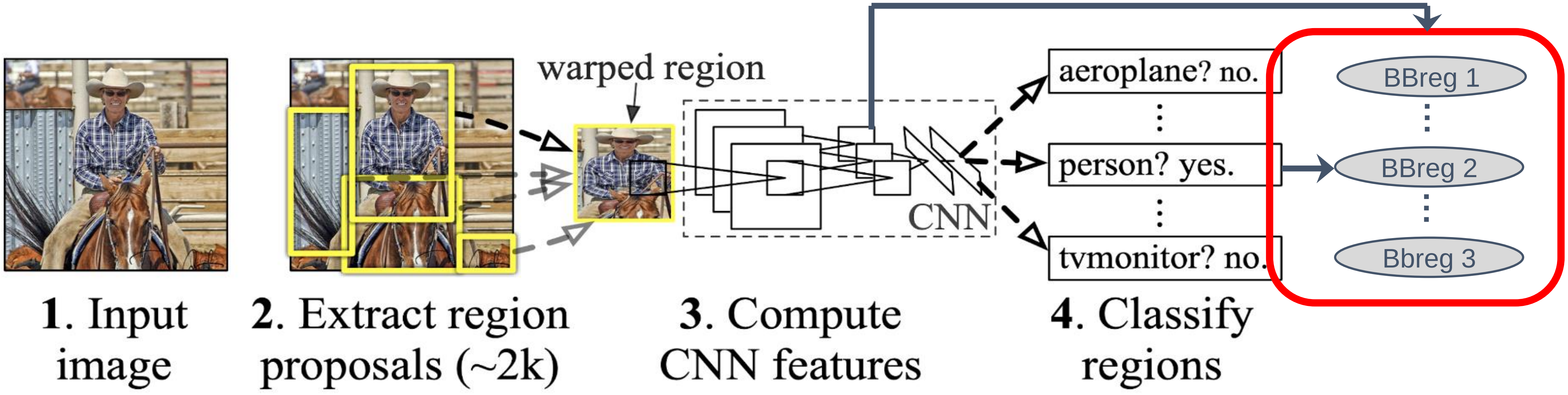
- CNN is freezed, providing a feature vector for SVMs as the input
  - **Separate training**: CNN should be trained first
  - **Training data preparation**: CNN output feature vector (after the final FC layer) for each box proposal should be stored in a **disk**

- Labeling bounding box proposals (training data)
  - SVM has much fewer parameters than an FC layer
    - Less risk of overfitting
    - Few labeled samples are enough
  - Methodology
    - If a bounding box is class A's ground truth bounding box, the bounding box's label for class A is positive (1)
    - If a bounding box has <0.3 **IoU** with the ground truth, its label for class A is negative (0)
    - Otherwise, when IoU is between 0.3 and 1, the bounding box is **NOT** used for training

- Better performance than using 21-way Softmax layer

R-CNN [CVPR’14] – Detection Flow

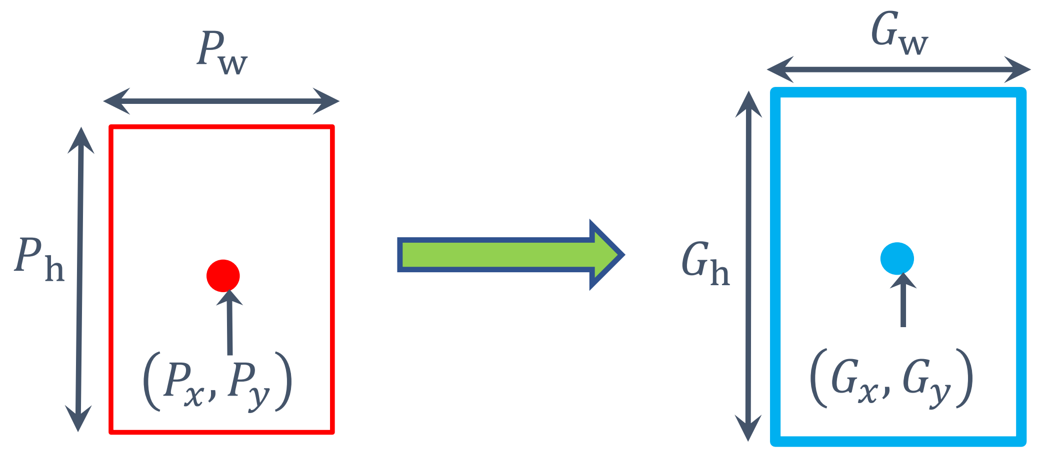
- Step 5) Bounding box regression: a class-specific **linear** regressor
  - Now we have bounding boxes having valid objects and know the class of these objects
  - However, the bounding boxes given by selective search may be quite different from ground-truth boxes, which need to be **fine-tuned** to reduce localization errors

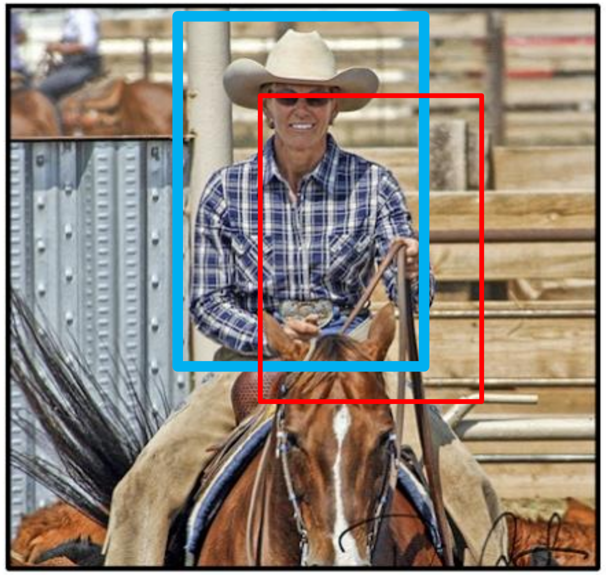


[The figures are from R. Girshick et al., “Rich feature hierarchies for accurate object detection and semantic segmentation.”]

R-CNN [CVPR'14] – BBreg Training

- Fine-tune 4 elements of each bounding box
  - A proposed BB becomes a valid training data if and only if it is nearby (IoU > 0.6) at least one ground-truth box
  - BBreg is trained on BB coordinates and **CNN feature vectors** (before FC layers)
    - Again, CNN should be trained first and its feature vectors should be stored





[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

R-CNN [CVPR’14] – BBreg Training

- Fine-tune 4 elements of each bounding box
  - $\hat{G}_x = \boxed{P_w} \boldsymbol{d_x(P)} + P_x$
  - $\hat{G}_y = \boxed{P_h} \boldsymbol{d_y(P)} + P_y$
  - $\hat{G}_w = \boxed{P_w} \exp(\boldsymbol{d_w(P)})$
  - $\hat{G}_h = \boxed{P_h} \exp(\boldsymbol{d_h(P)})$
  - $d_x(P) = \boldsymbol{w_x^T} f(P)$
  - $d_y(P) = \boldsymbol{w_y^T} f(P)$
  - $d_w(P) = \boldsymbol{w_w^T} f(P)$
  - $d_h(P) = \boldsymbol{w_h^T} f(P)$

$\boldsymbol{d_*(P)}$: Class-specific linear regressor
$\boldsymbol{f(P)}$: A BB’s feature vector from CNN

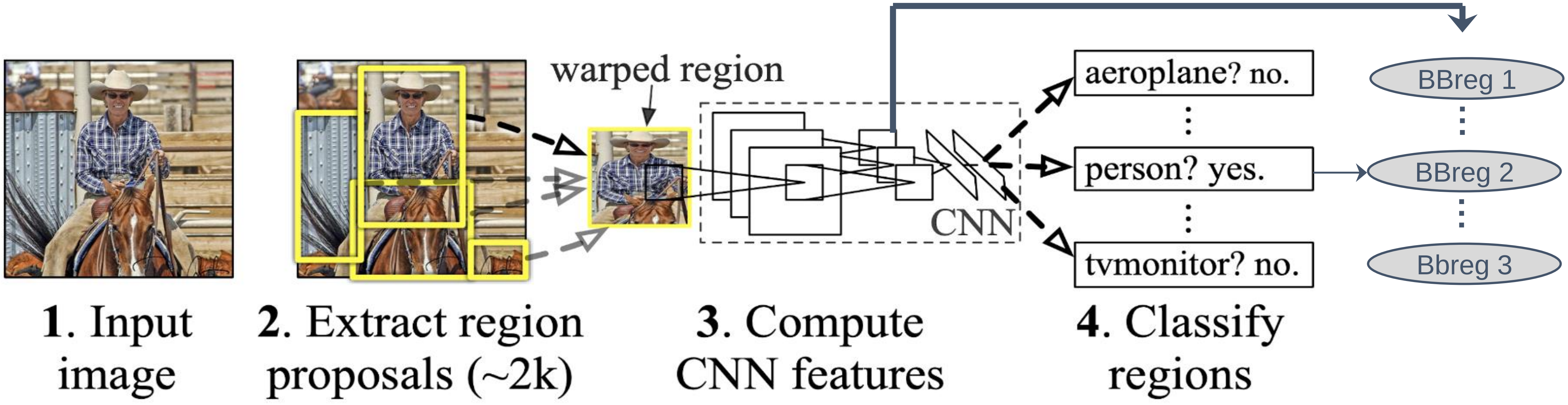
Normalization for scale-invariant transformation

- $t_x^i = \frac{G_x^i - P_x^i}{P_w^i}$
- $t_y^i = \frac{G_y^i - P_y^i}{P_h^i}$
- $t_w^i = \log(\frac{G_w^i}{P_w^i})$
- $t_h^i = \log(\frac{G_h^i}{P_h^i})$

- $\boldsymbol{w_*} = argmin_{\boldsymbol{\hat{w}_*}} \left[\sum_i^N (t_*^i - \boldsymbol{\hat{w}_*^T} f(P^i))^2 + \lambda \|\boldsymbol{\hat{w}_*}\|^2\right]$
  - A standard regularized least square problem (closed form solution)

R-CNN [CVPR'14] – Detection Flow

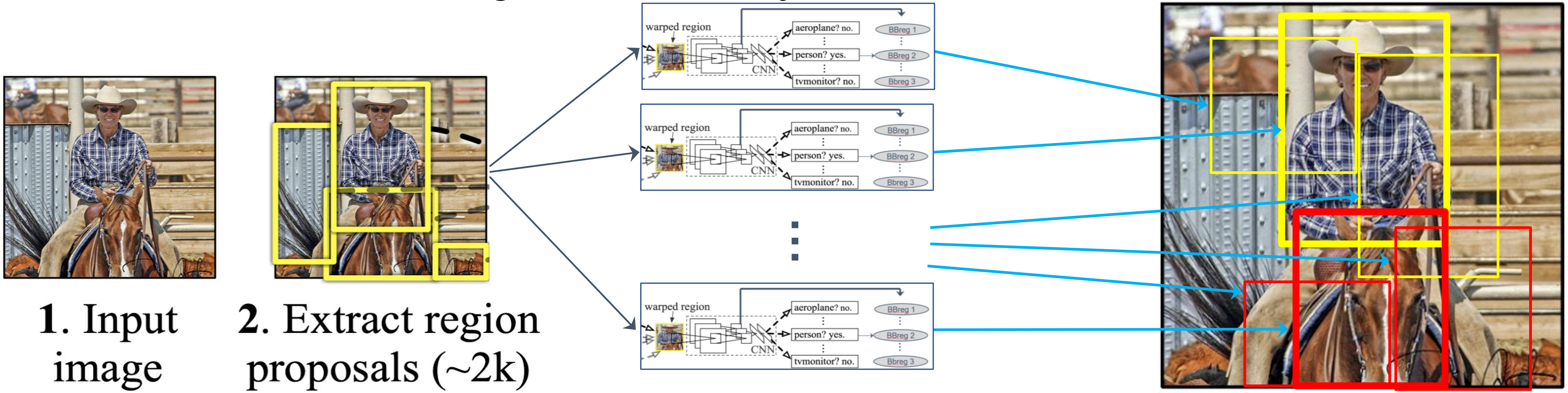
- Step 6) Non-max suppression
  - After SVM and BBreg, now we have **fine-tuned** bounding boxes having valid objects and know the class of these objects
  - However, there could be **many** bounding boxes for one object. We need to select the **best** bounding box for each object



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

R-CNN [CVPR'14] – Detection Flow

- Step 6) Non-max suppression
  - After SVM and BBreg, now we have **fine-tuned** bounding boxes having valid objects and know the class of these objects
  - However, there could be **many** bounding boxes for one object. We need to select the **best** bounding box for each object



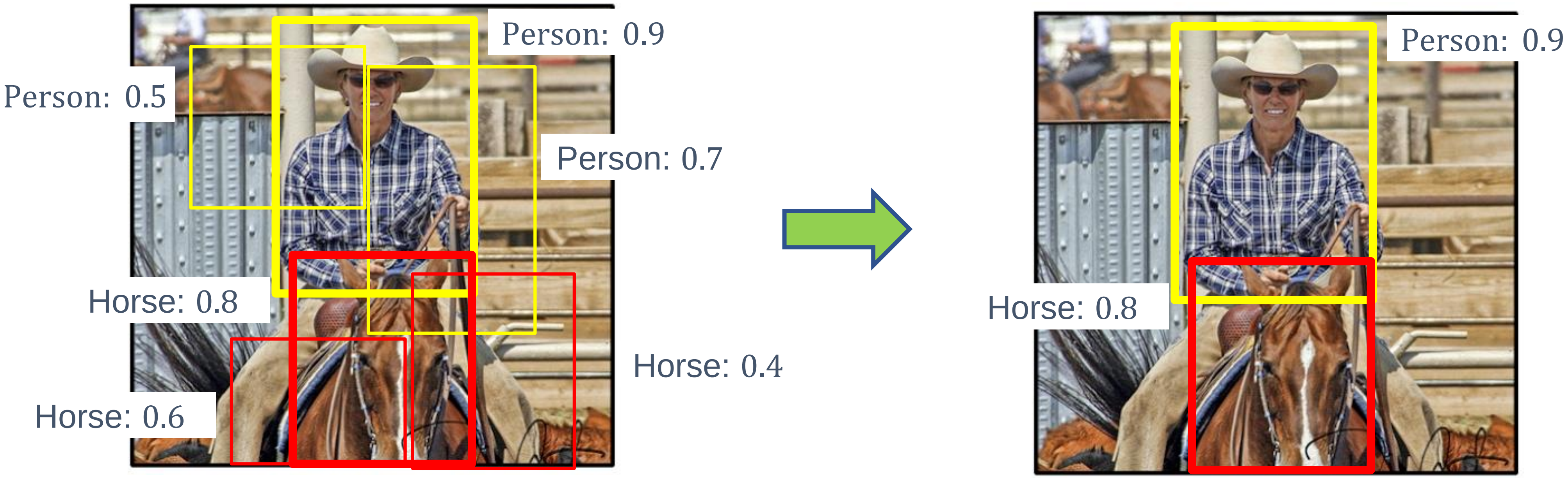
1. Input image

2. Extract region proposals (~2k)

[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

R-CNN [CVPR'14] – Detection Flow

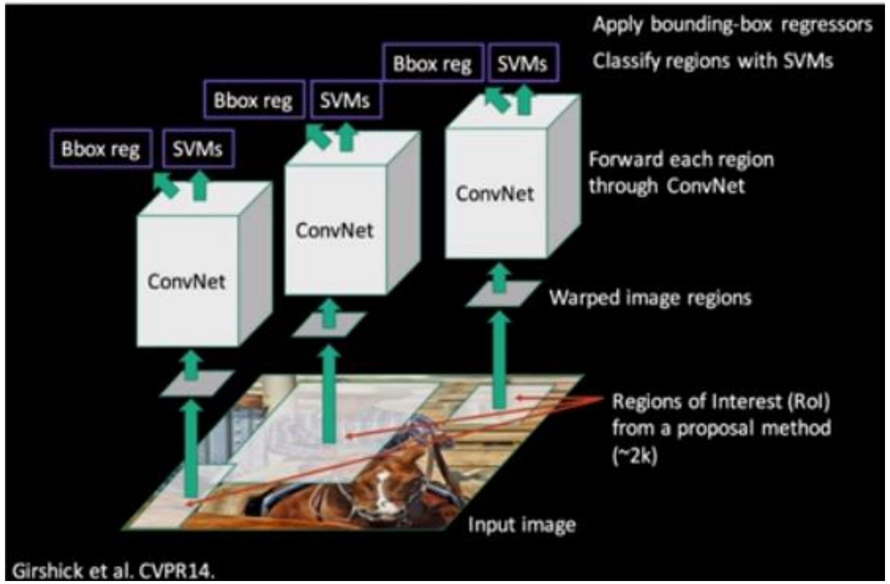
- Step 6) Non-max suppression: For each class, repeat the following two steps until checking all bounding boxes containing the class object
  - 1) Select a region with the highest score
  - 2) Reject regions that overlap with the best region (high IoU)



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

R-CNN [CVPR'14] – Limitations

- **Still** slow detection at test time
  - Have to run CNN, SVM, BB regression **~2,000 times** (for each bounding box)
  - **CPU** implementation of selective search
  - **~40 seconds** to process an image ☹
- Training **many models**, all separately
  - Train a CNN on ILSVRC 2012
  - Fine tune the CNN on VOC
  - **Store** output feature vectors of CNN
    - ~2000 feature vectors for each image…
  - Train class-specific SVMs on CNN features
  - Train class-specific BBregs on CNN features using SVM outputs
- Information loss
  - Warping and cropping to provide a fixed-size input for CNN

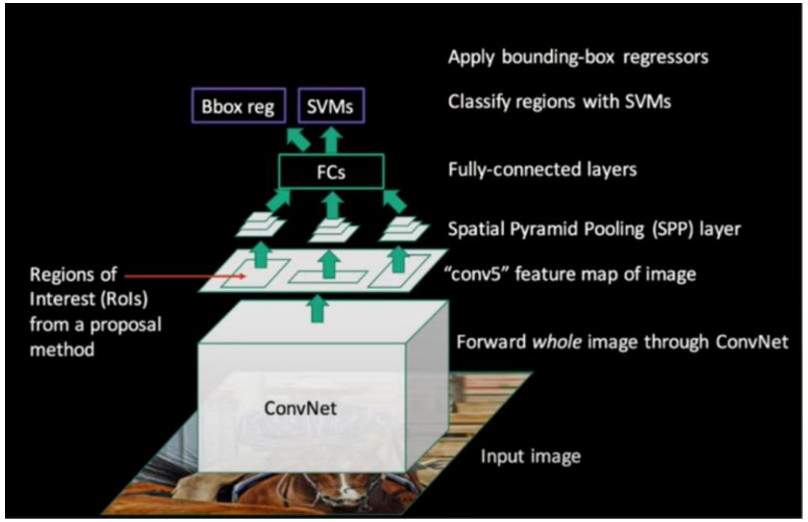


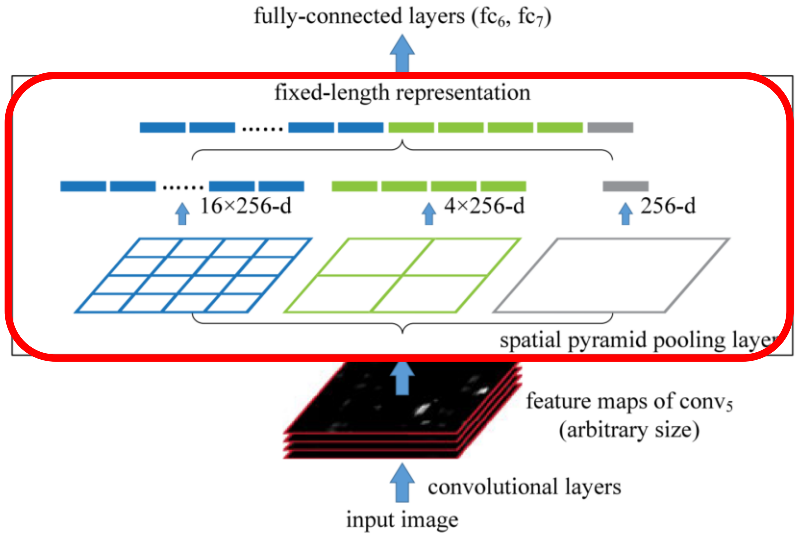
Let's fix the problems!

*1) Running CNN **only once** per image to save time*
*2) **Arbitrary input** size for CNN to improve accuracy*

SPPNet [TPAMI'15]

- Crop boxes (Region of Interest) **on a CNN feature map** (before FC layers), instead of an original image
  - 3x faster training and 10~100x faster inference by running CNN once per image
- **Spatial pyramid pooling (SPP) layers**, instead of warping, to resize each box proposal before FC layers
  - In contrast to regular sliding window pooling, SPP divides an input into a **fixed number** of spatial bins, resulting in a fixed size output **regardless of input size**





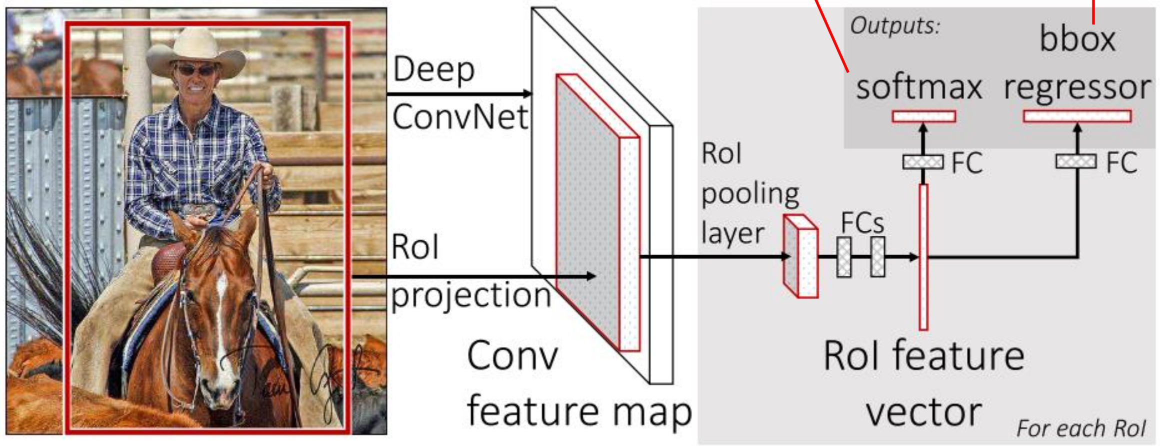
[The figures are from https://www.youtube.com/watch?v=Jo32zrxr6l8. and "Spatial pyramid pooling in deep convolutional networks for visual recognition."]

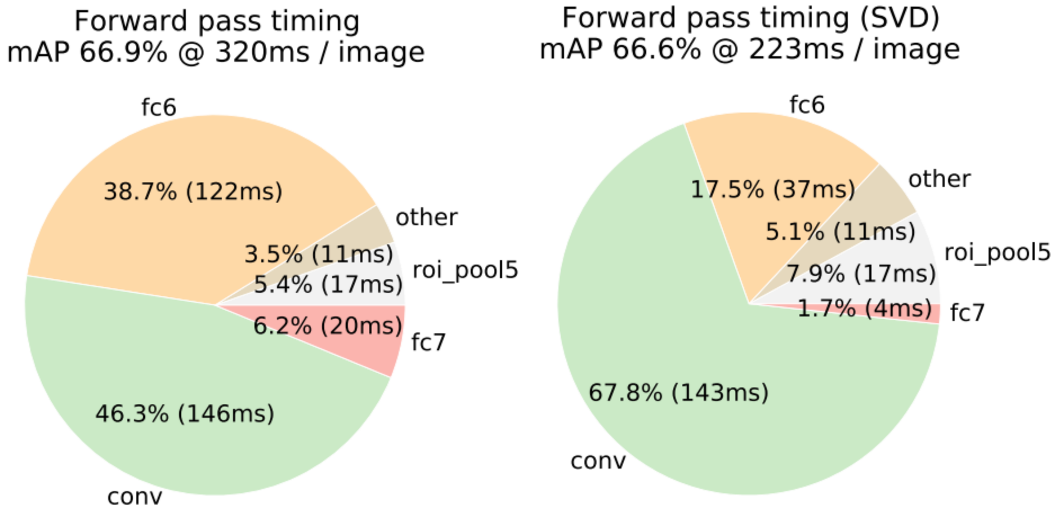
Let’s fix the problems!

1) *Running CNN **only once** per image to save time*
2) ***Arbitrary input** size for CNN to improve accuracy*
3) ***Single-stage training** to save training time and memory*

Fast R-CNN [CVPR'15]

- **Single-stage** training (3x faster training than SPPnet, save xxx GB storage)
  - **Softmax**, instead of SVM, that does not need to be trained
  - **Multi-task** loss that includes both classification and localization errors
    - $L(p, u, t^u, v) = \boxed{L_{\text{cls}}(p, u)} + \boxed{\lambda[u \geq 1] L_{\text{loc}}(t^u, v)}$





- Truncated SVD to compress FC layers, Softmax, and a single BBreg
  - 10x faster inference compared to SPPnet

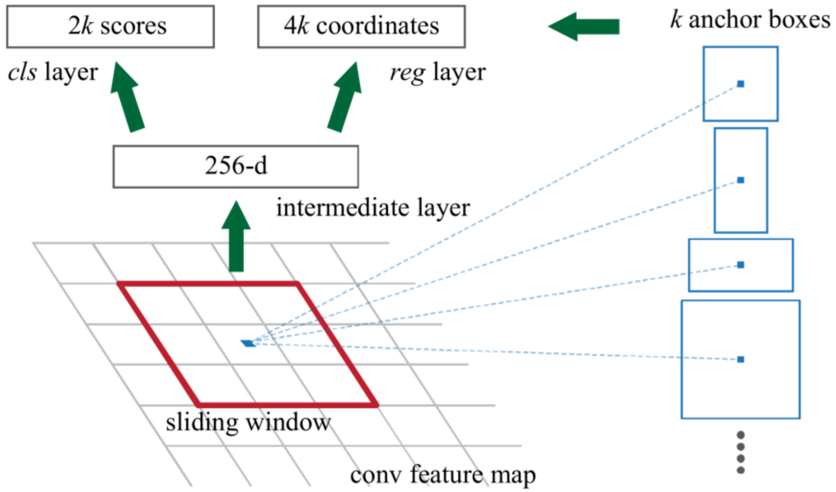
[The figures are from https://www.youtube.com/watch?v=Jo32zrxr6l8. and "Fast R-CNN."]

Let's fix the problems!

1) *Running CNN **only once** per image to save time*
2) ***Arbitrary input** size for CNN to improve accuracy*
3) ~~*3) **Single-stage training** to save training time and memory*~~
4) *Fast region proposal using **GPU***

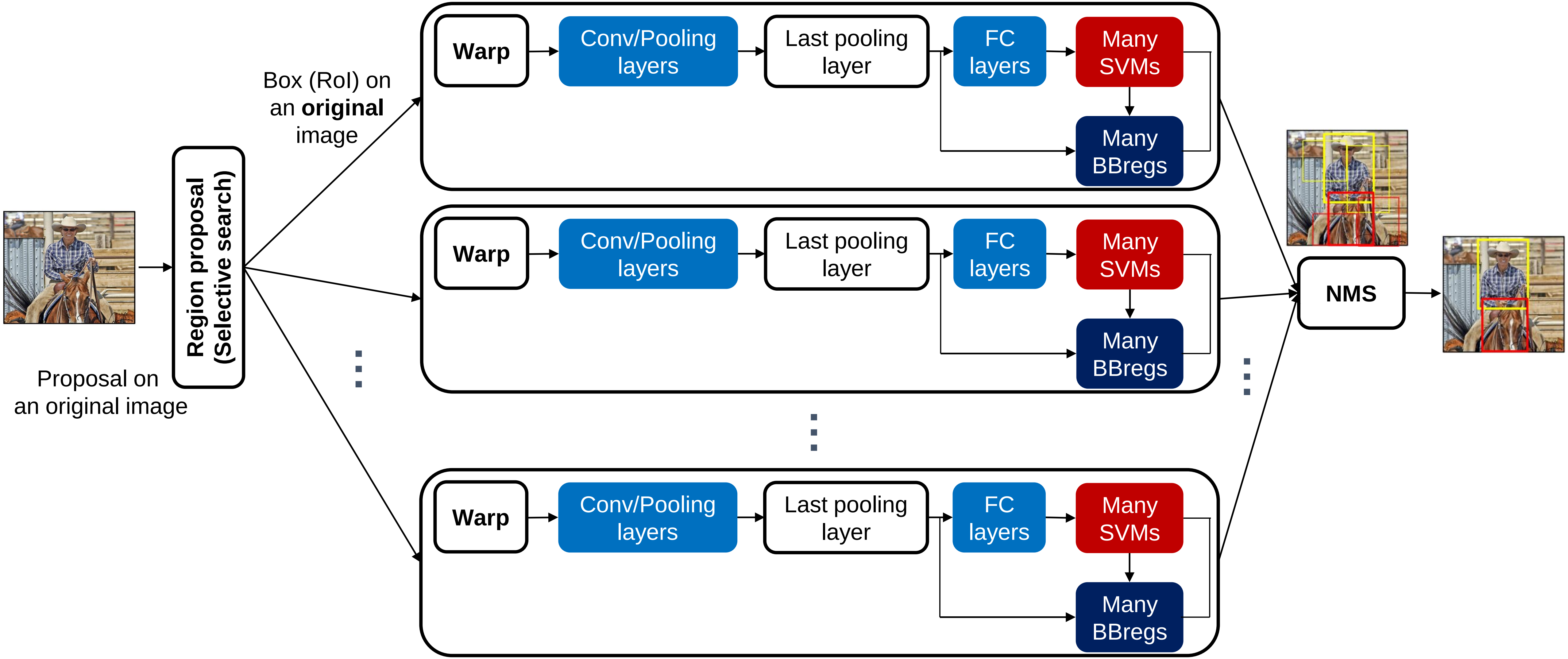
Faster R-CNN [NIPS'15]

- Region proposal network (**RPN**), instead of selective search
  - CPU ⟹ GPU, Original image ⟹ Conv feature map
  - Structure: Sliding window (3x3 size), 9 anchor boxes per window, and FC layers
  - Output: ~300 boxes, Coordinate (4 values) and objectness probability for each box
  - 10x faster inference than Fast R-CNN
- Four-stage training again since (shared) CNN affects both RPN and Fast R-CNN training process
  - 1. (pretrained) **CNN** and **RPN**
  - 2. (pretrained) **CNN**, (fixed) RPN and **Fast R-CNN**
  - 3. (fixed) CNN, **RPN** and (fixed) Fast R-CNN
  - 4. (fixed) CNN, (fixed) RPN and **Fast R-CNN**



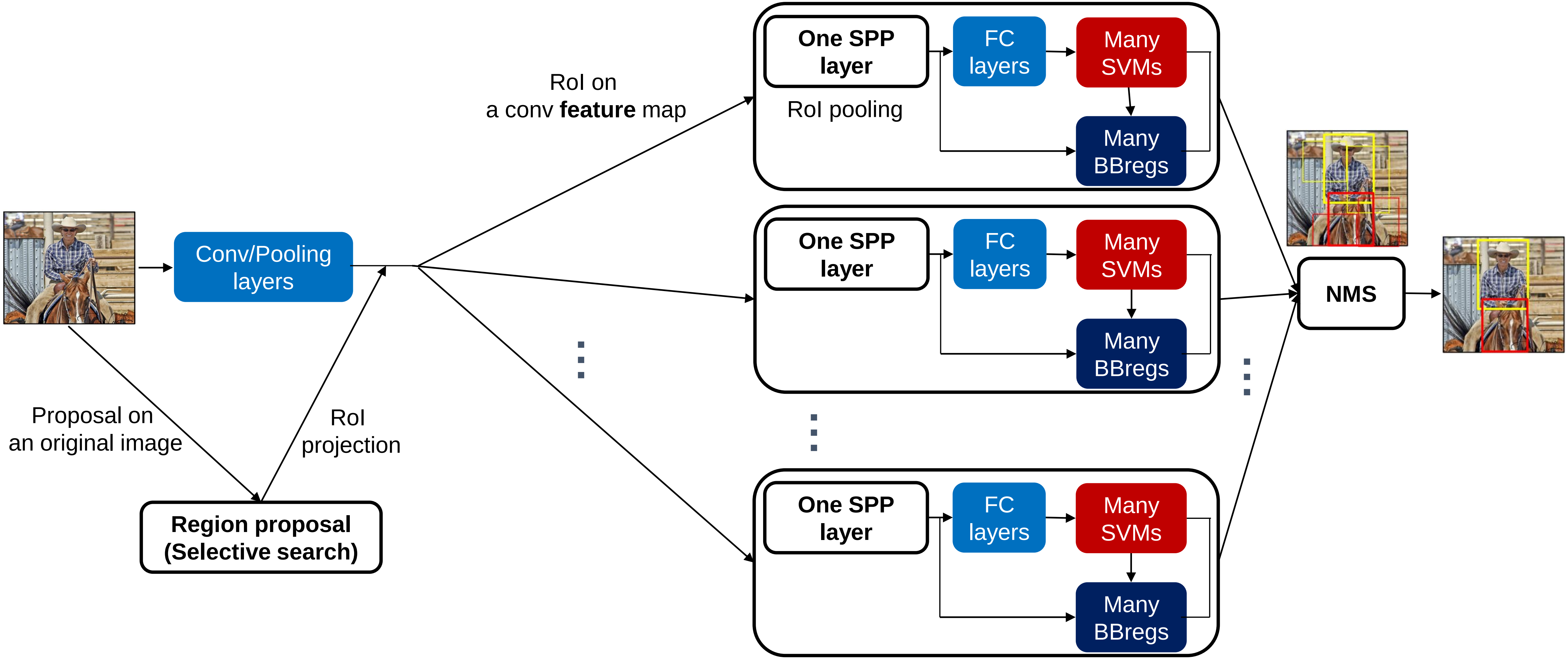
[The figures are from Ren etl al., "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks."]

Summary – R-CNN



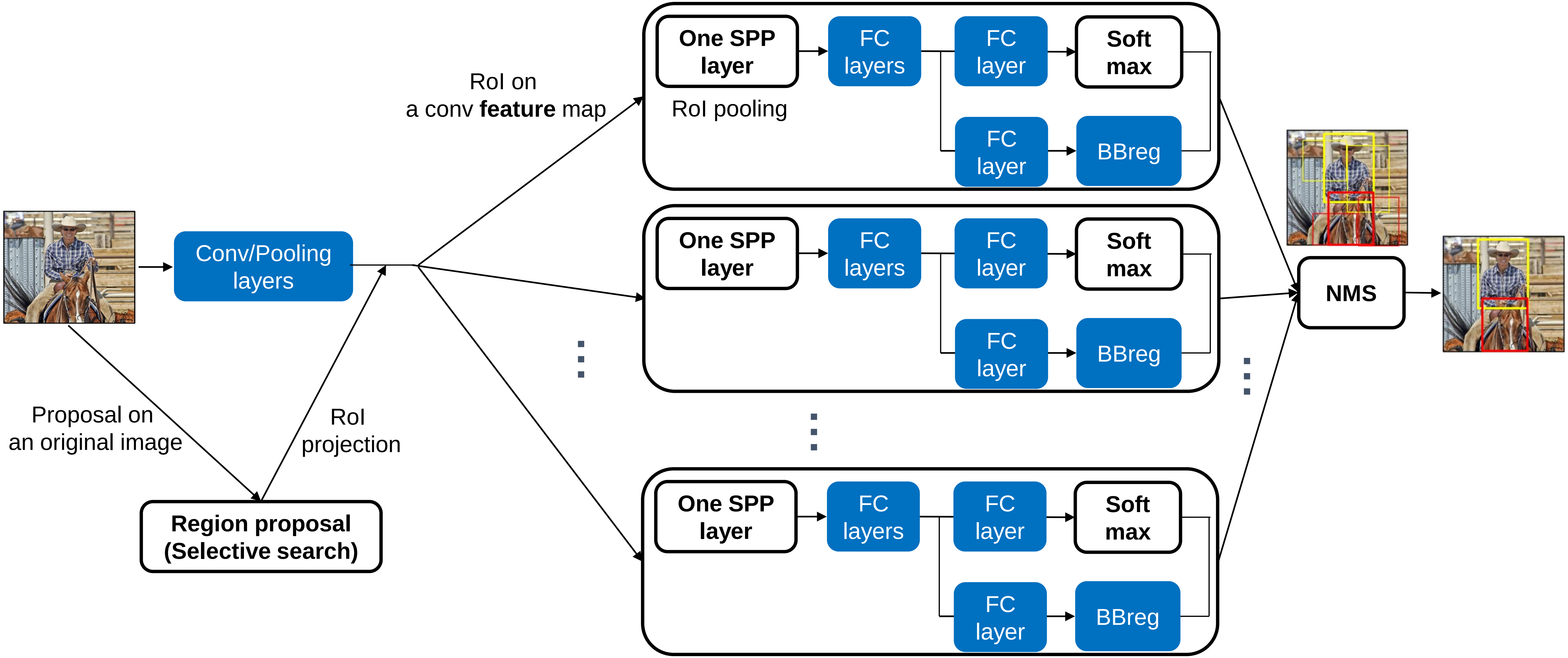
[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

Summary – SPPnet



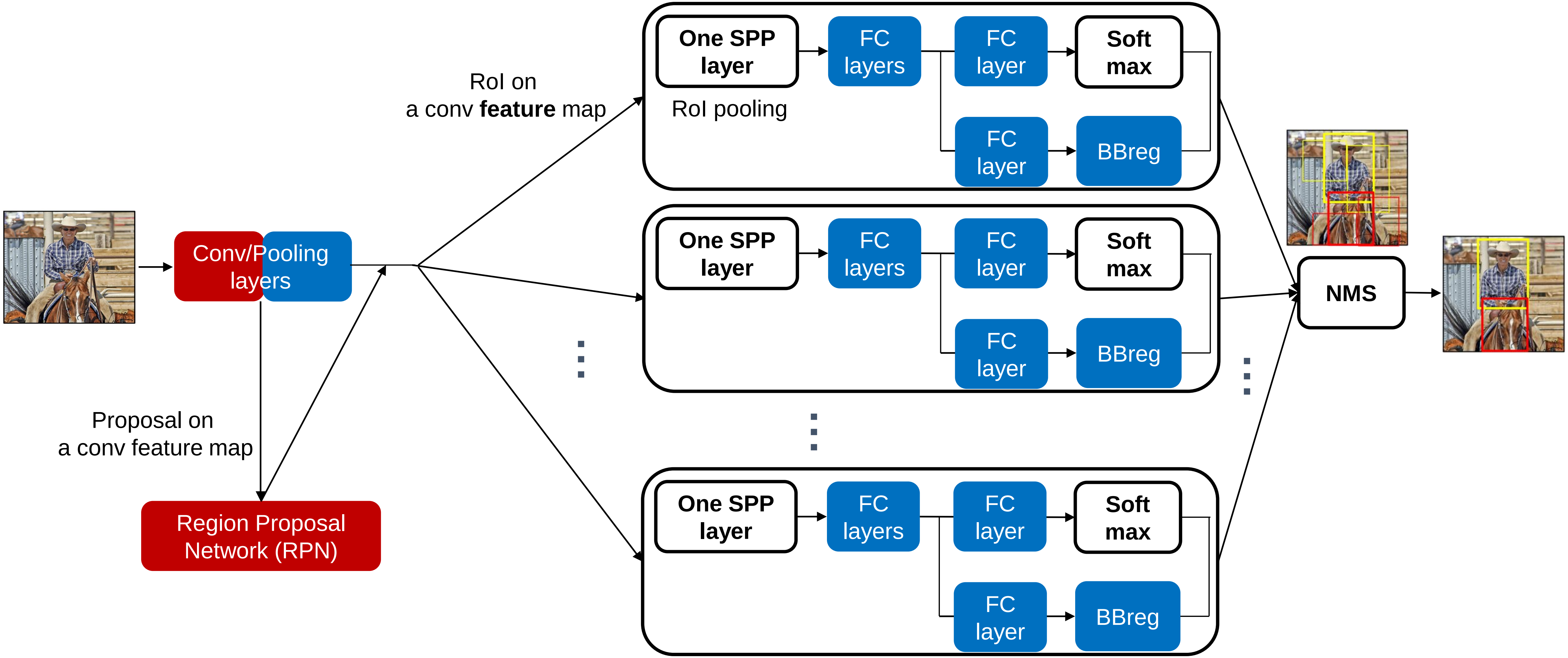
[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

Summary – Fast R-CNN



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

Summary – Faster R-CNN



[The figures are from R. Girshick et al., "Rich feature hierarchies for accurate object detection and semantic segmentation."]

Stepping Forward...

- Mask R-CNN [CVPR'17]
  - https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf

- Feature Pyramid Network (FPN) [CVPR'17]
  - https://openaccess.thecvf.com/content_cvpr_2017/papers/Lin_Feature_Pyramid_Networks_CVPR_2017_paper.pdf

Thanks!