EECS 498/598: Deep Learning
Lecture 3. CNNs for Computer Vision: Case Studies

Honglak Lee

02/01/2019

Outline

- **Basic Application Principles of NNs in Supervised Learning**
- Case study 1: Image Classification
- Case study 2: Object Detection
- Case study 3: Segmentation
- Case study 4: Pose Estimation
- Case study 5: Style Transfer

Basic Application Principles of NNs in Supervised Learning

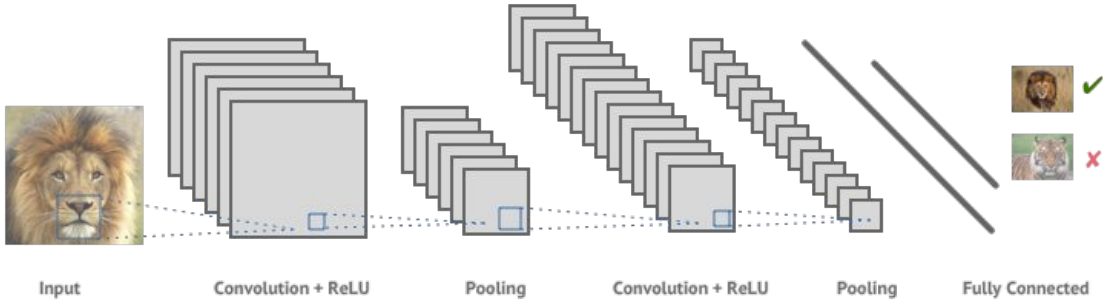
- Training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$

Basic Application Principles of NNs in Supervised Learning

- Training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$
- Neural network as function approximator $\hat{y} = f(x; \theta)$

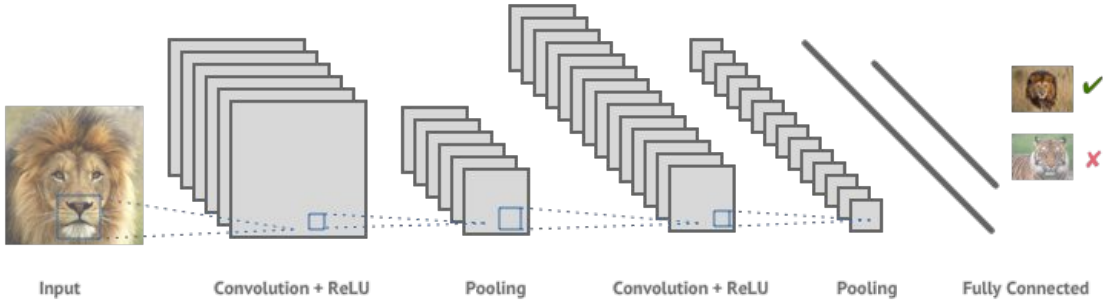
Basic Application Principles of NNs in Supervised Learning

- Training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$
- Neural network as function approximator $\hat{y} = f(x; \theta)$
  - Define architecture (by considering input/output)



Basic Application Principles of NNs in Supervised Learning

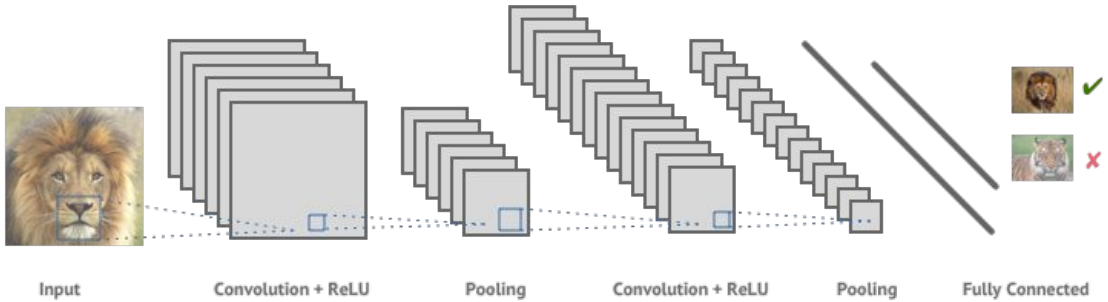
- Training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$
- Neural network as function approximator $\hat{y} = f(x; \theta)$
  - Define architecture (by considering input/output)



- Define loss function $\mathcal{L}(y, \hat{y})$

Basic Application Principles of NNs in Supervised Learning

- Training set $(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \cdots, (x^{(m)}, y^{(m)})$
- Neural network as function approximator $\hat{y} = f(x; \theta)$
  - Define architecture (by considering input/output)



- Define loss function $\mathcal{L}(y, \hat{y})$
- Objective $\min_{\theta} \sum_{i=1}^{m} \mathcal{L}(y^{(i)}, f(x^{(i)}; \theta))$ ← minimize via SGD

Outline

- Basic Application Principles of NNs in Supervised Learning
- **Case study 1: Image Classification**
- Case study 2: Object Detection
- Case study 3: Segmentation
- Case study 4: Pose Estimation
- Case study 5: Style Transfer

Object classification

- Given an input image, classify the object within it
- Input?

- Output? “Cat”
- Loss? Multi-class cross entropy

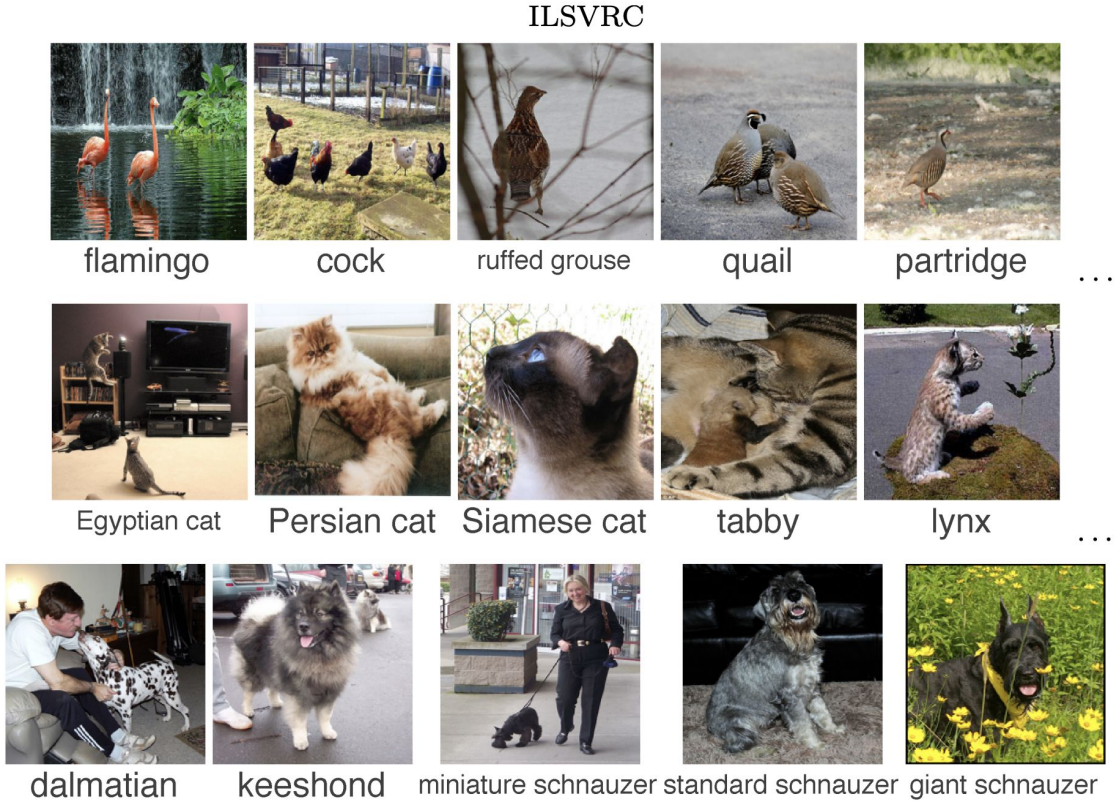
Object classification

- Given an input image, classify the object within it
- Input?

- Output? “Cat” (class probability: $\hat{y} = (\hat{p_1}, \hat{p_2}, \cdots, \hat{p_C})$)
- Loss? $\mathcal{L}(y, \hat{y}) = -y \log(\hat{y}) = -\sum_{c=1}^{C} p_c \log(\hat{p_c})$

Object classification

- Imagenet dataset
- Very large dataset of many object classes
- Enable learning models capable to identify many object features



Object Classification: Deeper is better

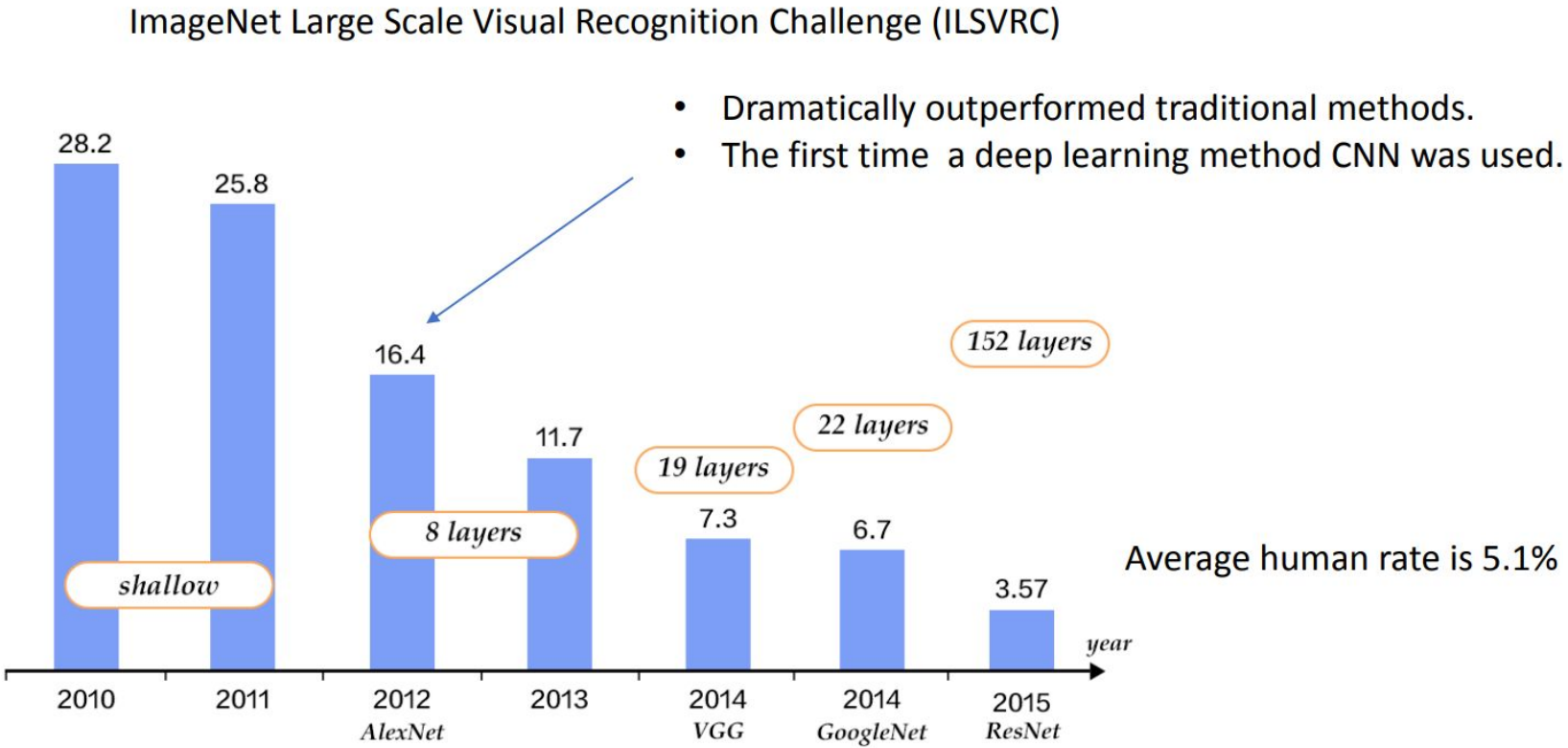
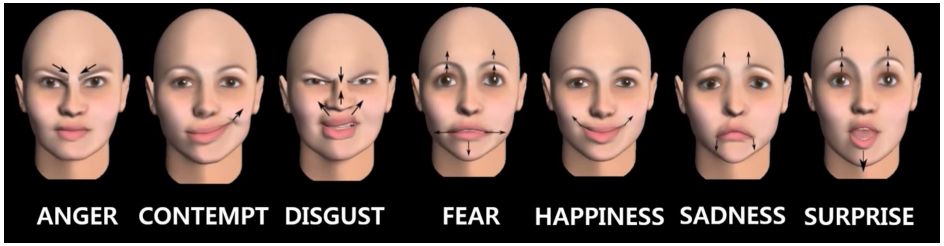


Figure source: http://www.rt-rk.uns.ac.rs/sites/default/files/materijali/predavanja/DL_5.pdf

Emotion classification

- Given an input human face, label the emotion
- Input?



Examples of emotion categories

- Output? “fear”
- Loss? Multi-class cross entropy

Figure from: https://www.youtube.com/watch?v=wlaR5F30hiU

Emotion Classification

Figure credit: https://github.com/oarriaga/face_classification

Emotion Classification

- Pipeline:
  - Input preprocessing (alignment, normalization, etc)
  - Given the preprocessed data, pass it to a CNN and get output
  - Output label with the most probability is the the assigned emotion

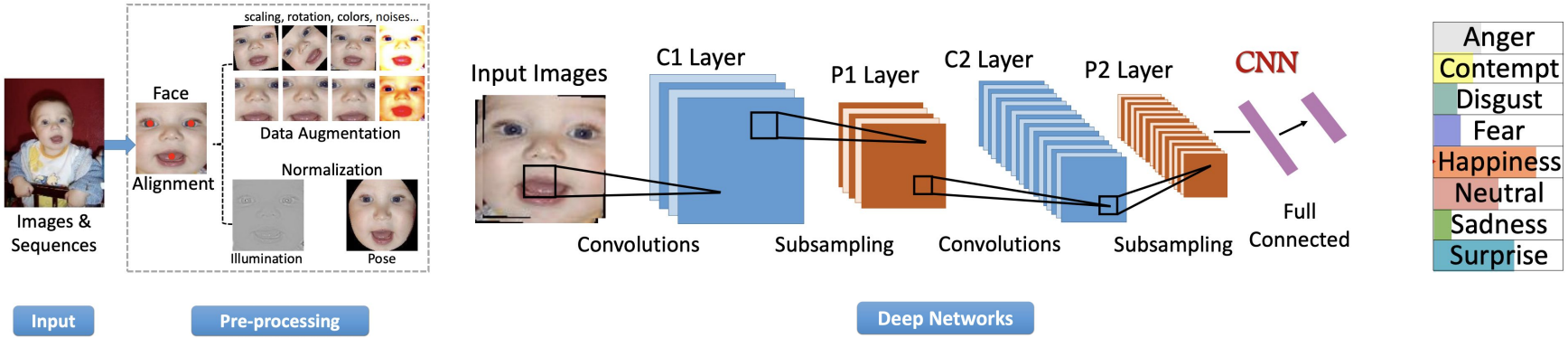


Figure credit: Shan Li and Weihong Deng. Deep Facial Expression Recognition: A Survey.

Food Category Classification

- Given an input image of food, classify it into one of the “pre-defined” categories
- Input? 
- Output? “chicken curry rice”
- Loss? Multi-class cross entropy

Food Category Classification

- Food 101 dataset:
  - 101 food categories with 101,000 images
  - (example categories: baby back ribs, chocolate cake, hot and sour soup, caesar, etc.)

https://www.vision.ee.ethz.ch/datasets_extra/food-101/
https://github.com/stratospark/food-101-keras

Food Attribute Classification

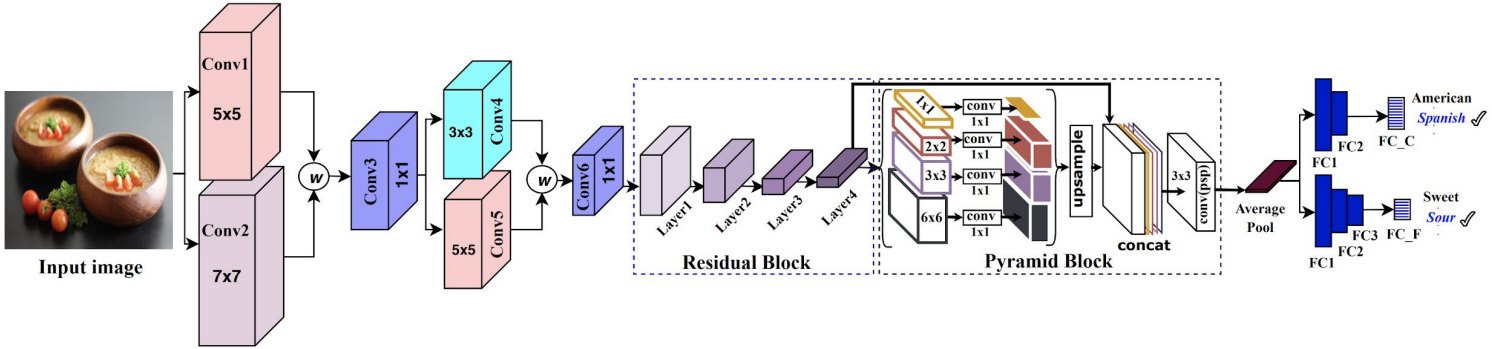
- Given an input image of food, specify attributes
- Input? 
- Output? “Cuisine: Indian”, “Taste: Meaty”
- Loss? Multi-class cross entropy per attribute

Food Attribute Classification

- The Yumly dataset

Figure credit: Sarker et. al. CuisineNet: Food Attributes Classification using Multi-scale Convolution Network. 2018

Food Attribute Classification



GT: *Cuisine*: American, *Flavor*: Sweet
PD: *Cuisine*: American, *Flavor*: Sweet

GT: *Cuisine*: Chinese, *Flavor*: Salty
PD: *Cuisine*: Chinese, *Flavor*: Salty

GT: *Cuisine*: Italian, *Flavor*: Meaty
PD: *Cuisine*: Italian, *Flavor*: Meaty

GT: *Cuisine*: Spanish, *Flavor*: Sour
PD: *Cuisine*: Spanish, *Flavor*: Sour

GT: *Cuisine*: Italian, *Flavor*: Sour
PD: *Cuisine*: Spanish, *Flavor*: Sour

GT: *Cuisine*: Thai, *Flavor*: Sweet
PD: *Cuisine*: Chinese, *Flavor*: Sweet

GT: *Cuisine*: Mexican, *Flavor*: Piquant
PD: *Cuisine*: Greek, *Flavor*: Meaty

GT: *Cuisine*: French, *Flavor*: Salty
PD: *Cuisine*: Spanish, *Flavor*: Meaty

Scene Classification

- Given an input image of a scene, label the place
- Input? 
- Output? “cafeteria”
- Loss? Multi-class cross entropy

Scene Classification

- The places2 dataset

Scene Classification

- Scene classification using VGG16

GT: cafeteria
top-1: cafeteria (0.179)
top-2: restaurant (0.167)
top-3: dining_hall (0.091)
top-4: coffee_shop (0.086)
top-5: restaurant_patio (0.080)

GT: classroom
top-1: locker_room (0.585)
top-2: lecture_room (0.135)
top-3: conference_center (0.061)
top-4: classroom (0.033)
top-5: elevator door (0.025)

GT: drugstore
top-1: supermarket (0.286)
top-2: hardware_store (0.248)
top-3: drugstore (0.120)
top-4: department_store (0.087)
top-5: pharmacy (0.052)

GT: natural canal
top-1: swamp (0.529)
top-2: marsh (0.232)
top-3: natural canal (0.063)
top-4: lagoon (0.047)
top-5: rainforest (0.029)

GT: creek
top-1: forest broadleaf (0.307)
top-2: forest_path (0.208)
top-3: creek (0.086)
top-4: rainforest (0.076)
top-5: cemetery (0.049)

GT: greenhouse indoor
top-1: greenhouse indoor (0.479)
top-2: greenhouse outdoor (0.055)
top-3: botanical_garden (0.044)
top-4: assembly_line (0.025)
top-5: vegetable_garden (0.022)

GT: chalet
top-1: ski_resort (0.141)
top-2: ice_floe (0.129)
top-3: igloo (0.114)
top-4: balcony exterior (0.103)
top-5: courtyard (0.083)

GT: crosswalk
top-1: crosswalk (0.720)
top-2: plaza (0.060)
top-3: street (0.055)
top-4: shopping_mall indoor (0.039)
top-5: bazaar outdoor (0.021)

GT: market outdoor
top-1: promenade (0.569)
top-2: bazaar outdoor (0.137)
top-3: boardwalk (0.118)
top-4: market outdoor (0.074)
top-5: flea_market indoor (0.029)

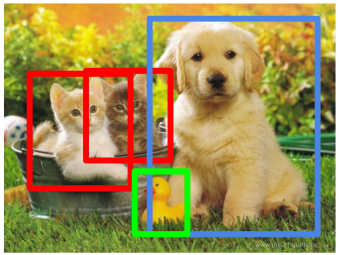
Figure credit: https://github.com/CSAILVision/places365

Outline

- Basic Application Principles of NNs in Supervised Learning
- Case study 1: Image Classification
- **Case study 2: Object Detection**
- Case study 3: Segmentation
- Case study 4: Pose Estimation
- Case study 5: Style Transfer

Object Detection

- Given an image, detect all objects and classify
- Input?
- Output?



CAT, DOG, DUCK

- Loss? Error in bounding box position?, bounding box scale?, class label?

Object Detection

Classes = [cat, dog, duck]

Cat ? NO

Dog ? NO

Duck? NO

Slide credit: Amaia Salvador

Object Detection

Classes = [cat, dog, duck]

Cat ? NO

Dog ? NO

Duck? NO

Slide credit: Amaia Salvador

Object Detection

Classes = [cat, dog, duck]

Cat ? YES

Dog ? NO

Duck? NO

Slide credit: Amaia Salvador

Object Detection

Classes = [cat, dog, duck]

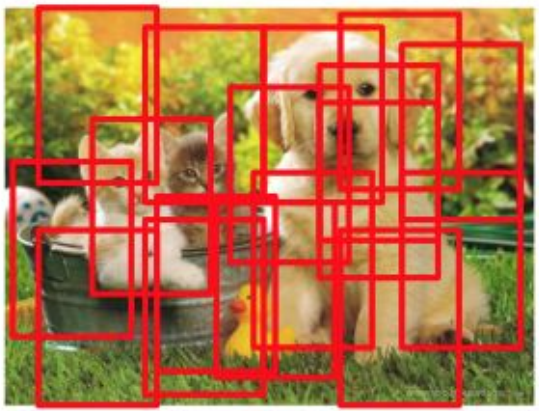
Cat ? NO

Dog ? NO

Duck? NO

Slide credit: Amaia Salvador

Object Detection

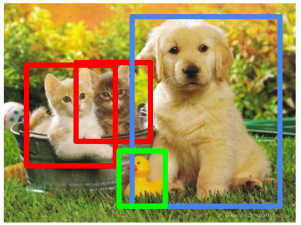


Problem:
Too many positions & scales to test

Solution: If your classifier is fast enough, go for it

Slide credit: Amaia Salvador

Object Detection

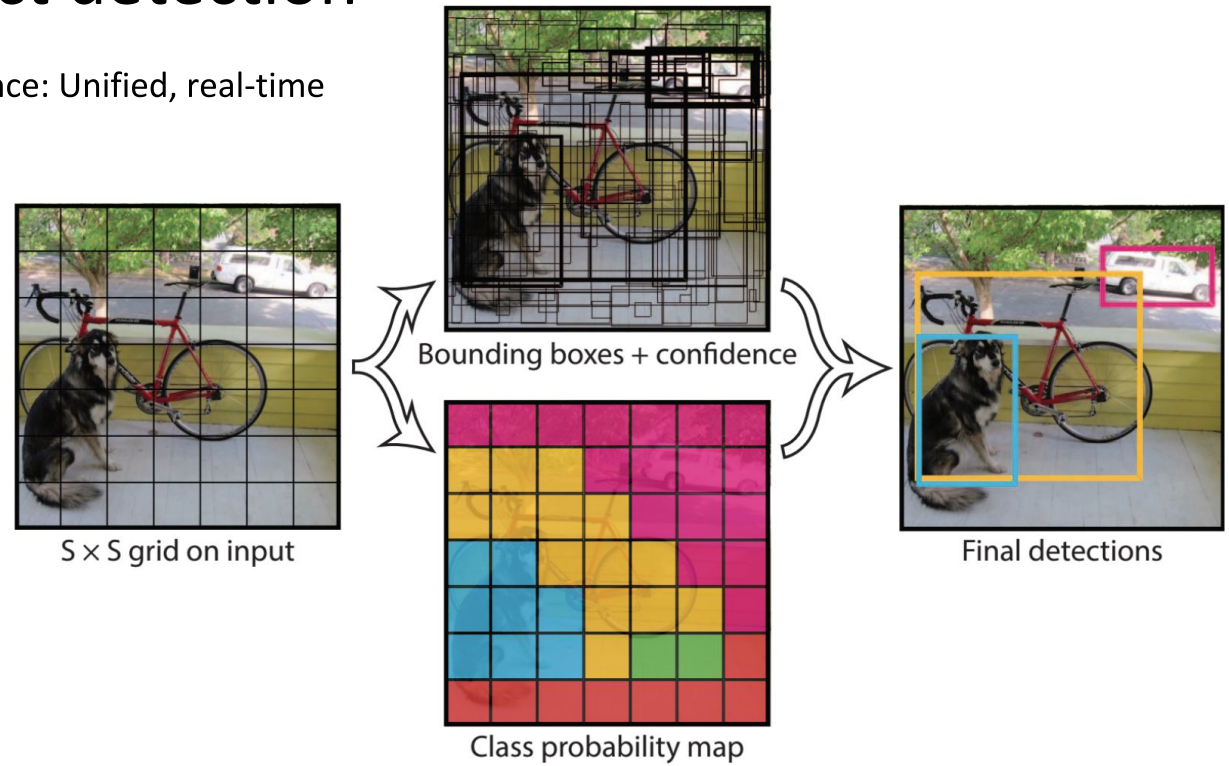


- Two sub-problems:
  - Object locations/size: localize each object with a tight bounding box in the image
  - Object classification: classify the object in each bounding box
- Typically solved by **“proposing candidate bounding boxes”** and **“classifying them”**
- Two main families:
  - A grid in the image where each cell is a proposal (YOLO)
  - Region proposal (Fast RCNN, Faster RCNN)

YOLO

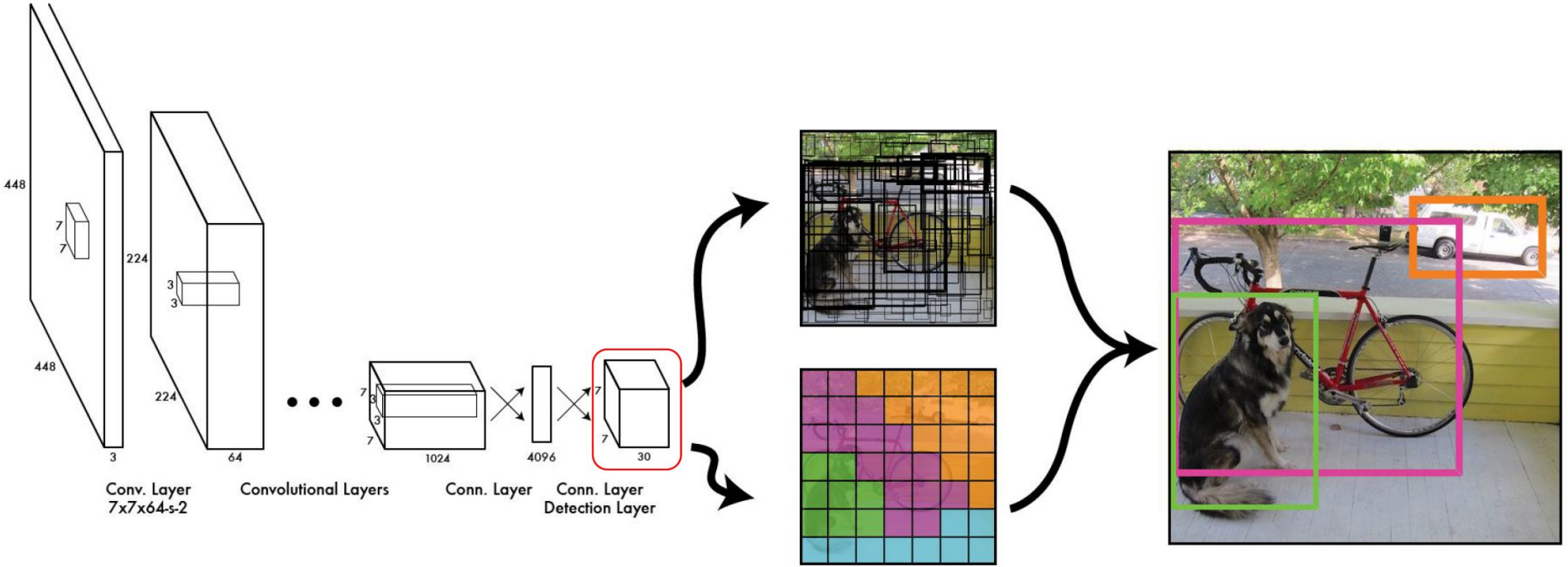
- Proposal-free object detection

Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." CVPR (2016)



Related work:
SSD: Liu et al. *ECCV 2016*
OverFeat: Sermanet et al., (2013)

YOLO

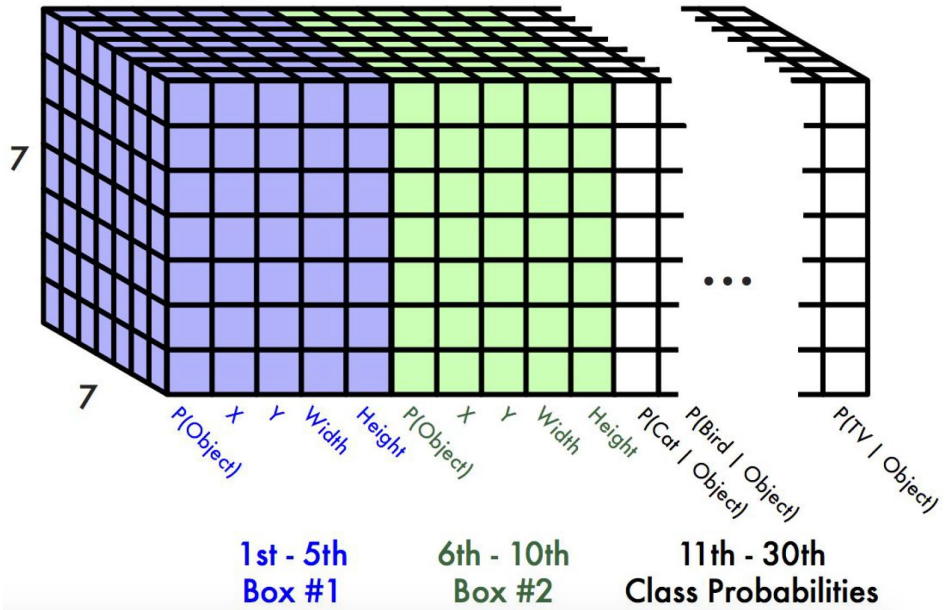


Redmon, Joseph, et al. "You only look once: Unified, real-time object detection." CVPR (2016)

YOLO

- Each cell predicts: B boxes
  - For each bounding box:
    - 4 coordinates (x, y, w, h)
    - 1 confidence value
  - Probabilities of classes
- For Pascal VOC:
  - 7x7 grid
  - 2 bounding boxes / cell
  - 20 classes

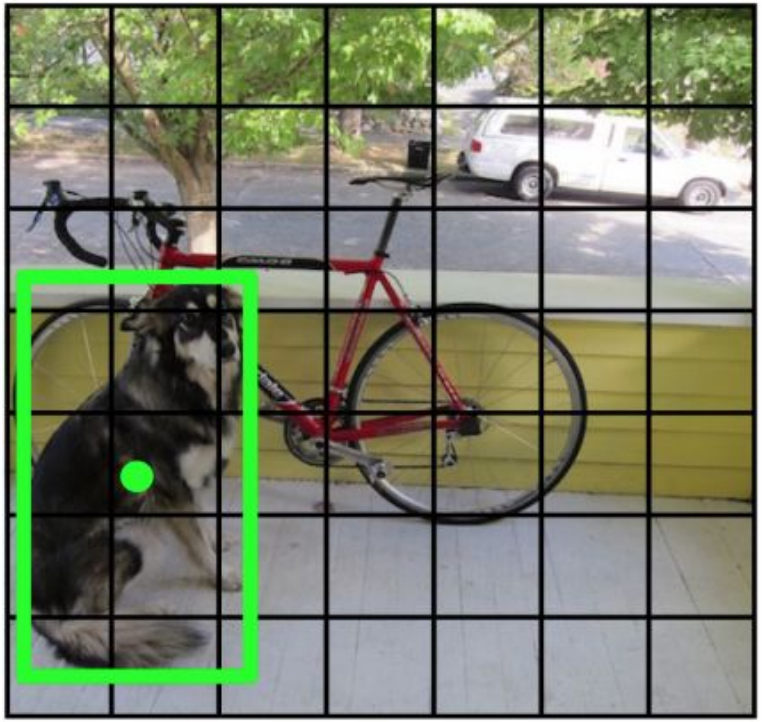
(B=2 in this example)



- 7 x 7 x (2 x 5 + 20) = 7 x 7 x 30 tensor = 1470 outputs

Slide credit: Amaia Salvador

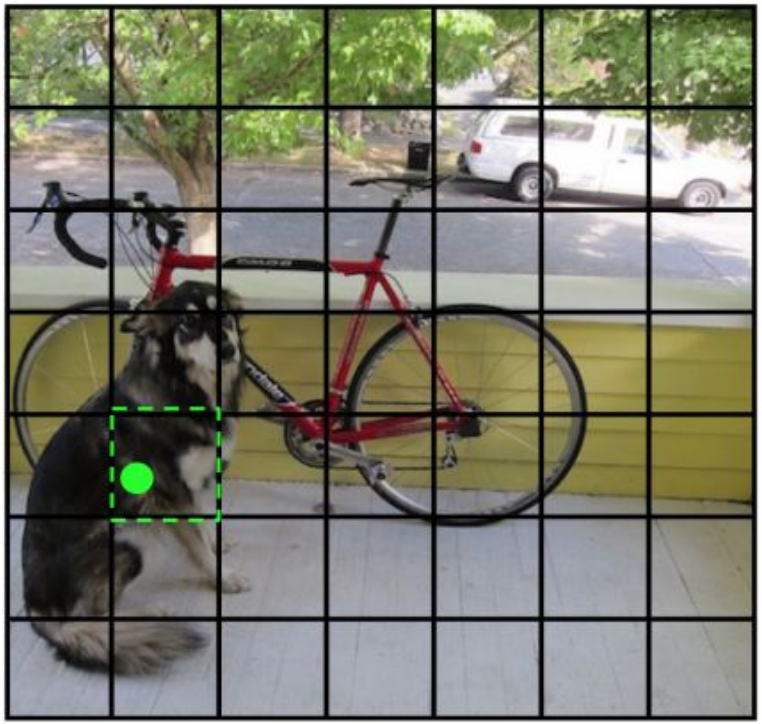
YOLO



For training, each ground truth bounding box is matched into the right cell

Slide credit: YOLO Presentation @ CVPR 2016

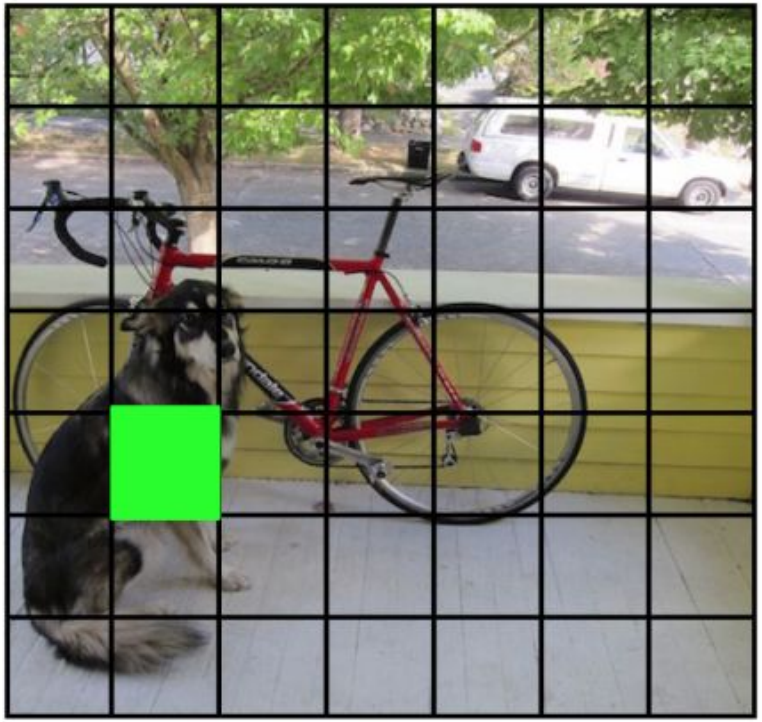
YOLO



For training, each ground truth bounding box is matched into the right cell

Slide credit: YOLO Presentation @ CVPR 2016

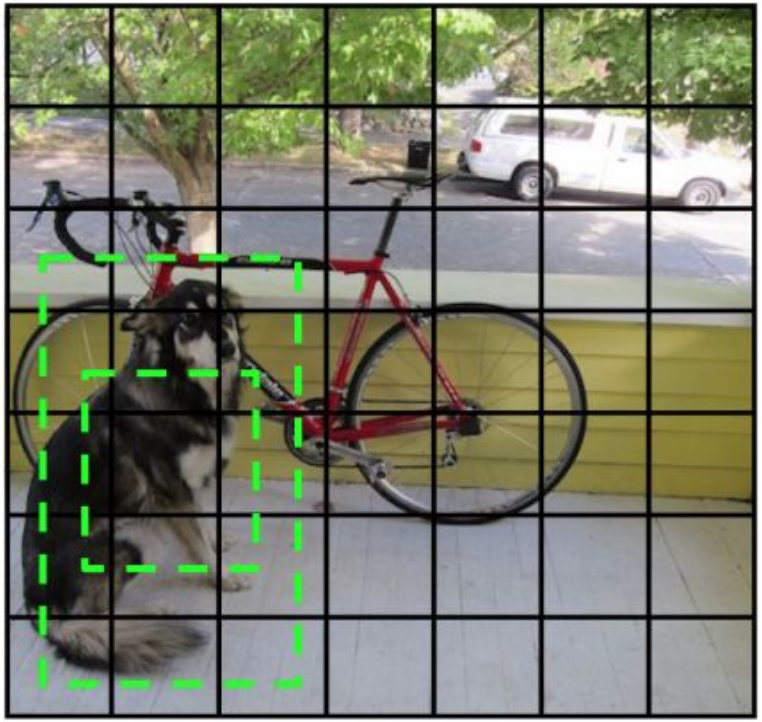
YOLO



Optimize class prediction in that cell:
dog: 1, cat: 0, bike: 0, ...

Slide credit: YOLO Presentation @ CVPR 2016

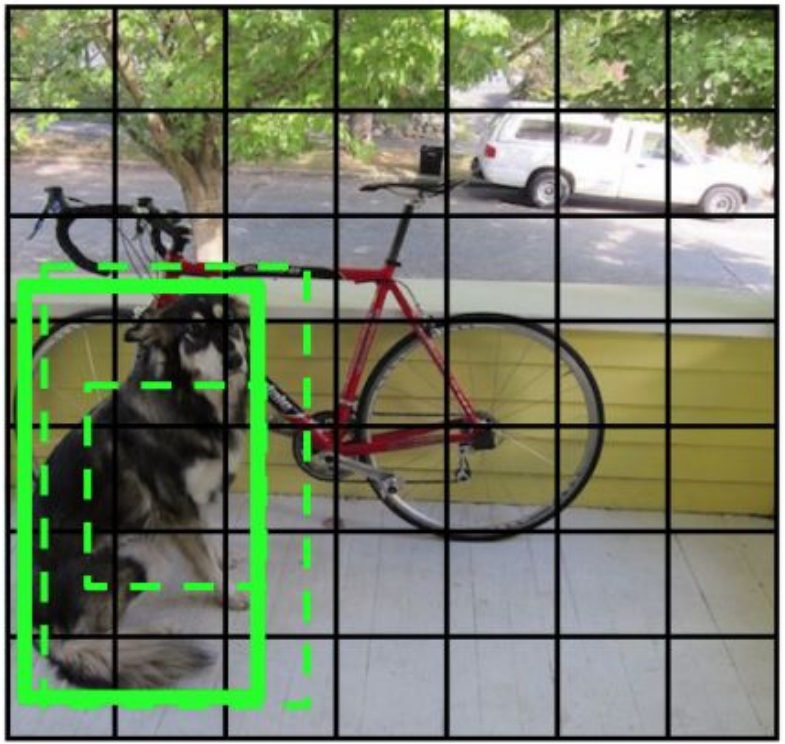
YOLO



Predicted boxes for this cell

Slide credit: YOLO Presentation @ CVPR 2016

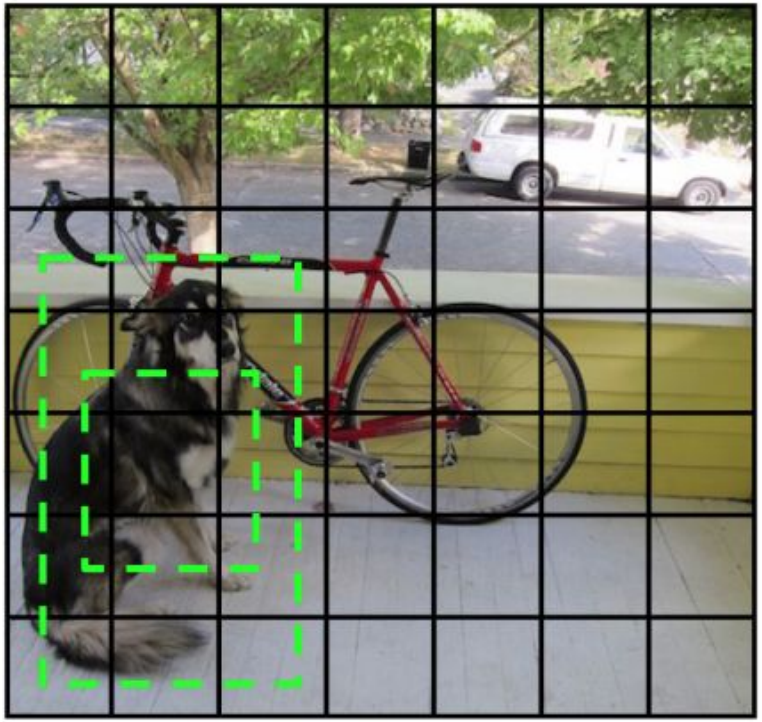
YOLO



Find the best one wrt ground truth bounding box, optimize it (i.e. adjust its coordinates to be closer to the ground truth's coordinates)

Slide credit: YOLO Presentation @ CVPR 2016

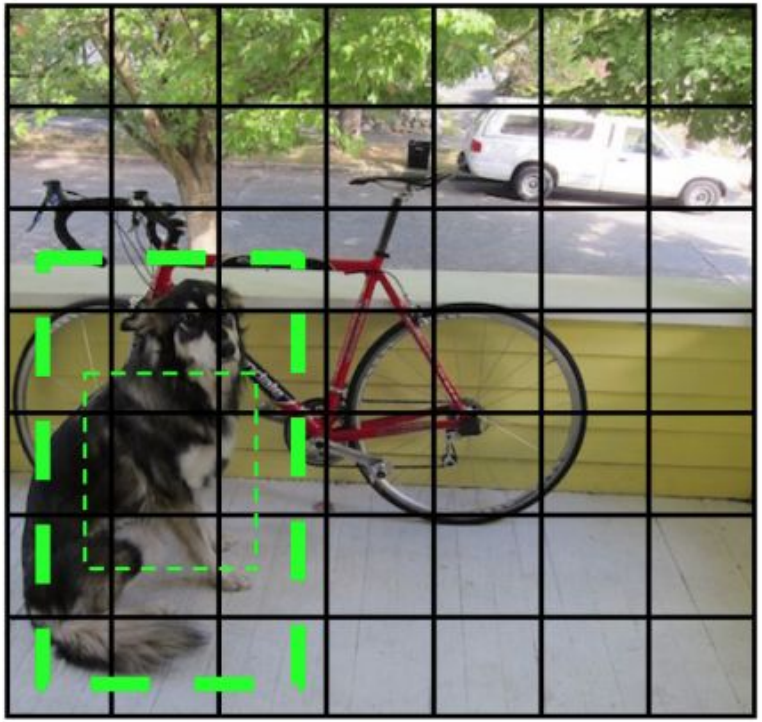
YOLO



Increase matched box's confidence, decrease non-matched boxes confidence

Slide credit: YOLO Presentation @ CVPR 2016

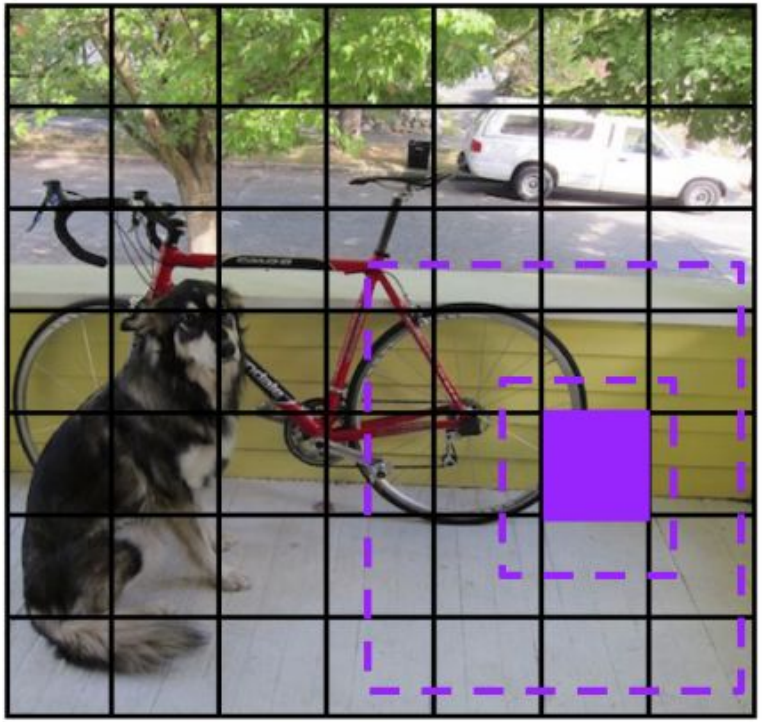
YOLO



Increase matched box's confidence, decrease non-matched boxes confidence

Slide credit: YOLO Presentation @ CVPR 2016

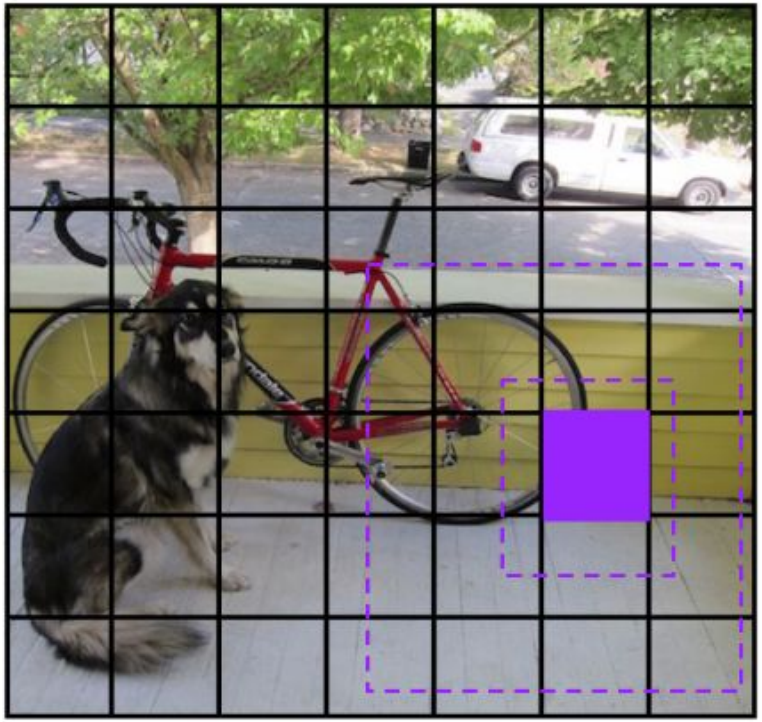
YOLO



For cells with no ground truth detections, confidences of all predicted boxes are decreased

Slide credit: YOLO Presentation @ CVPR 2016

YOLO



For cells with no ground truth detections:

- Confidences of all predicted boxes are decreased
- Class probabilities are not adjusted

Slide credit: YOLO Presentation @ CVPR 2016

YOLO

= 1 if box *j* and cell *i* are matched together, 0 otherwise

Bounding box coordinate regression

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

Bounding box score prediction

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

= 1 if box *j* and cell *i* are NOT matched together

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

Class score prediction

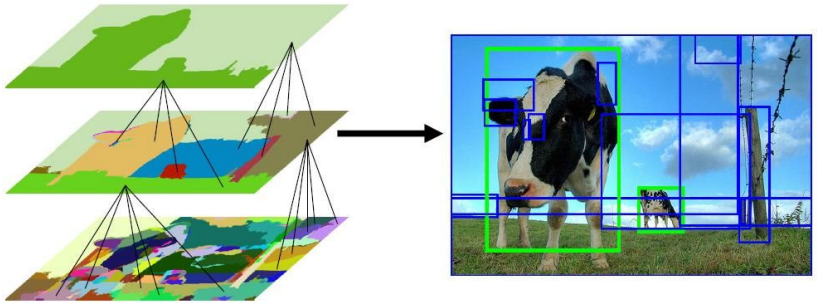
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

= 1 if cell *i* has an object present

Slide credit: YOLO Presentation @ CVPR 2016

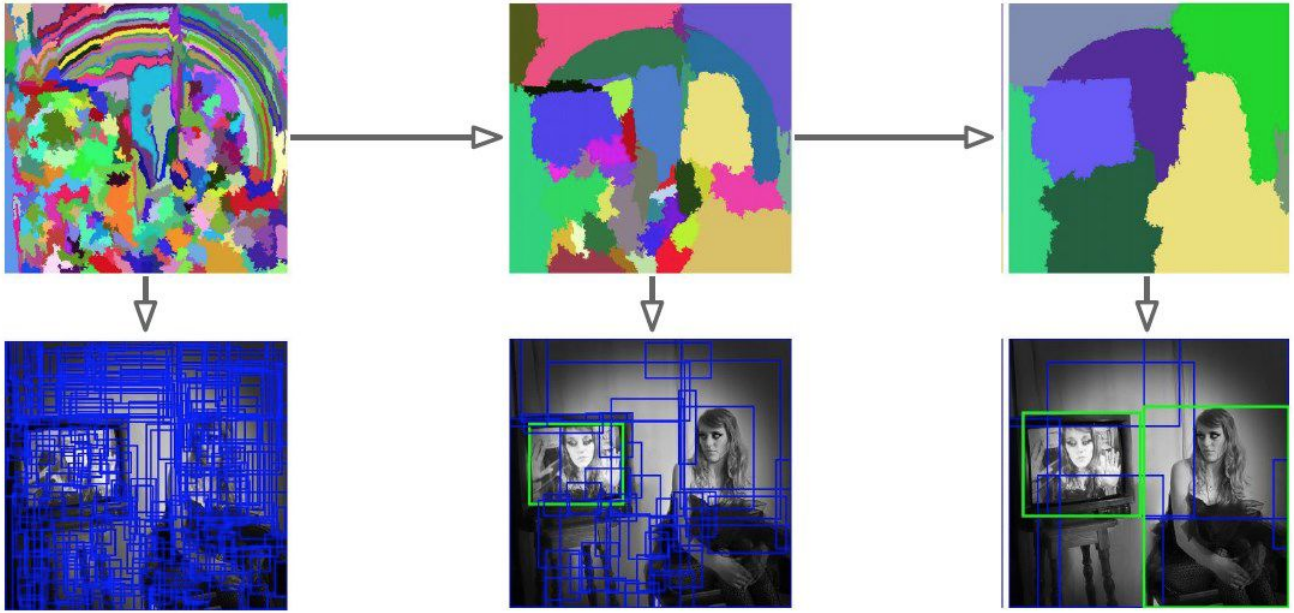
Bounding Box Proposals

- Instead of having a predefined set of box proposals, find them on the image:
  - Selective Search (from pixels, not learnt)
  - Region Proposal Network (RPN, learnt)



Selective Search

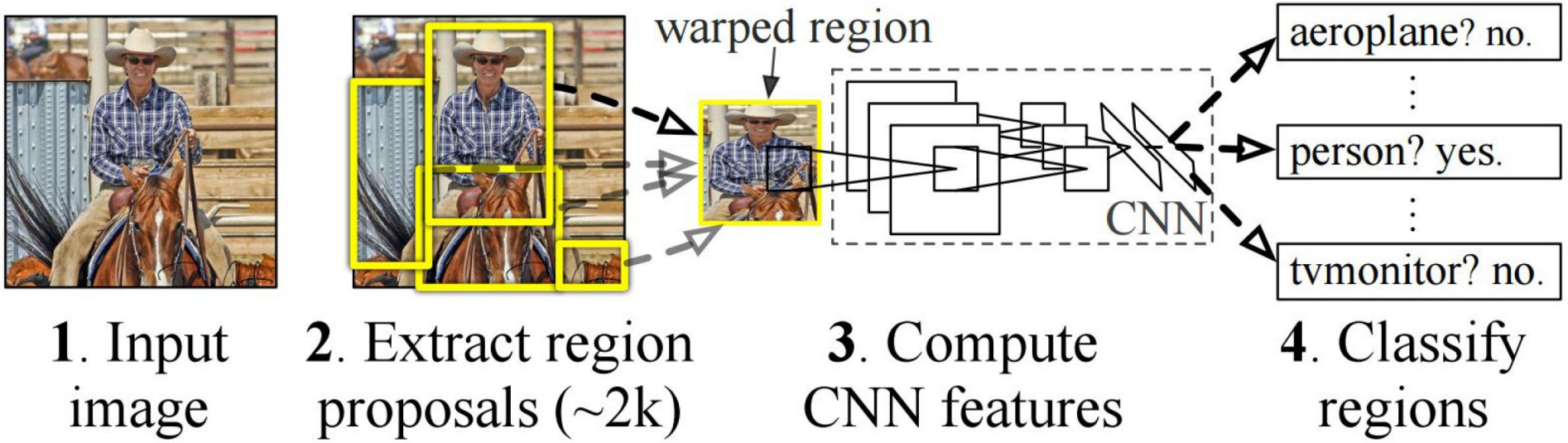
- bottom up segmentation, merging regions at multiple scales, convert regions to boxes



Uijlings, J. R., Van De Sande, K. E., Gevers, T., & Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, *104*(2), 154-171.

RCNN

For each region proposal, use CNN to predict the categories (with confidence)

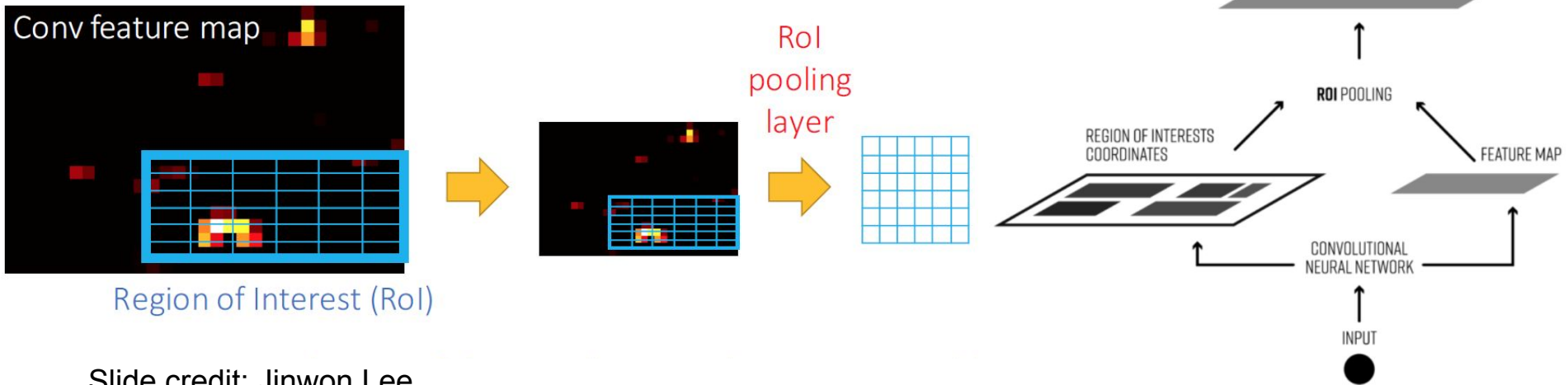


Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 580-587).

Making RCNN faster via RoI Pooling

Main idea: calculate convolutional feature map only once and reuse it!

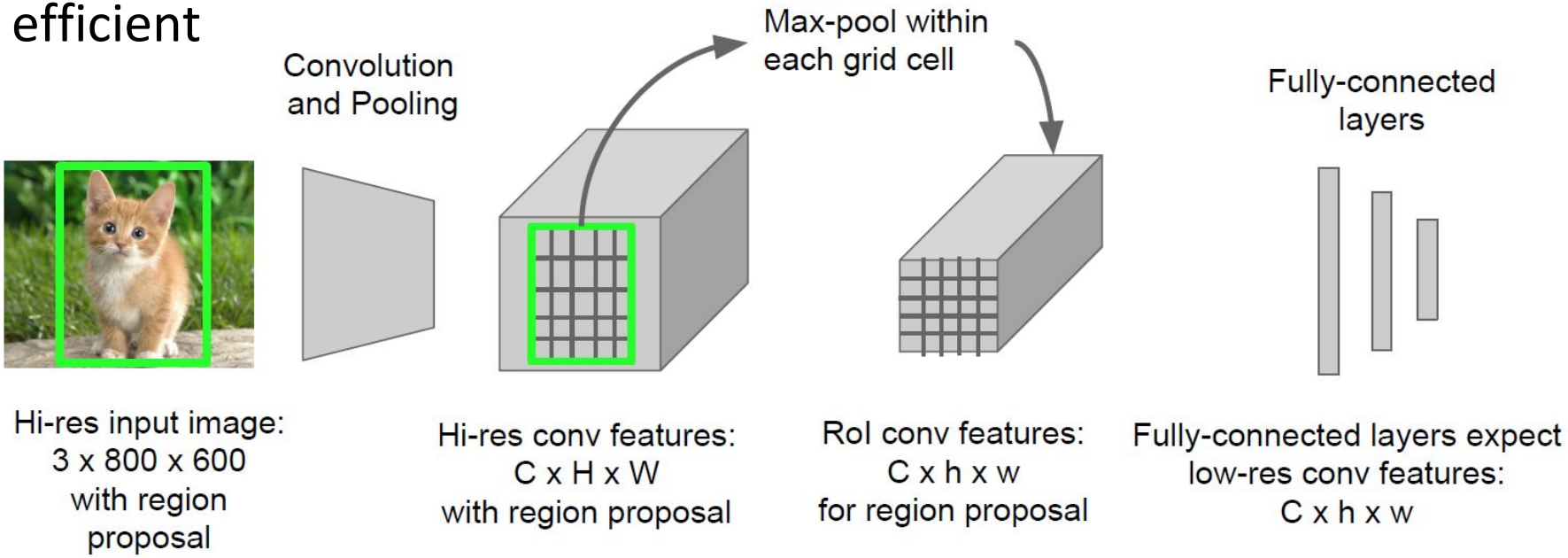
- RoI in conv feature map : 21x14⟶ 3x2 max pooling with stride(3, 2) ⟶ output : 7x7
- RoI in conv feature map : 35x42⟶ 5x6 max pooling with stride(5, 6) ⟶ output : 7x7



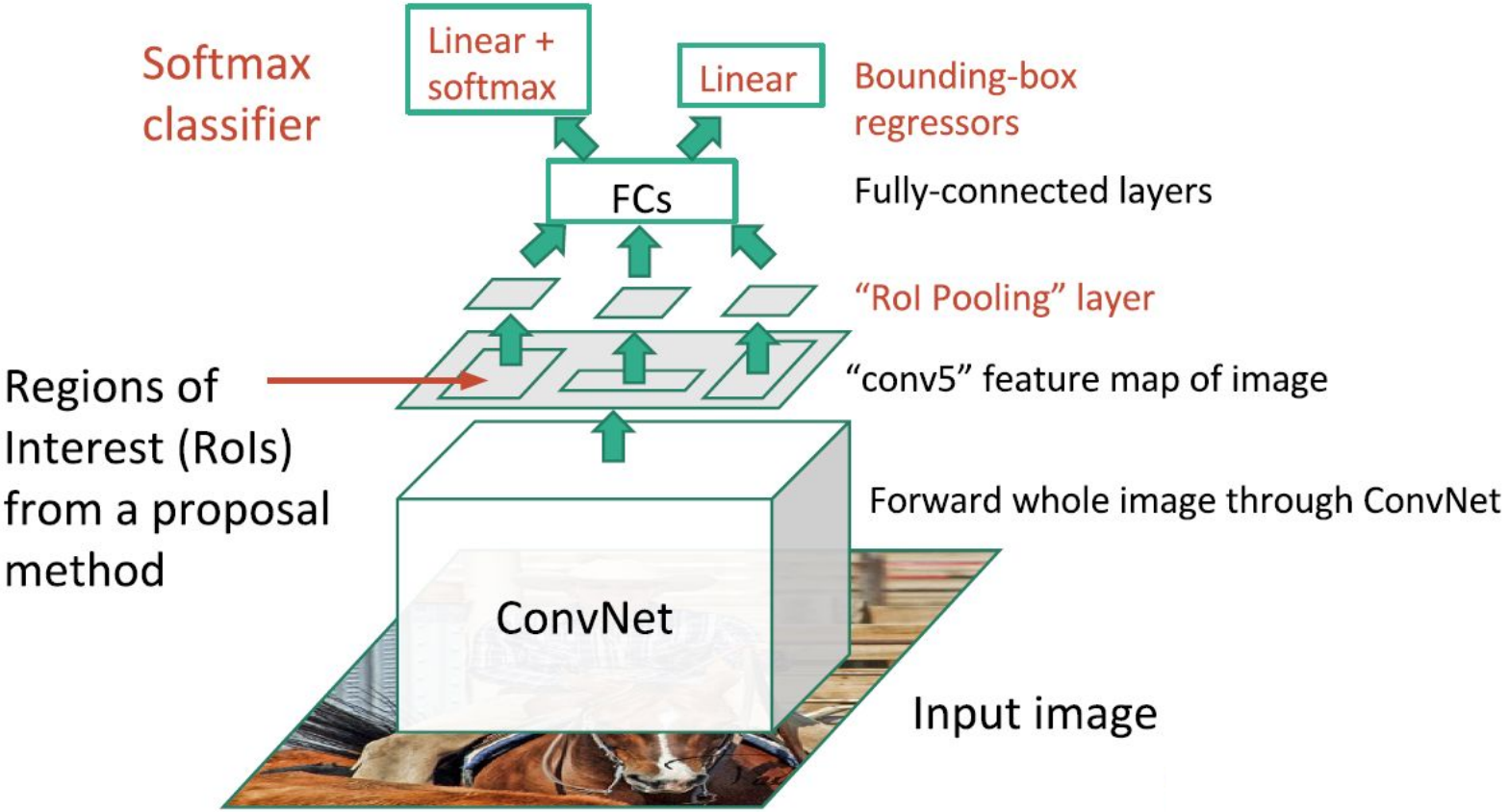
Slide credit: Jinwon Lee

RoI Pooling

- Input: convolutional map + regions of interest
- Output: tensor of boxes Nx7x7xdepth
- Allows to propagate gradient only on interesting regions, and efficient



Fast RCNN



Slide credit: Jinwon Lee

Fast RCNN

- Takes an input and a set of bounding boxes
- Generate convolutional feature maps
- For each bbox, get a fixed-length feature vector from RoI pooling layer
- Outputs have two information
  - K+1 class labels
  - Bounding box locations
- Loss function

True box coordinates

Predicted box coordinates

$$L(p, u, t^u, v) = L_{\text{cls}}(p, u) + \lambda[u \geq 1] L_{\text{loc}}(t^u, v)$$

Predicted class scores

True class scores

Log loss

Smooth L1 loss

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x,y,w,h}\}} \text{smooth}_{L_1}(t_i^u - v_i), \qquad (2)$$

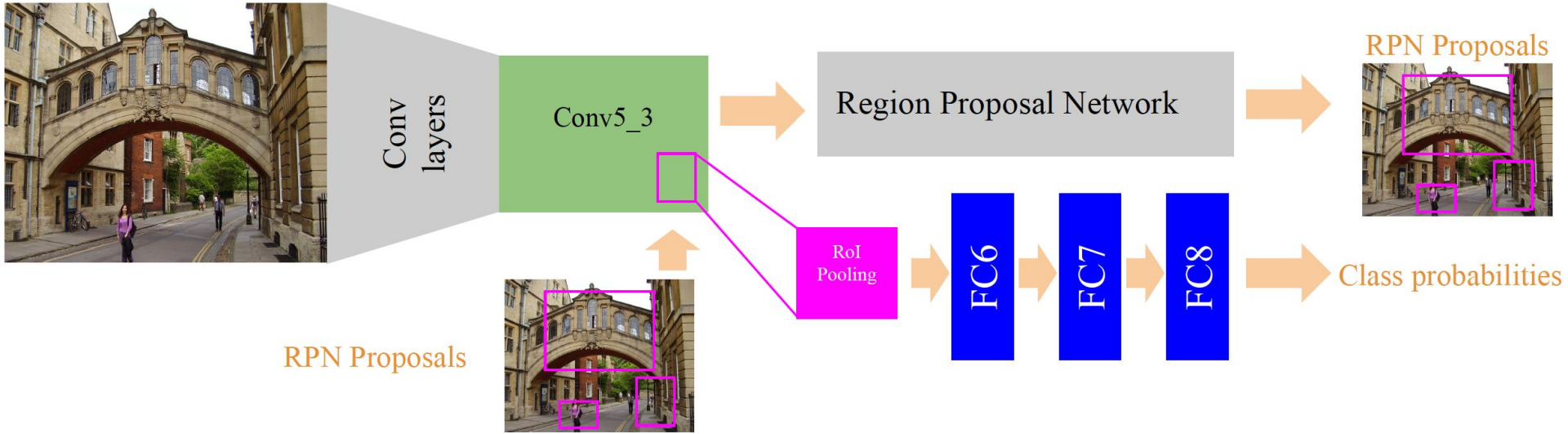
in which

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise,} \end{cases} \qquad (3)$$

Slide credit: Jinwon Lee

Faster RCNN

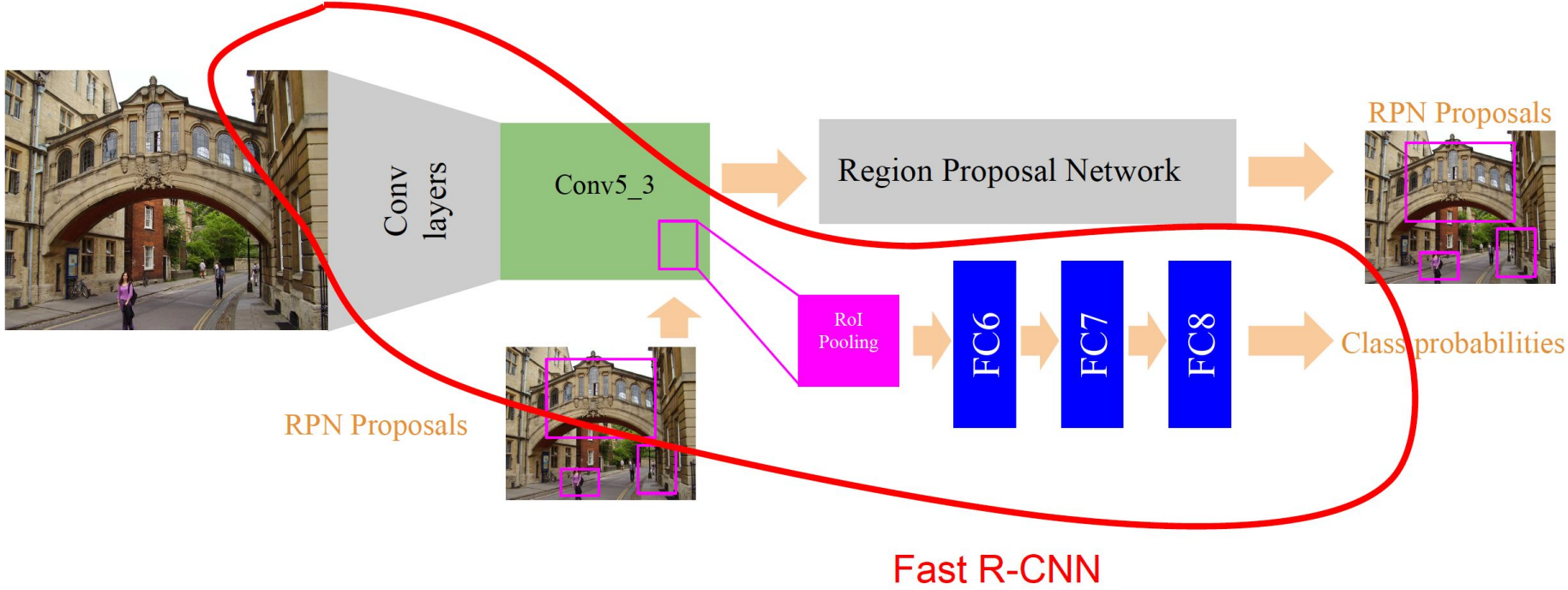
- Learn proposals end-to-end sharing parameters with the classification network



Slide credit: Amaia Salvador

Faster RCNN

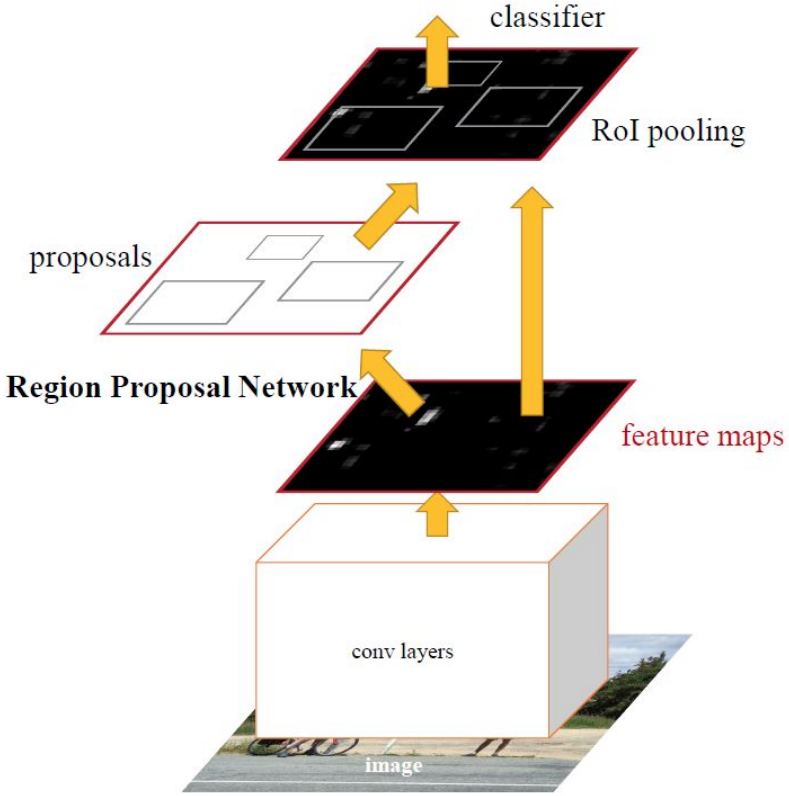
- Learn proposals end-to-end sharing parameters with the classification network



Slide credit: Amaia Salvador

Faster RCNN

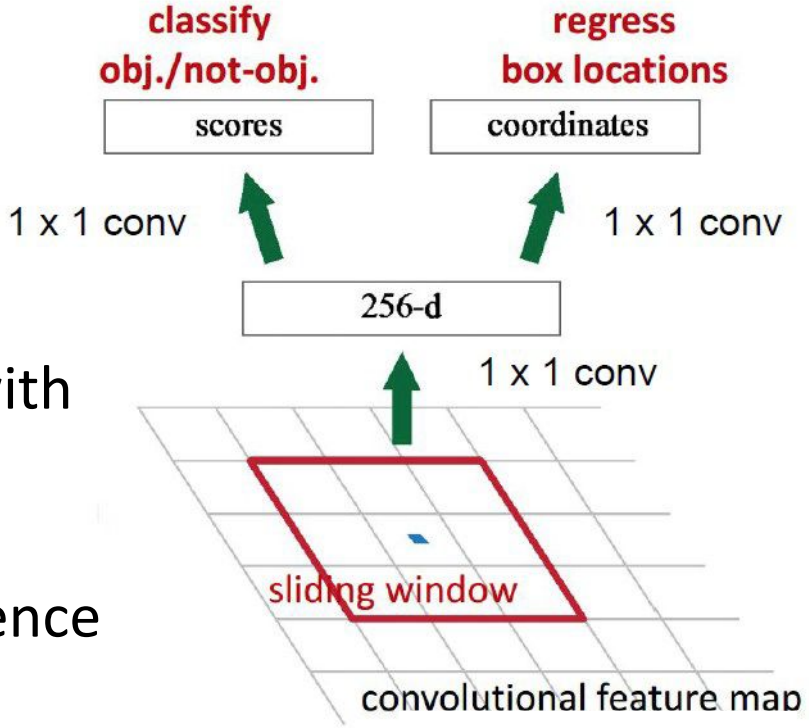
- Insert a Region Proposal Network (RPN) after the last convolutional layer!
- RPN trained to produce region proposals directly; no need for external region proposals
- After RPN, use RoI Pooling and an upstream classifier and bbox regressor



Slide credit: Jinwon Lee

RPN

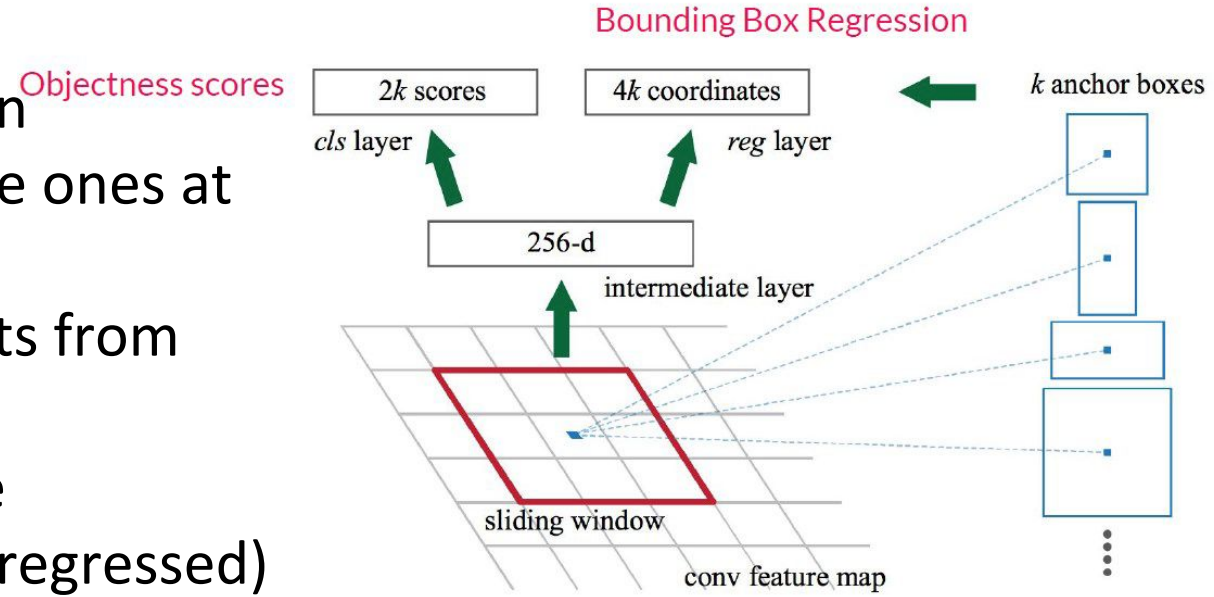
- Slide a small window on the feature map
- Build a small network for
  - Classifying object or not-object
  - Regressing bbox locations
- Position of the sliding window provides localization information with reference to the image
- Box regression provides finer localization information with reference to this sliding window



Slide credit: Jinwon Lee

RPN

- Use k anchor boxes at each location
- Anchors are translation invariant: use the same ones at every location
- Regression gives offsets from anchor boxes
- Classification gives the probability that each (regressed) anchor shows an object



Slide credit: Jinwon Lee

RPN

i = anchor index in minibatch

Log loss

Ground truth objectness label

Smooth L1 loss

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

Predicted probability of being an object for anchor i

Coordinates of the predicted bounding box for anchor i

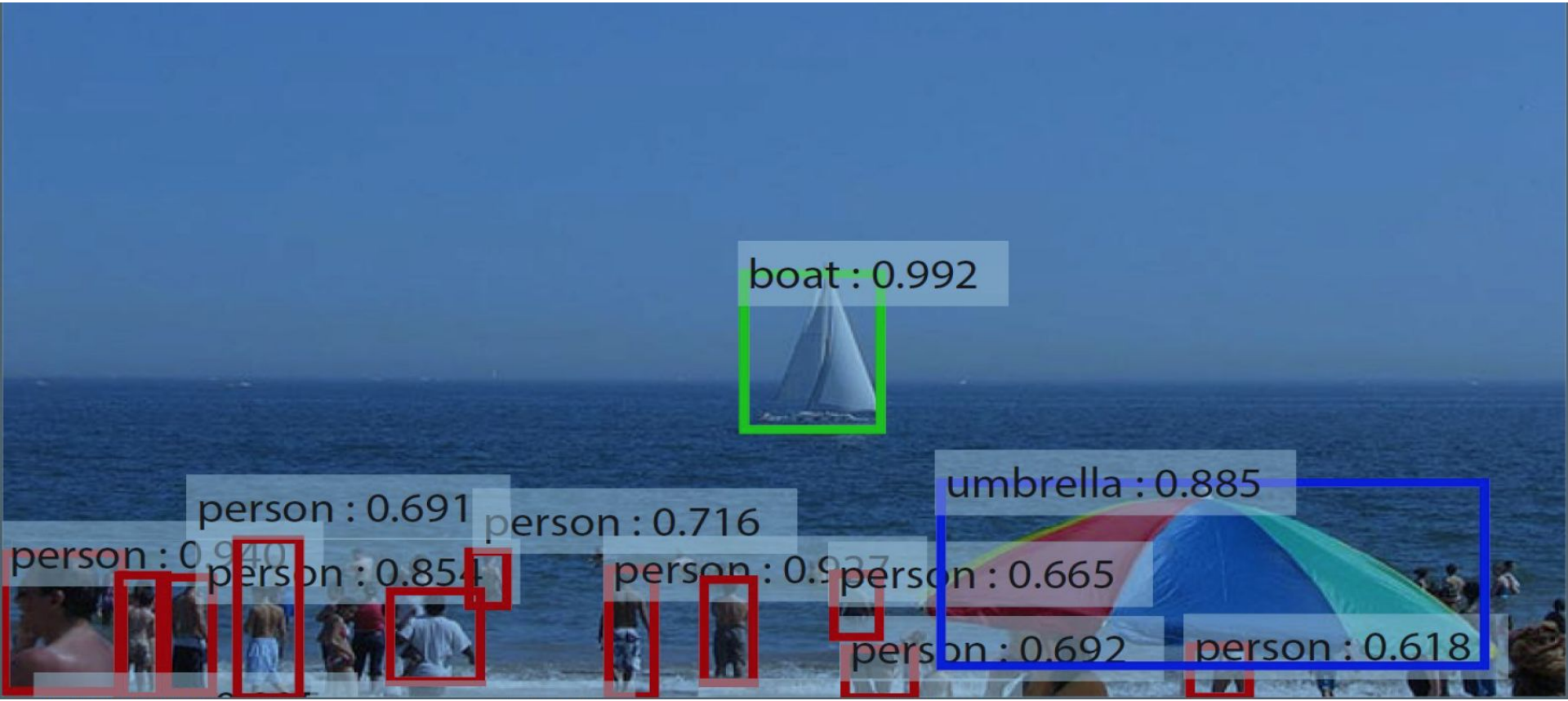
True box coordinates

In practice $\lambda = 10$, so that both terms are roughly equally balanced

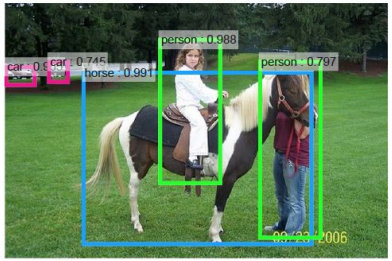
N_{cls} = Number of anchors in minibatch (~ 256)
N_{reg} = Number of anchor locations (~ 2400)

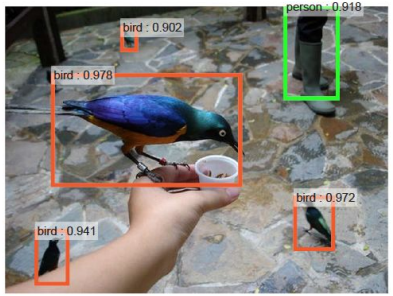
Slide credit: Jinwon Lee

Faster RCNN

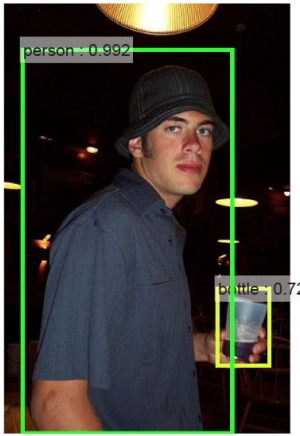

boat : 0.992
person : 0.691
person : 0.716
umbrella : 0.885
person : 0.665
person : 0.692
person : 0.618

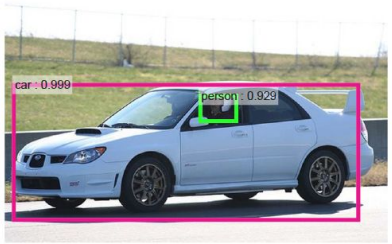
Faster RCNN

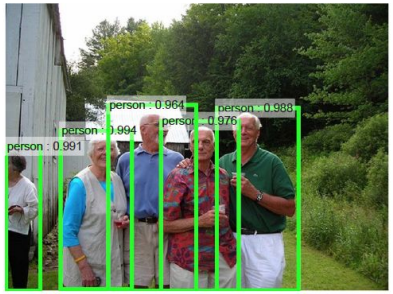

person : 0.988
horse : 0.991
person : 0.797

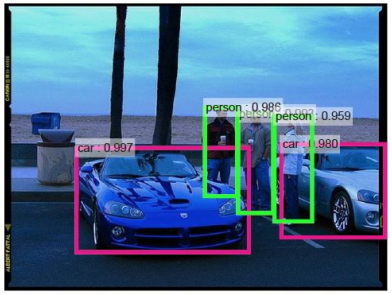

person : 0.918
bird : 0.902
bird : 0.978
bird : 0.972
bird : 0.941

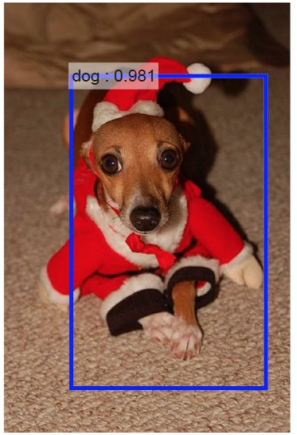

cow : 0.995
cow : 0.998

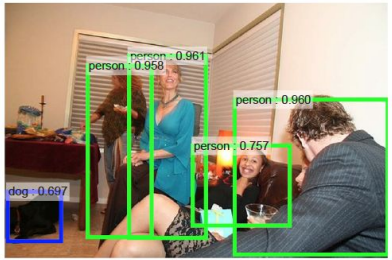

person : 0.992


car : 0.999
person : 0.929

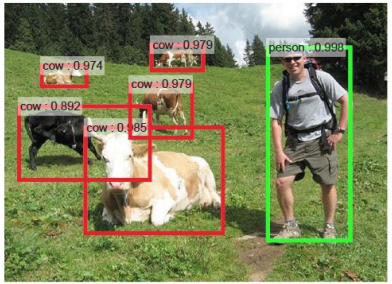

person : 0.964
person : 0.988
person : 0.994
person : 0.991


person : 0.986
person : 0.959
car : 0.997
car : 0.980

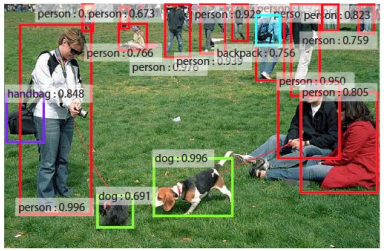

dog : 0.981


person : 0.961
person : 0.958
person : 0.960
person : 0.757
dog : 0.697

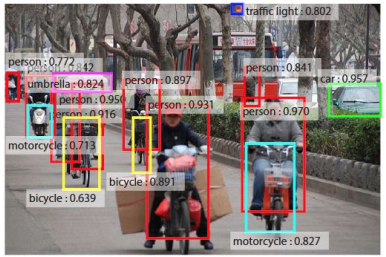

bus : 0.999
person : 0.985
person : 0.995
person : 0.996


cow : 0.979
cow : 0.974
cow : 0.979
cow : 0.892
person : 0.998

Faster RCNN


person : 0.673
person : 0.766
backpack : 0.756
person : 0.759
person : 0.950
person : 0.805
handbag : 0.848
dog : 0.996
dog : 0.691
person : 0.996


airplane : 0.997
person : 0.867

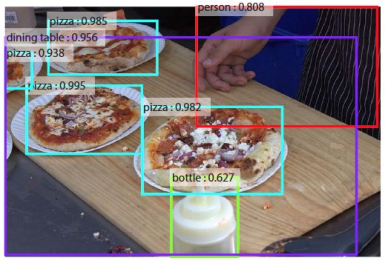

traffic light : 0.802
person : 0.772
umbrella : 0.824
person : 0.897
person : 0.841
car : 0.957
person : 0.950
person : 0.931
person : 0.970
motorcycle : 0.713
bicycle : 0.891
bicycle : 0.639
motorcycle : 0.827

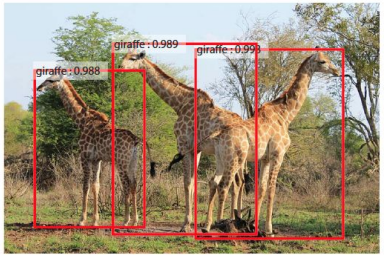

clock : 0.986
clock : 0.981
person : 0.800

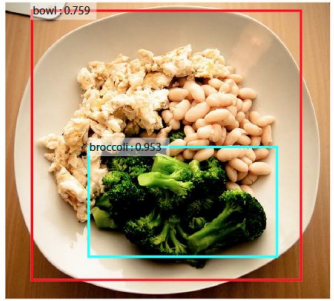

person : 0.998
skis : 0.919

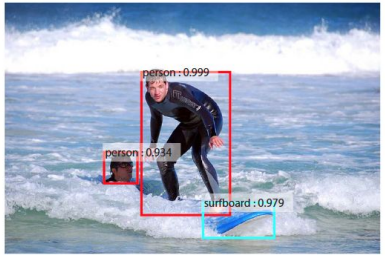

clock : 0.982


bed : 0.999


person : 0.808
pizza : 0.985
dining table : 0.956
pizza : 0.938
pizza : 0.995
pizza : 0.982
bottle : 0.627


giraffe : 0.989
giraffe : 0.988
giraffe : 0.990


bowl : 0.759
broccoli : 0.953


person : 0.999
person : 0.934
surfboard : 0.979


boat : 0.992
person : 0.691
person : 0.716
umbrella : 0.885
person : 0.665
person : 0.692
person : 0.618
person : 0.813
person : 0.864

Object detection: benchmark performance

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288×288	2007+2012	69.0	91
YOLOv2 352×352	2007+2012	73.7	81
YOLOv2 416×416	2007+2012	76.8	67
YOLOv2 480×480	2007+2012	77.8	59
YOLOv2 544×544	2007+2012	**78.6**	40

Table 3: Detection frameworks on PASCAL VOC 2007. YOLOv2 is faster and more accurate than prior detection methods. It can also run at different resolutions for an easy tradeoff between speed and accuracy. Each YOLOv2 entry is actually the same trained model with the same weights, just evaluated at a different size. All timing information is on a Geforce GTX Titan X (original, not Pascal model).

Redmon, J., & Farhadi, A. (2017). YOLO9000: better, faster, stronger. *arXiv preprint*.

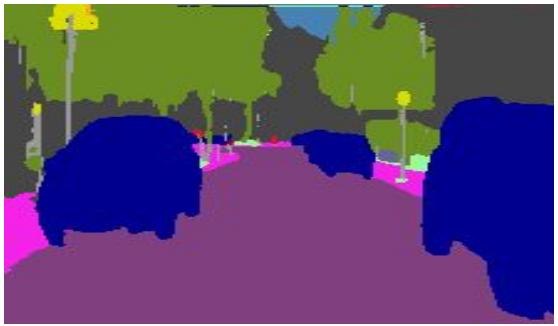
Outline

- Basic Application Principles of NNs in Supervised Learning
- Case study 1: Image Classification
- Case study 2: Object Detection
- **Case study 3: Segmentation**
- Case study 4: Pose Estimation
- Case study 5: Style Transfer

Semantic Segmentation

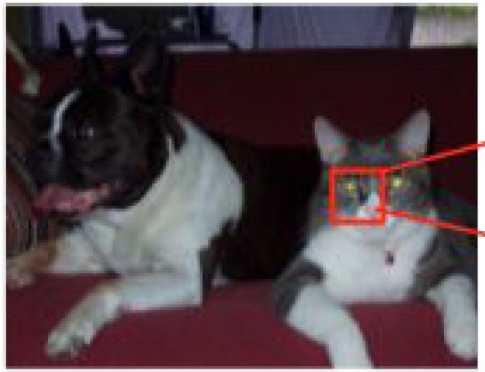
- Given an image, partition the image into coherent parts

- Input?
- Output? Pixel-level class map

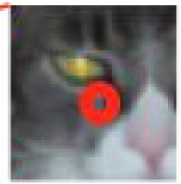


- Loss? Pixel level classification loss

From Classification to Semantic Segmentation



extract a patch



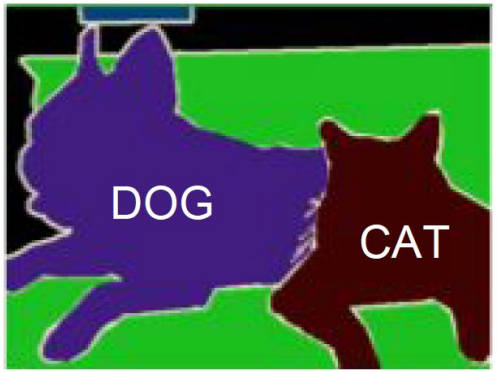
run through a CNN trained for image classification

CNN

classify the center pixel

CAT

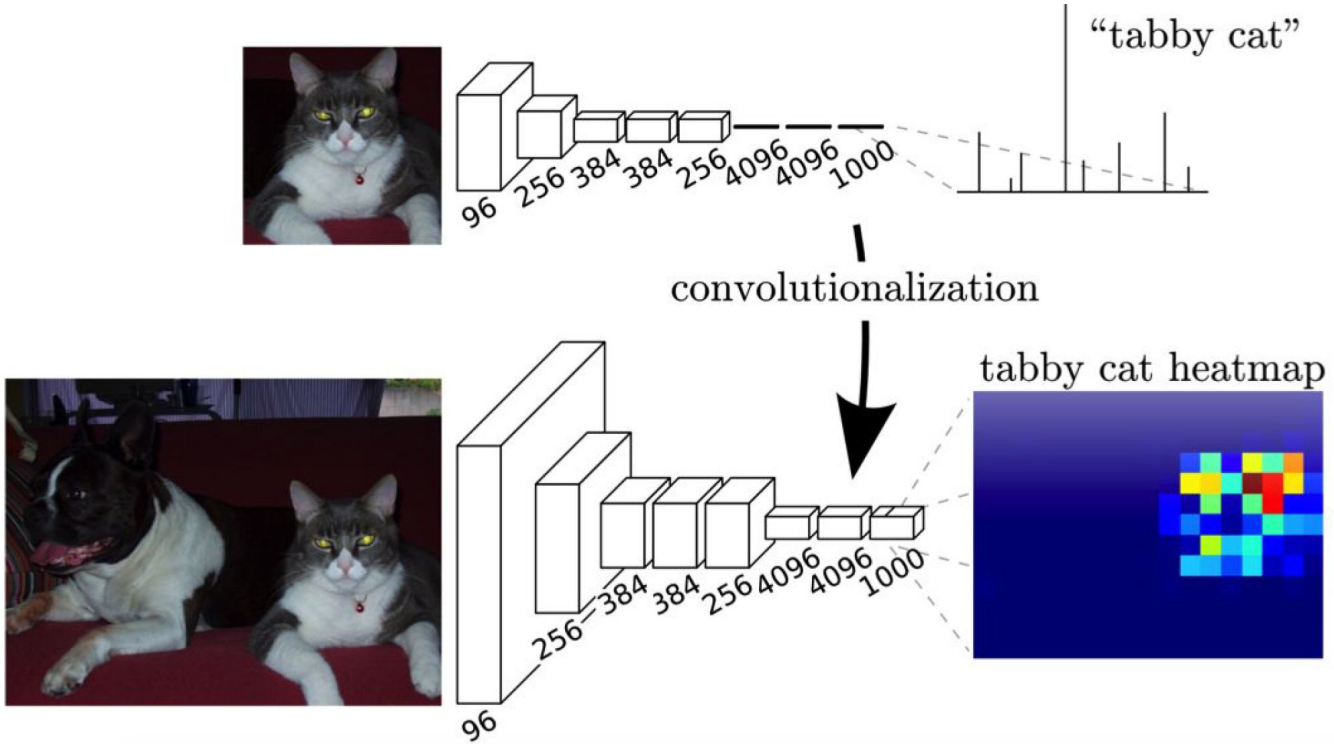
repeat for every pixel



Long, Jonathan, et al. "Fully convolutional networks for semantic segmentation." CVPR 2015

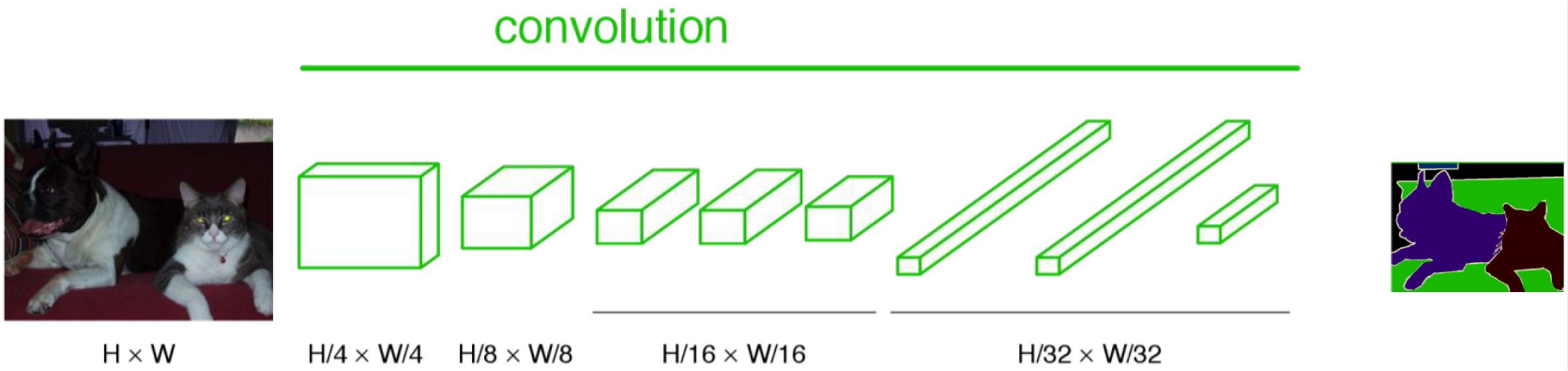
From Classification to Semantic Segmentation

- A classification network becoming fully convolutional



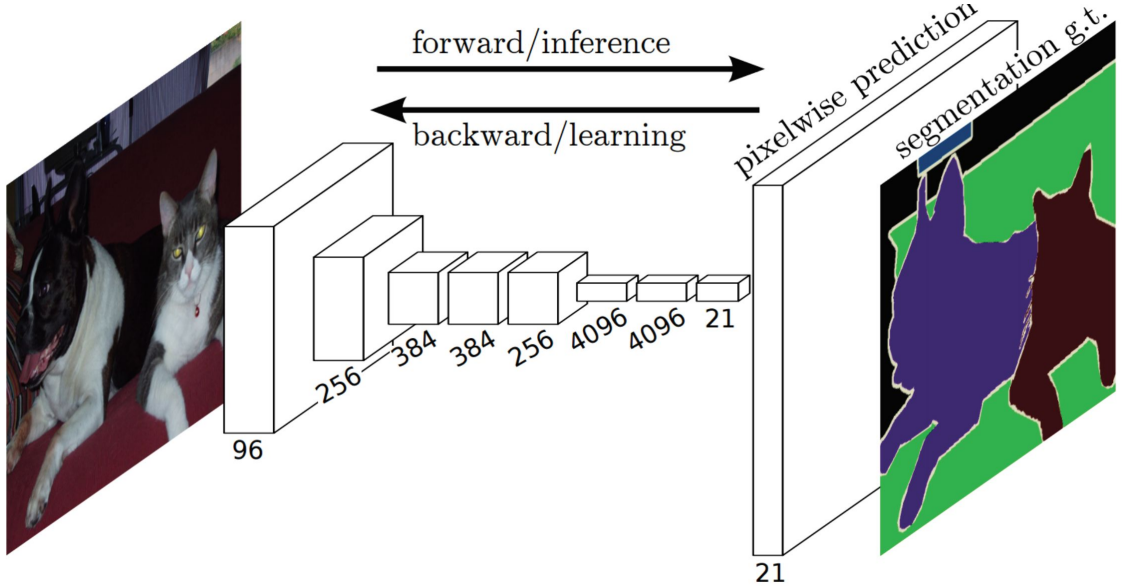
From Classification to Semantic Segmentation

- Dense prediction: fully convolutional, end to end, pixel-to-pixel network
- Problems
  - Output is smaller than input → Add upsampling layers



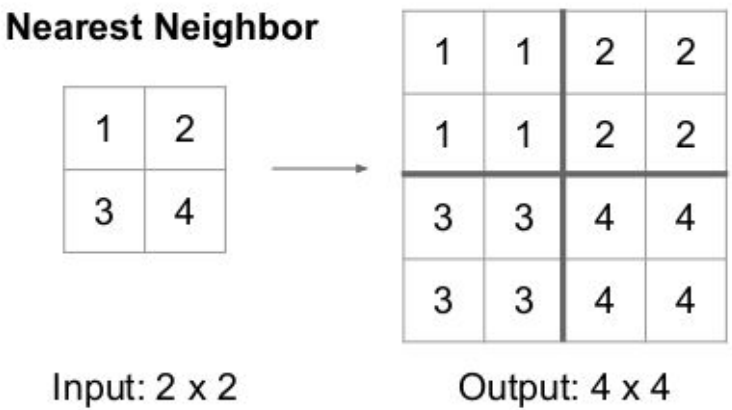
From Classification to Semantic Segmentation

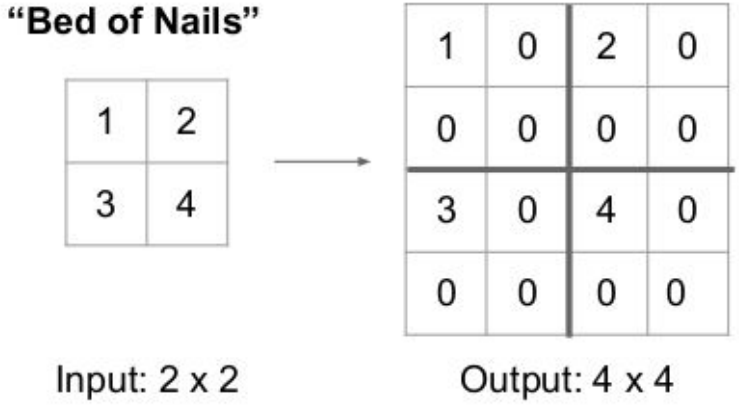
- Dense prediction: fully convolutional, end to end, pixel-to-pixel network
- Learnable upsampling



Long, Jonathan, et al. "Fully convolutional networks for semantic segmentation." CVPR 2015

Learnable Upsampling: Unpooling





Learnable Upsampling: Max Unpooling

Max Pooling
Remember which element was max!

1	2	6	3
3	5	2	1
1	2	2	1
7	3	4	8

Input: 4 x 4

5	6
7	8

Output: 2 x 2

Rest of the network

Max Unpooling
Use positions from pooling layer

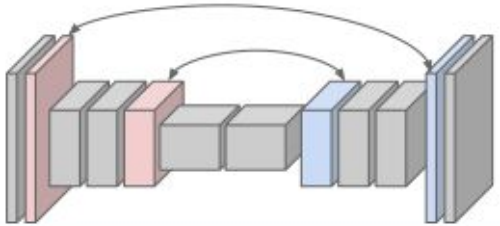
1	2
3	4

Input: 2 x 2

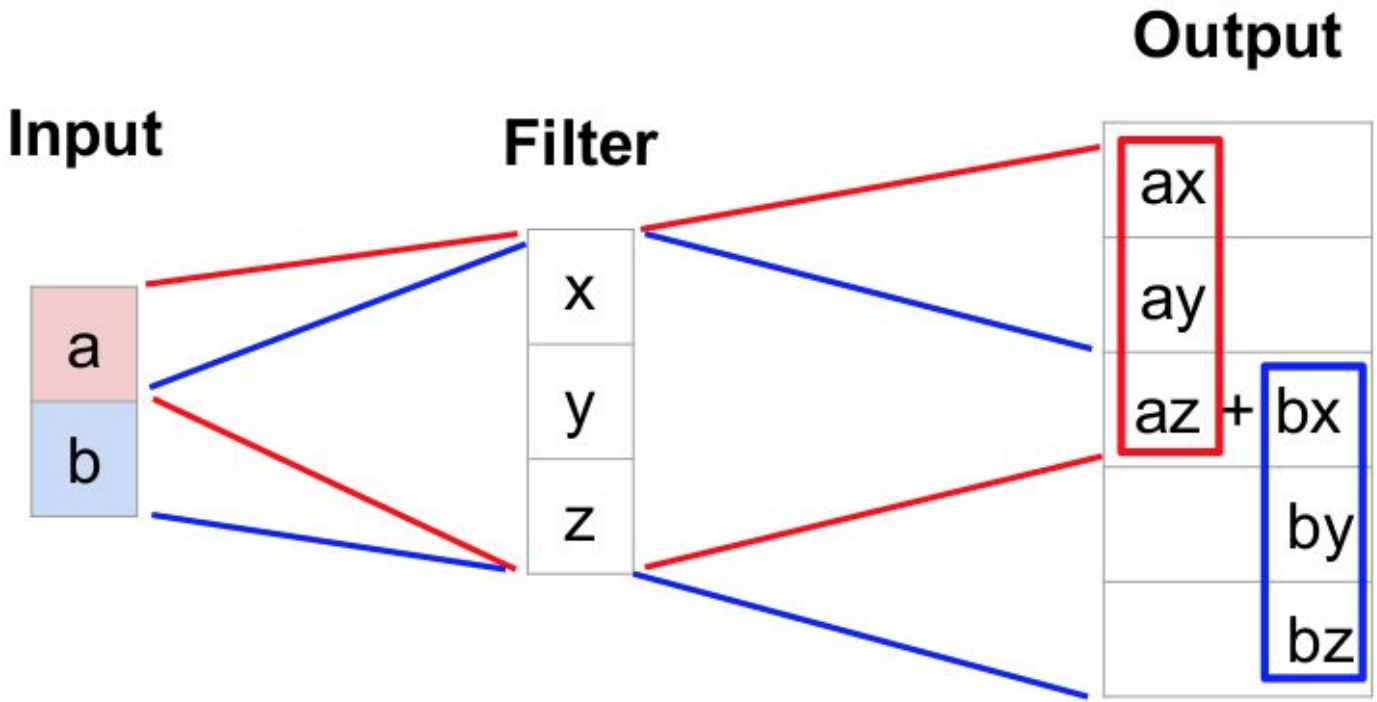
0	0	2	0
0	1	0	0
0	0	0	0
3	0	0	4

Output: 4 x 4

Corresponding pairs of downsampling and upsampling layers



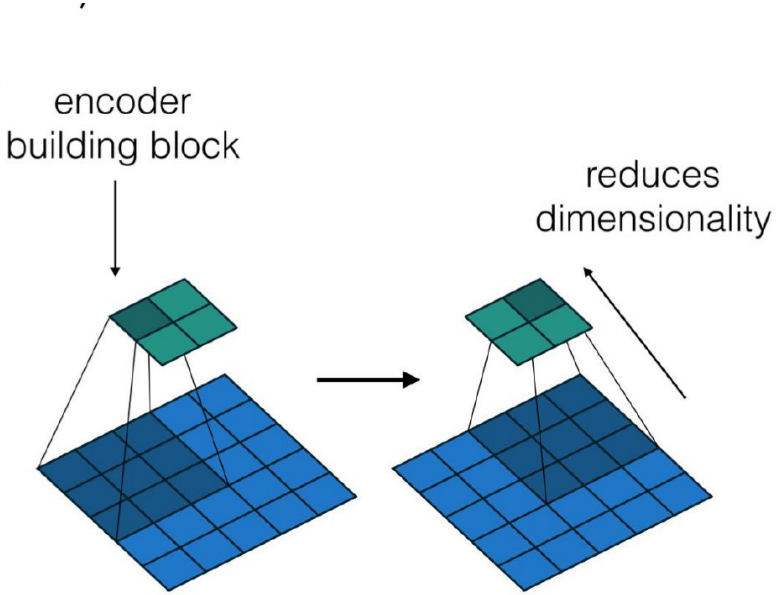
Learnable Upsampling: Deconvolution

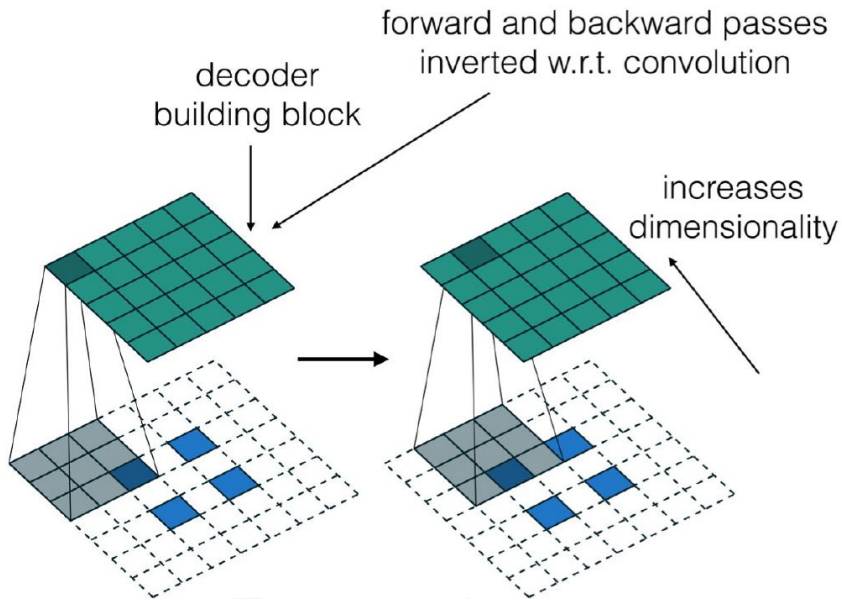


Output contains copies of the filter weighted by the input, summing at where at overlaps in the output

Need to crop one pixel from output to make output exactly 2x input

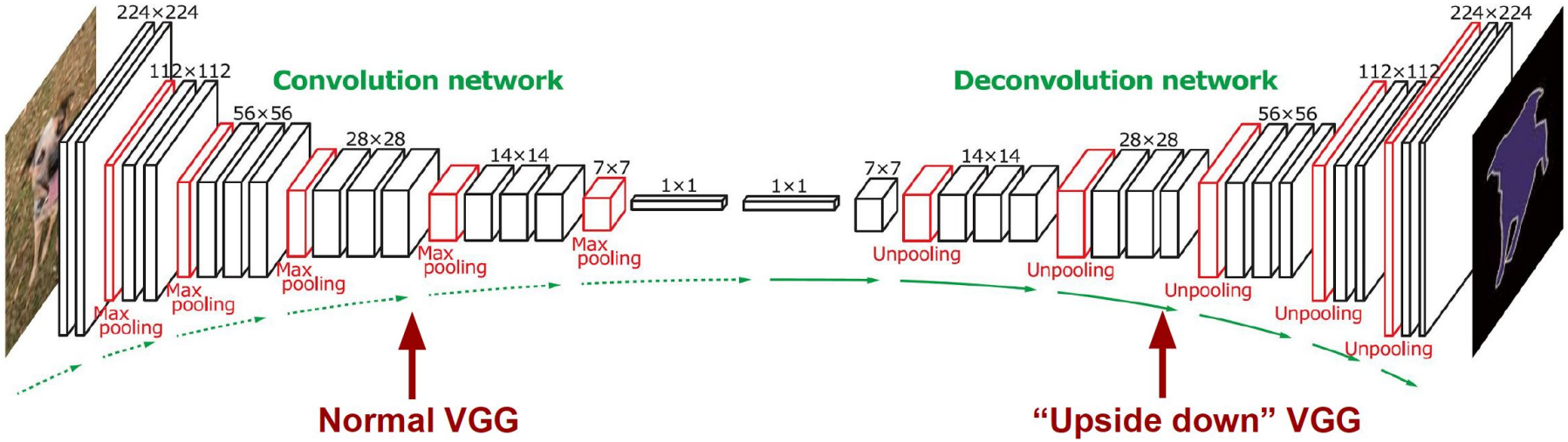
Deconvolution





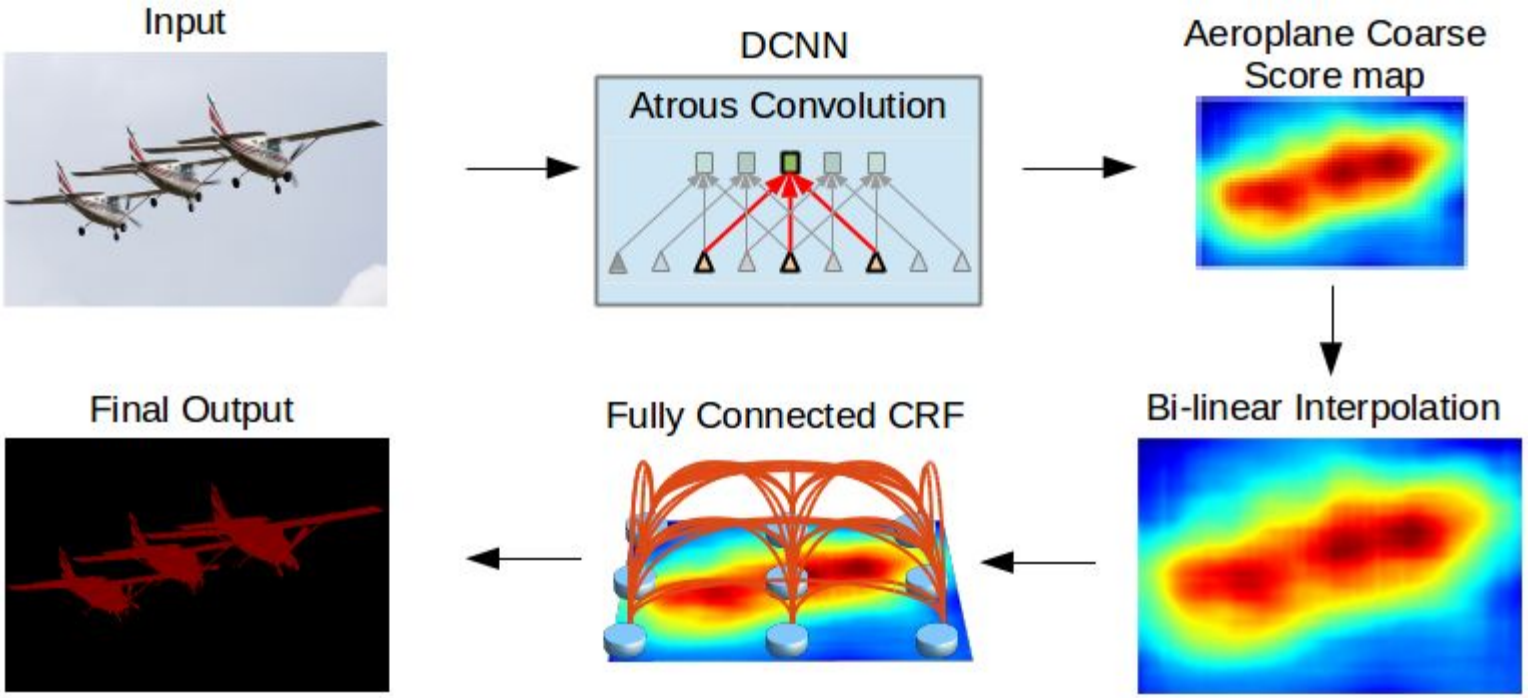
Deconvolutional Network

- DeconvNet: VGG-16 (conv+Relu+MaxPool) + mirrored VGG (Unpooling+deconv+Relu)



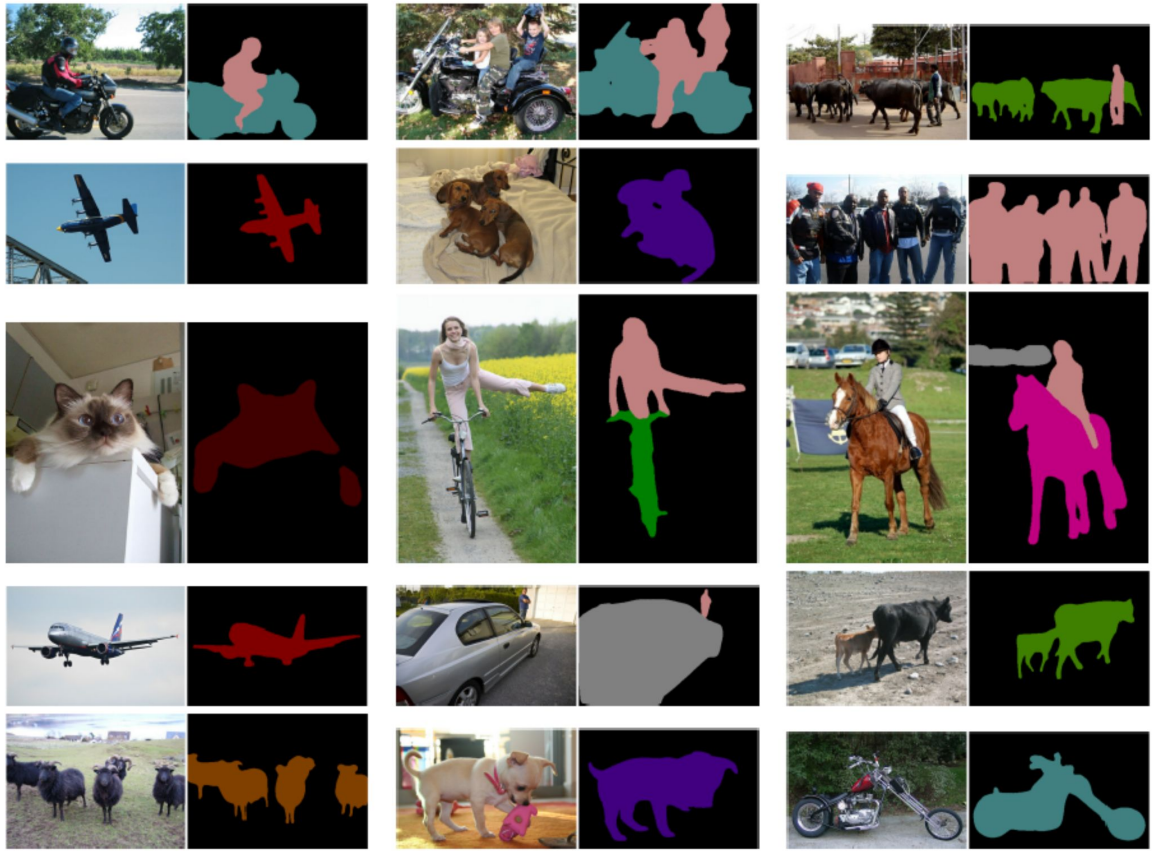
Noh, Hyeonwoo, et al. "Learning deconvolution network for semantic segmentation." ICCV 2015

DeepLab



Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 834-848.

Example results



Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, *40*(4), 834-848.

Semantic Segmentation:Benchmark results

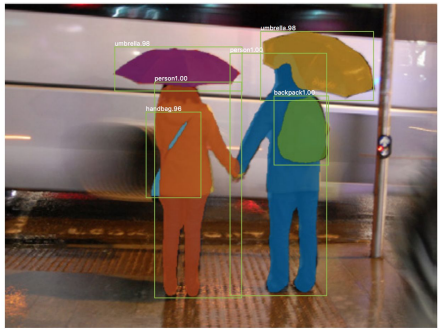
Method	mIOU
Adelaide_VeryDeep_FCN_VOC [85]	79.1
LRR_4x_ResNet-CRF [25]	79.3
DeepLabv2-CRF [11]	79.7
CentraleSupelec Deep G-CRF [8]	80.2
HikSeg_COCO [80]	81.4
SegModel [75]	81.8
Deep Layer Cascade (LC) [52]	82.7
TuSimple [84]	83.1
Large_Kernel_Matters [68]	83.6
Multipath-RefineNet [54]	84.2
ResNet-38_MS_COCO [86]	84.9
PSPNet [95]	85.4
IDW-CNN [83]	86.3
CASIA_IVA_SDN [23]	86.6
DIS [61]	86.8
DeepLabv3	85.7
DeepLabv3-JFT	86.9

Table 7. Performance on PASCAL VOC 2012 *test* set.

Chen, Liang-Chieh, et al. "Rethinking atrous convolution for semantic image segmentation." *arXiv preprint arXiv:1706.05587* (2017)

Instance Segmentation

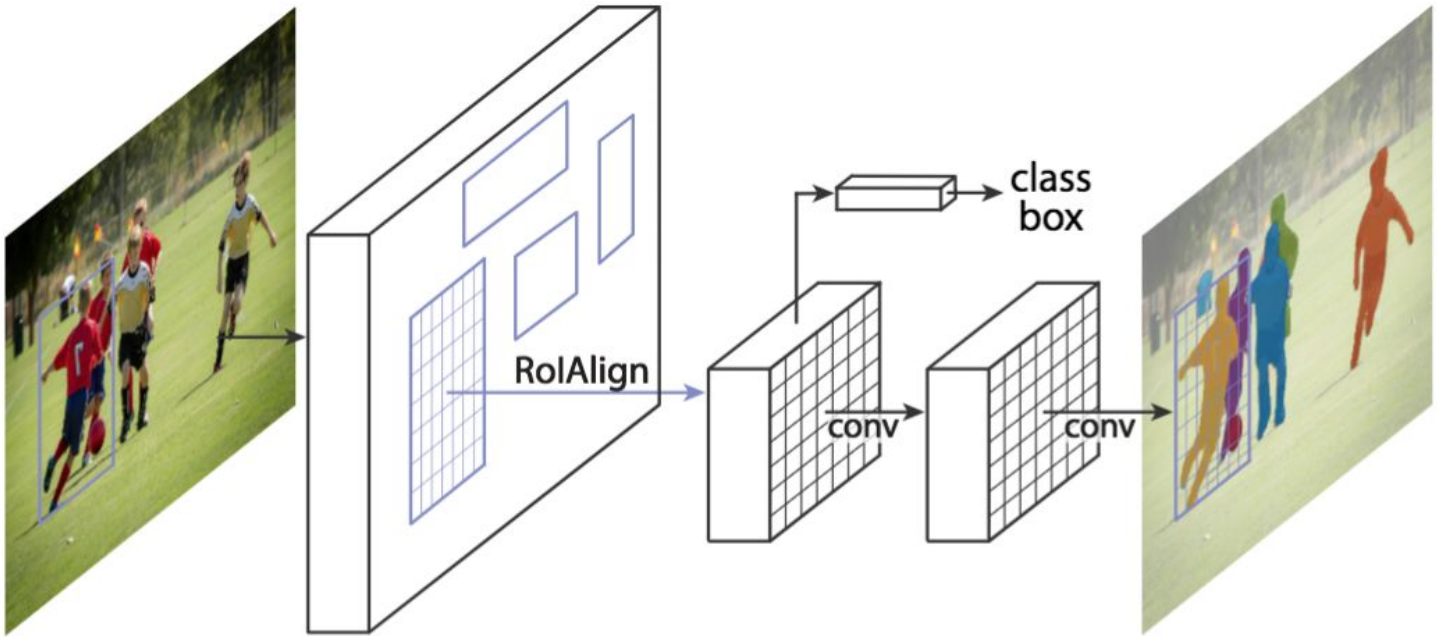
- Given an image, partition the image into coherent parts and identify each entity separately
- Input?



- Output? A class map
- Loss? Pixel level classification loss

Mask RCNN

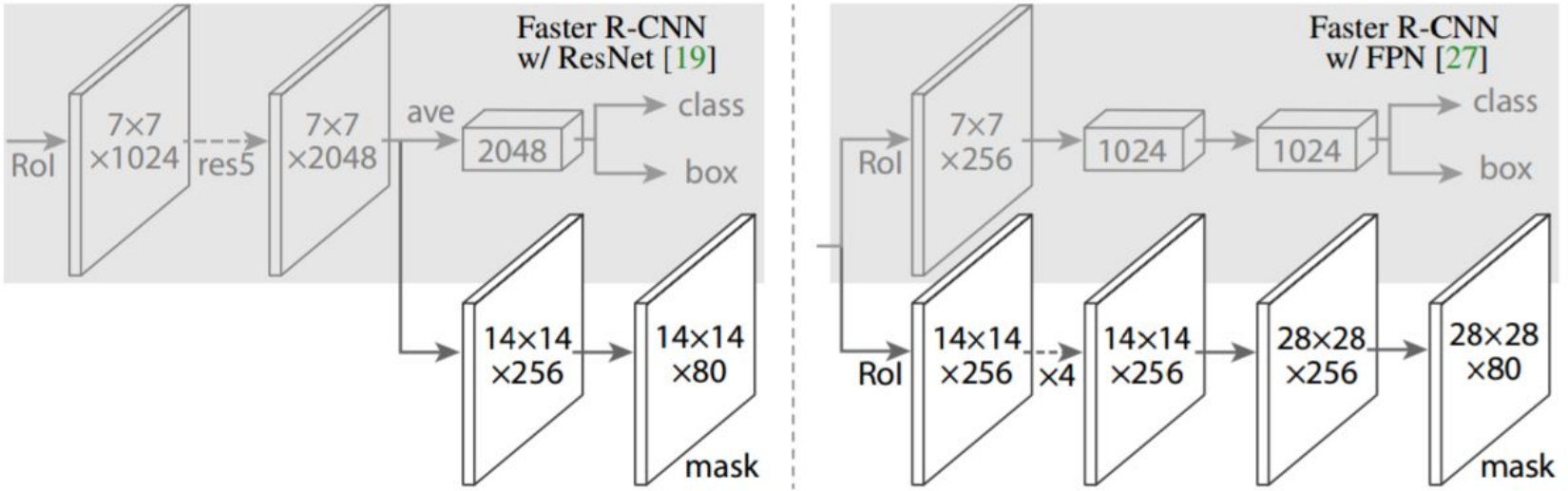
- Faster-RCNN architecture with a third mask head



K. He and al. Mask Region-based Convolutional Network (Mask R-CNN) NIPS 2017

Mask RCNN

- Mask R-CNN extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding box regression



K. He and al. Mask Region-based Convolutional Network (Mask R-CNN) NIPS 2017

Mask RCNN

$$L = L_{cls} + L_{box} + L_{mask}$$

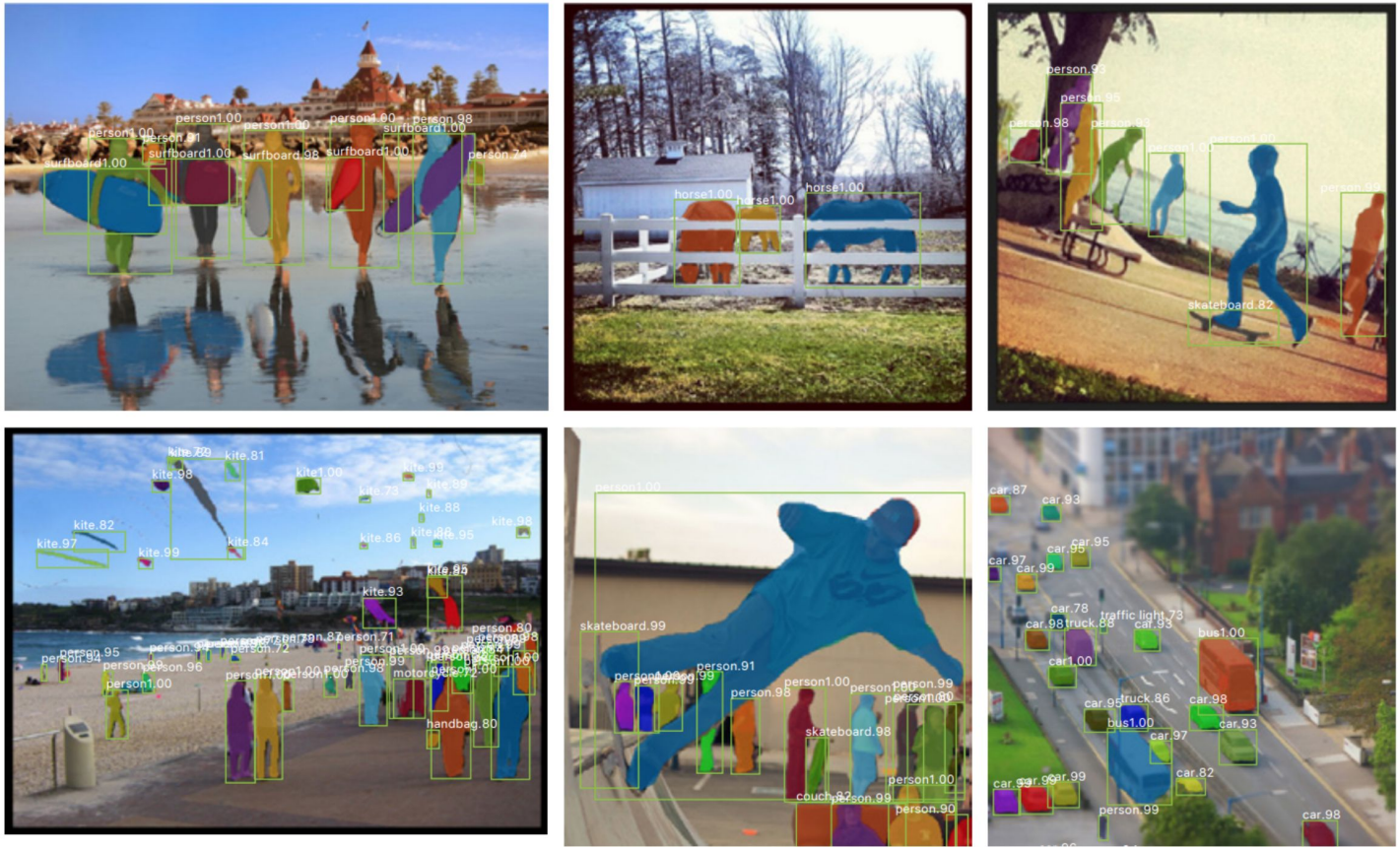
- The mask branch has a K x m x m -dimensional output for each RoI, which encodes K binary masks of resolution mxm, one for each of the K classes.
- Applying per-pixel sigmoid
- For an RoI associated with ground-truth class k, Lmask is only defined on the k-th mask

K. He and al. Mask Region-based Convolutional Network (Mask R-CNN) NIPS 2017

Mask RCNN

K. He and al. Mask Region-based Convolutional Network (Mask R-CNN) NIPS 2017

Mask RCNN



K. He and al. Mask Region-based Convolutional Network (Mask R-CNN) NIPS 2017

Outline

- Basic Application Principles of NNs in Supervised Learning
- Case study 1: Image Classification
- Case study 2: Object Detection
- Case study 3: Segmentation
- **Case study 4: Pose Estimation**
- Case study 5: Style Transfer

Pose estimation

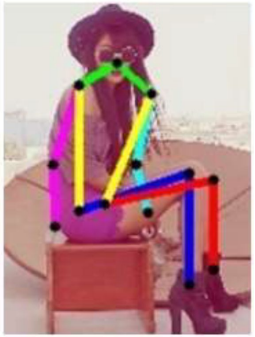
What's the "pose" of an object?

- Rigid-body object: "Orientation"
  - pitch, yaw, roll
  - Represented by quaternions, Euler angles, etc.
- Articulated object (e.g., with joints): position of individual joints
  - Represented by 2d or 3d coordinates for individual joints

Human Pose Estimation

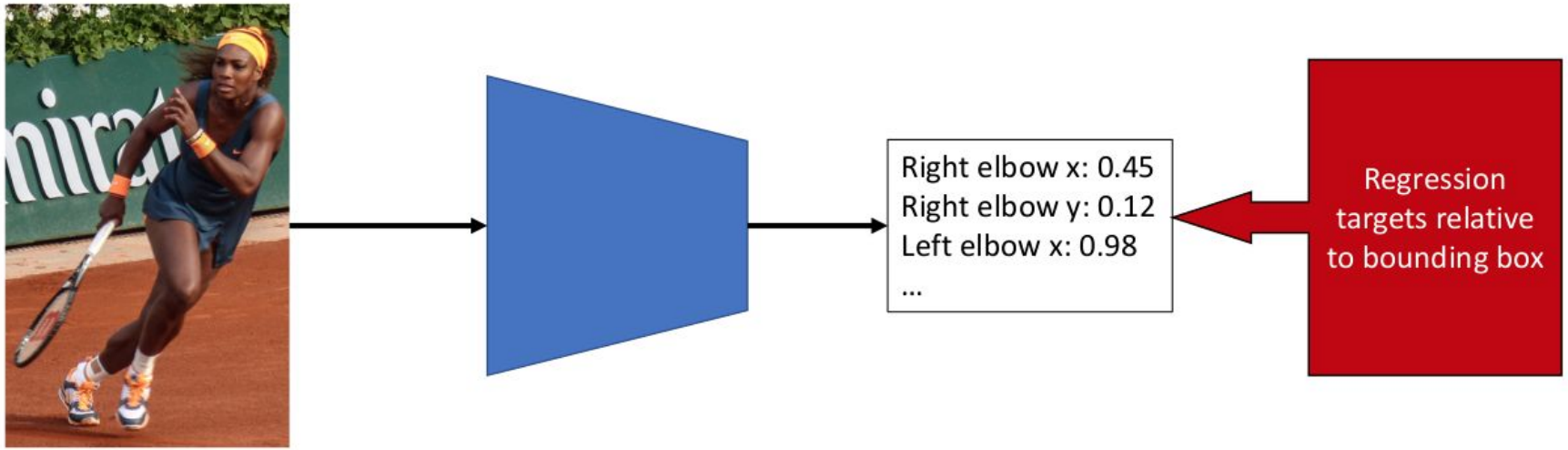
- Given an human body image, mark joint locations
- Input?

- Output?



- Loss? Error in the joint location

Strategy 1: Regression



DeepPose: Human Pose Estimation via Deep Neural Networks. Alexander Toshev and Christian Szegedy. In *CVPR,* 2014.

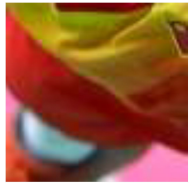
DeepPose

- Can you tell which part is from an image patch?

Slide credit: Wei Yang

DeepPose

- What about this patch?



Slide credit: Wei Yang

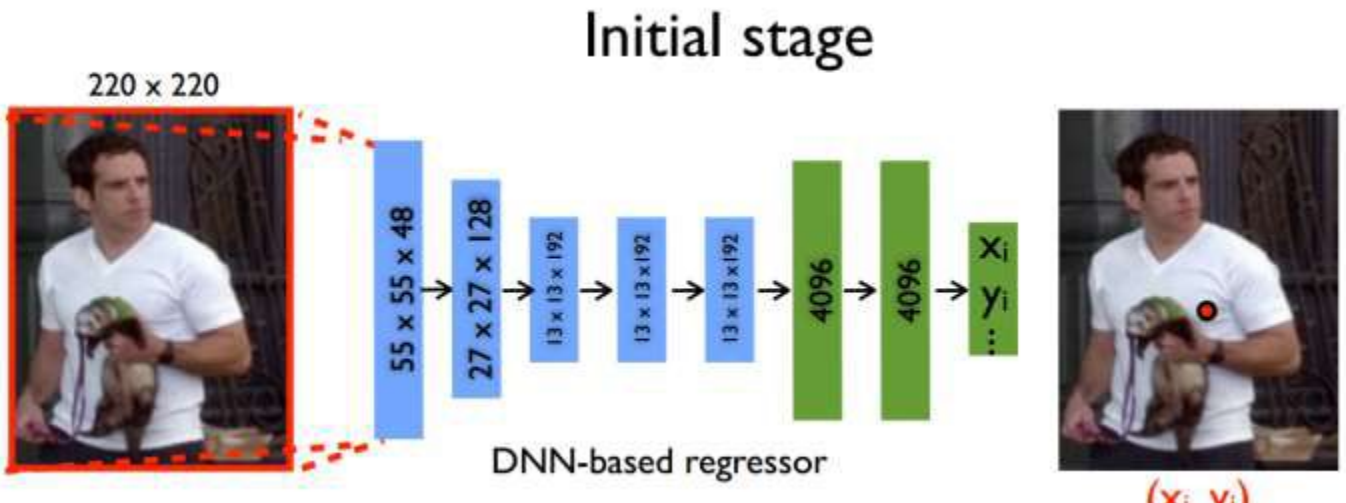
DeepPose

Slide credit: Wei Yang

DeepPose

- Holistic human pose estimation as a DNN
- Training: Minimizing L2 distance between the prediction and the true pose vector.

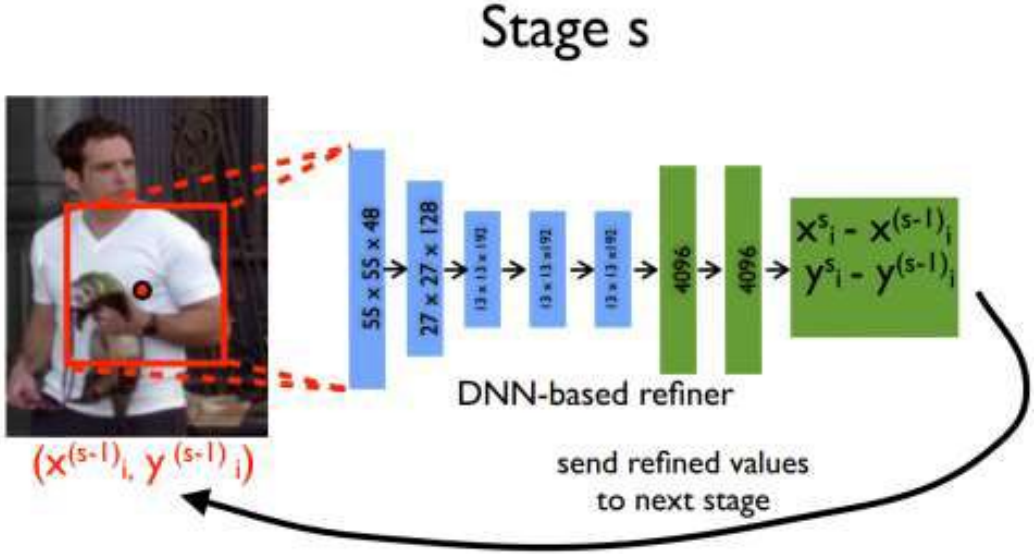
$$\arg\min_{\theta} \sum_{(x,y)\in D_N} \sum_{i=1}^{k} ||\mathbf{y}_i - \psi_i(x;\theta)||_2^2$$



Slide credit: Wei Yang

DeepPose: Cascade of Pose Regressors

- due to the fixed input size of 220 x 220, the network has limited capacity to look at detail
- it learns filters capturing pose properties at coarse scale



Slide credit: Wei Yang

DeepPose: Cascade of Pose Regressors

- pipeline

Slide credit: Wei Yang

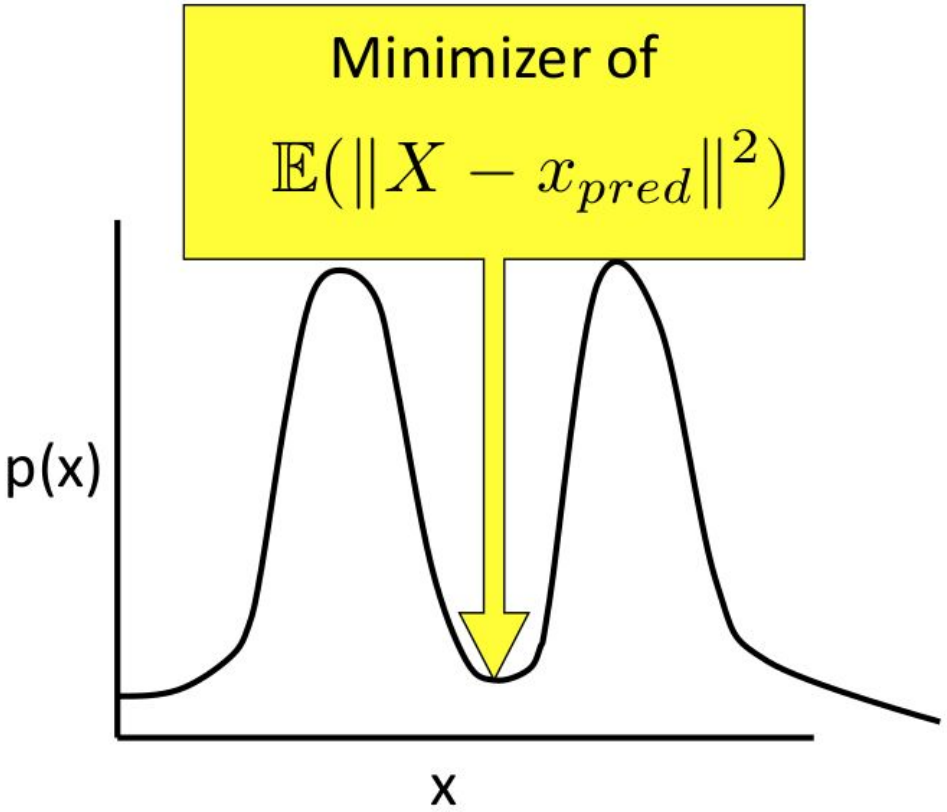
DeepPose: Cascade of Pose Regressors

- Experiment

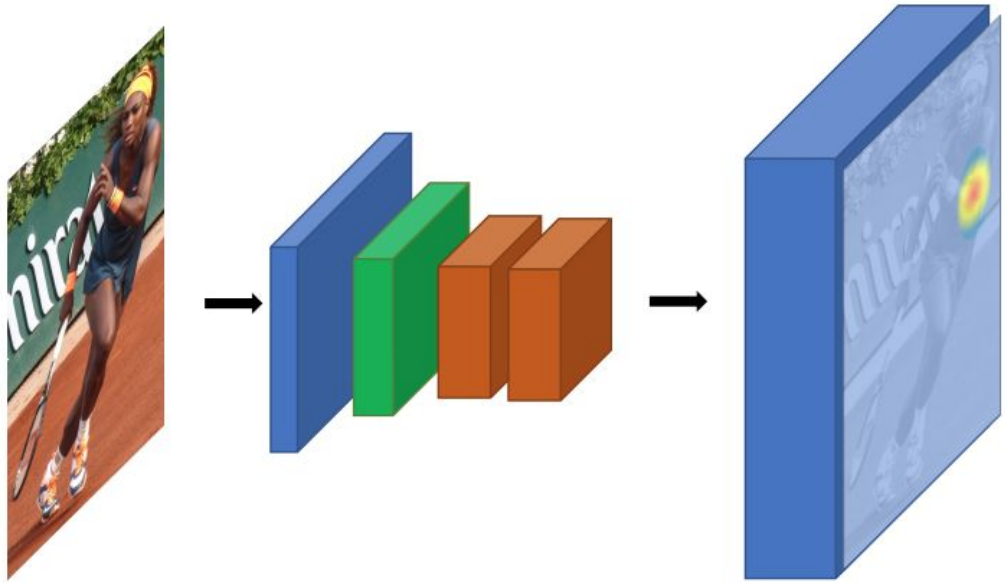
Slide credit: Wei Yang

Strategy 1: Regression

- Multimodal distributions?



Strategy 2: Heatmaps

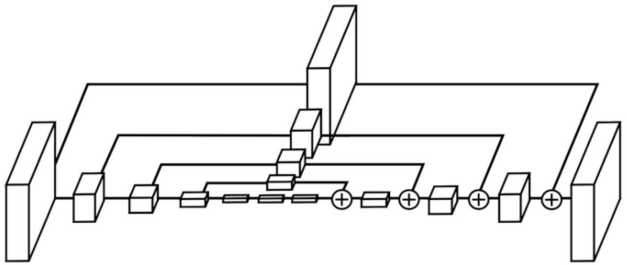


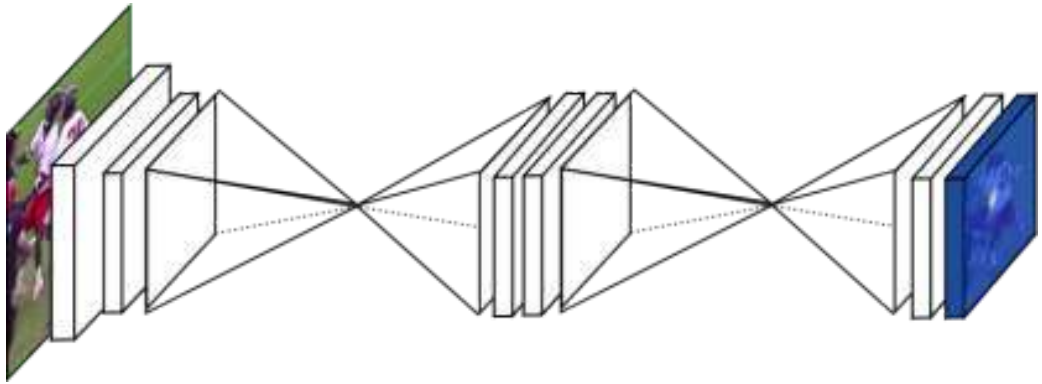
Stacked Hourglass

- Why is cascade not efficient enough?
- Refinement of position within a local window could not offer much in the way of improvement for:
  - occluded limb
  - limb out of the window
- We need to search over all scales of image

Stacked Hourglass

- Local appearance: essential for part detection
- Global understanding: orientation of the body, limb arrangement, relationships between parts, …
- Single branch with bottom-up,top-down mechanism

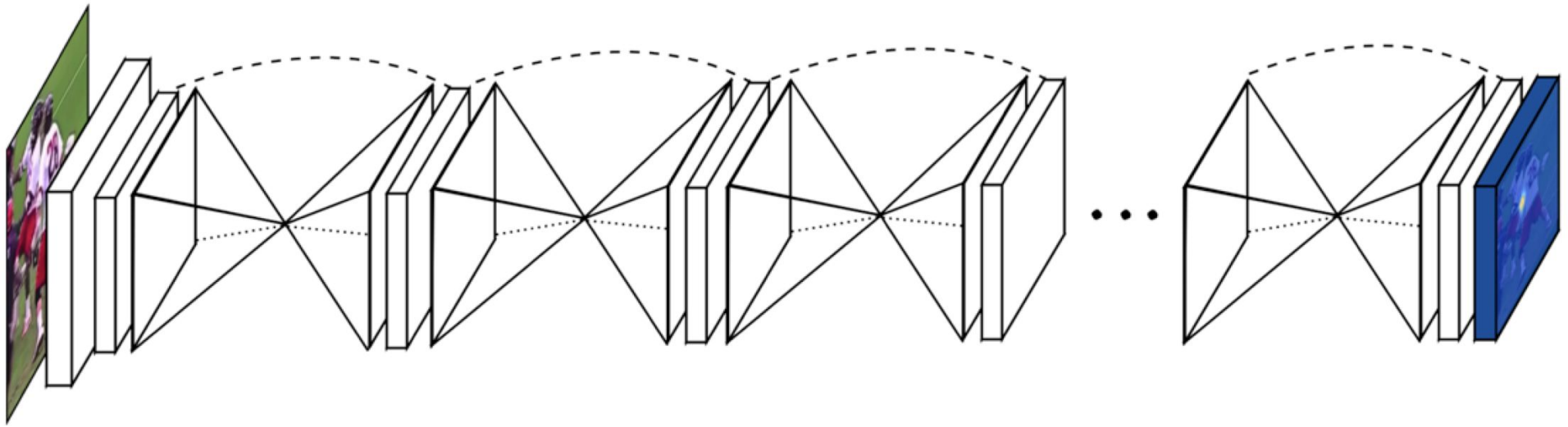




Stacked Hourglass Networks for Human Pose Estimation. Alejandro Newell, Kaiyu Yang, and Jia Deng. ECCV2016.

Stacked Hourglass

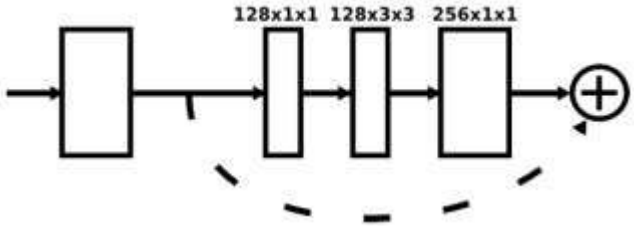
- Each refinement round has to
  - Combine global information about pose
  - Use global pose information to produce new precise pose estimate

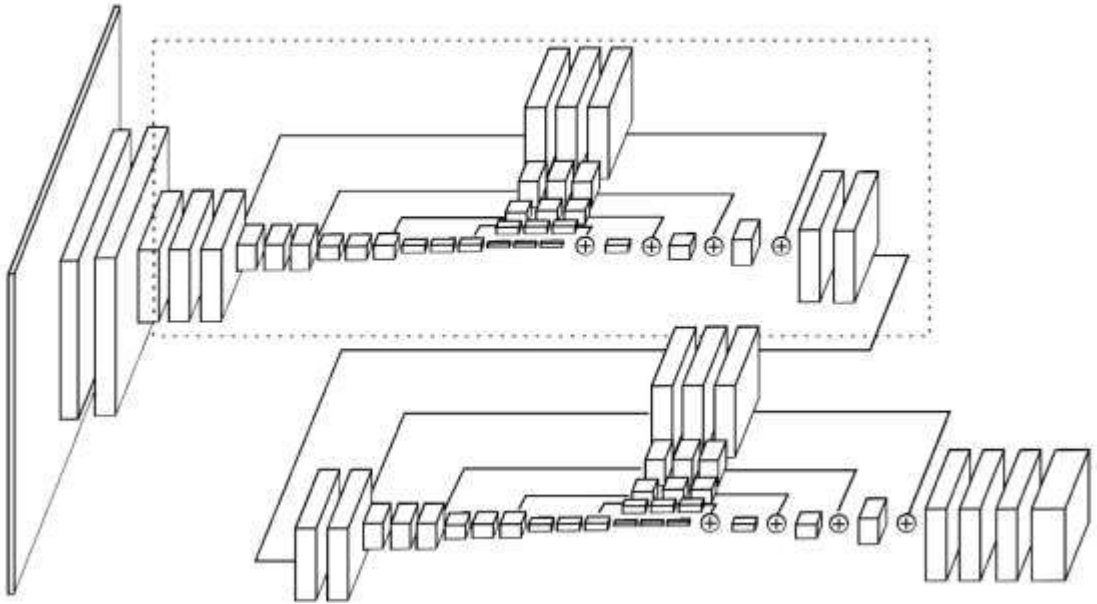


Stacked Hourglass Networks for Human Pose Estimation. Alejandro Newell, Kaiyu Yang, and Jia Deng. ECCV2016.

Stacked Hourglass

- Two stacked hourglass module
- Layers are identical across each module
- Each box corresponds to a residual module





Benchmark results

	Head	Shoulder	Elbow	Wrist	Hip	Knee	Ankle	Total
Tompson et al. [16], CVPR'15	96.1	91.9	83.9	77.8	80.9	72.3	64.8	82.0
Carreira et al. [19], CVPR'16	95.7	91.7	81.7	72.4	82.8	73.2	66.4	81.3
Pishchulin et al. [17], CVPR'16	94.1	90.2	83.4	77.3	82.6	75.7	68.6	82.4
Hu et al. [27], CVPR'16	95.0	91.6	83.0	76.6	81.9	74.5	69.5	82.4
Wei et al. [18], CVPR'16	97.8	95.0	88.7	84.0	88.4	82.8	79.4	88.5
Hourglass networks	**98.2**	**96.3**	**91.2**	**87.1**	**90.1**	**87.4**	**83.6**	**90.9**

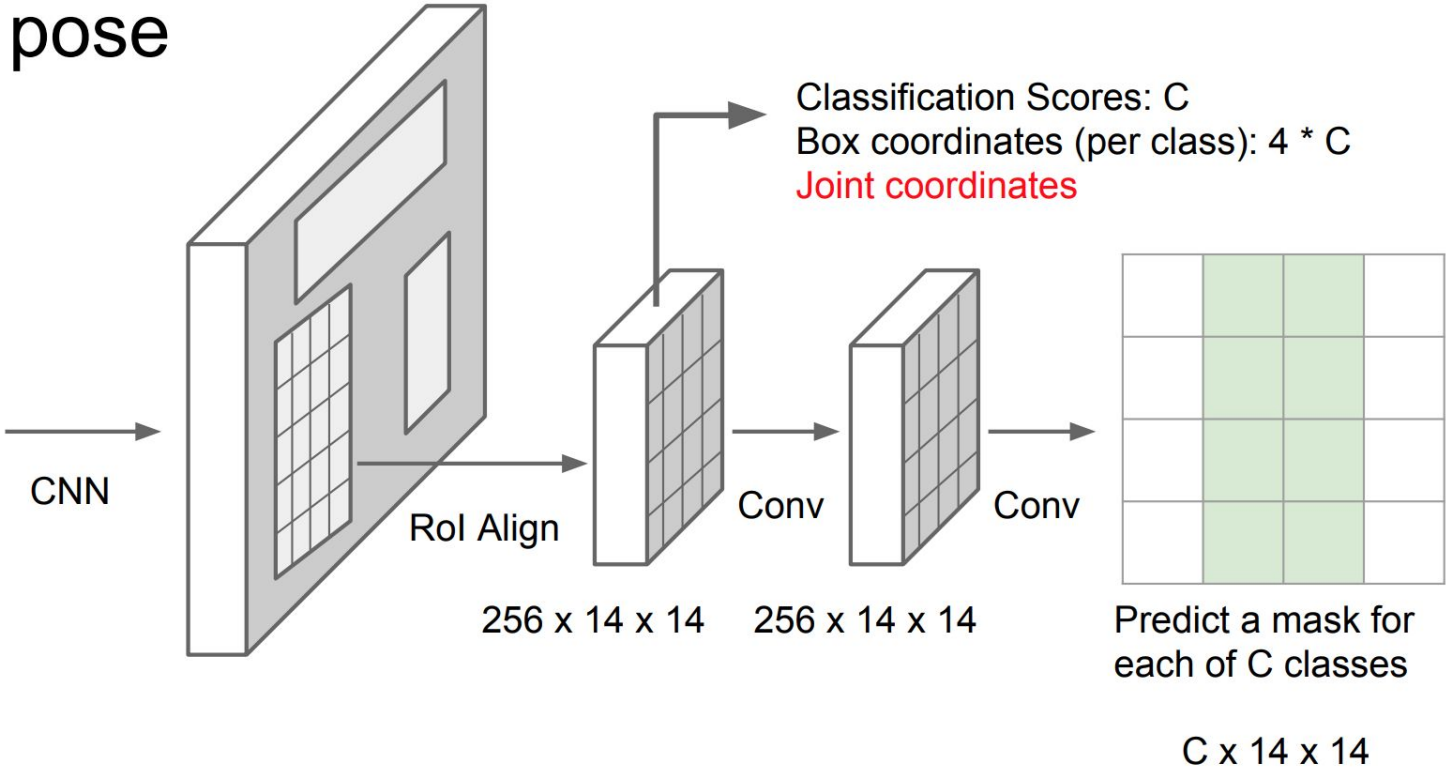
Table 2. Results on MPII Human Pose (PCKh@0.5)

(Hourglass Net)

Newell, A., Yang, K., & Deng, J. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision* 2016

Pose estimation with Mask RCNN

Also does pose



He et al, “Mask R-CNN”, arXiv 2017

Mask RCNN

Mask RCNN

Outline

- Basic Application Principles of NNs in Supervised Learning
- Case study 1: Image Classification
- Case study 2: Object Detection
- Case study 3: Segmentation
- Case study 4: Pose Estimation
- **Case study 5: Style Transfer**

Style Transfer

- Given a content image, transfer it according to style image
- Input?

content image

style image

- Output?

- Loss? “Style loss” + “content loss”

Style Transfer

- Given a content image, transfer it according to style image
- Input?

content image

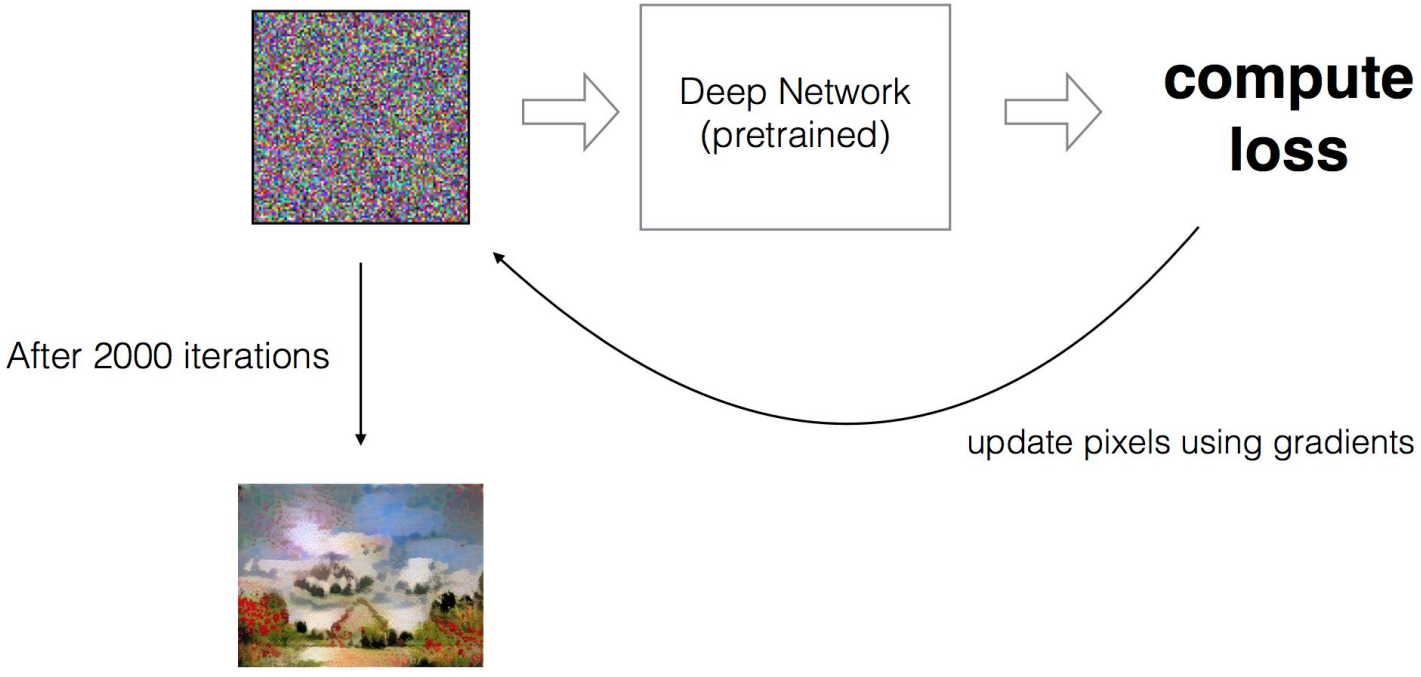
style image

- Output?

- Loss? $L = \left|\left|Content_C - Content_G\right|\right|_2^2 + \left|\left|Style_S - Style_G\right|\right|_2^2$

Style Transfer

- We are not learning parameters by minimizing L. We are learning an image!



Feature Inversion

- Given a feature vector for an image, find a new image such that:
  - Its features are similar to the given features
  - It "looks natural" (image prior regularization)

$$\mathbf{x}^* = \underset{\mathbf{x} \in \mathbb{R}^{H \times W \times C}}{\operatorname{argmin}} \ \ell(\Phi(\mathbf{x}), \Phi_0) + \lambda \mathcal{R}(\mathbf{x})$$

Given feature vector

Features of new image

$$\ell(\Phi(\mathbf{x}), \Phi_0) = \|\Phi(\mathbf{x}) - \Phi_0\|^2$$

$$\mathcal{R}_{V^\beta}(\mathbf{x}) = \sum_{i,j} \left((x_{i,j+1} - x_{ij})^2 + (x_{i+1,j} - x_{ij})^2 \right)^{\frac{\beta}{2}}$$

Total Variation regularizer
(encourages spatial smoothness)

Mahendran and Vedaldi, "Understanding Deep Image Representations by Inverting Them", CVPR 2015

Feature Inversion

original image

Reconstructions from the 1000 log probabilities for ImageNet (ILSVRC) classes

Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015

Feature Inversion

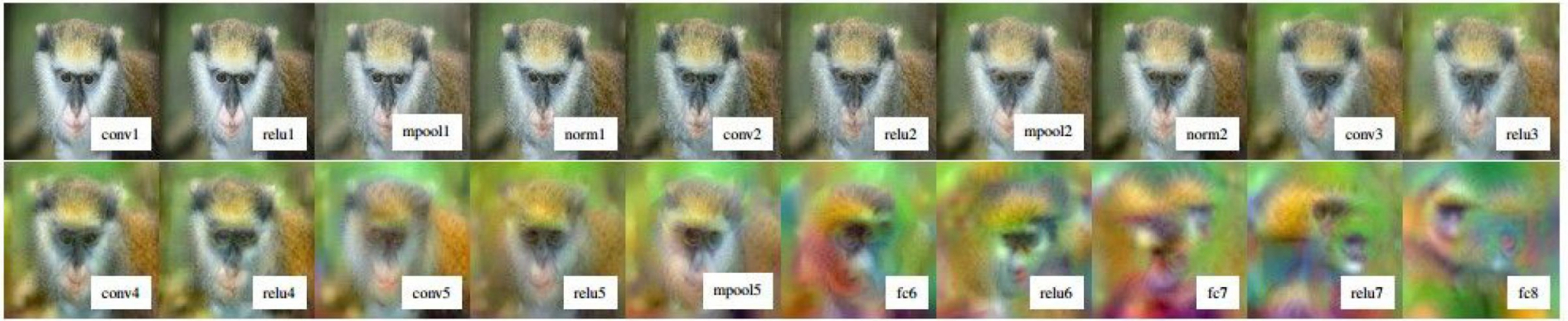
- Reconstructions from the representation after last last pooling layer (immediately before the first Fully Connected layer)

Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015

Feature Inversion

Reconstructions from intermediate layers

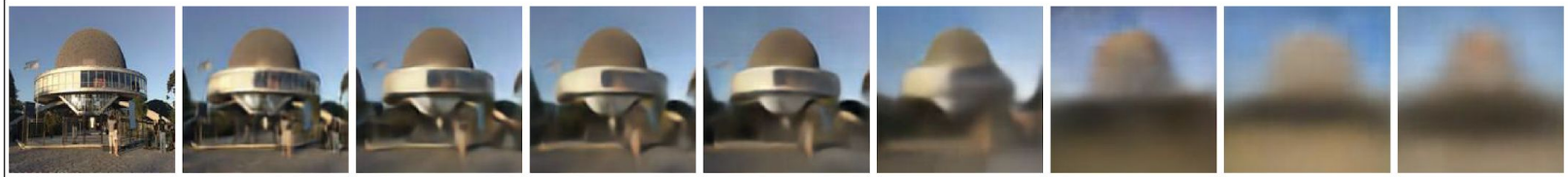
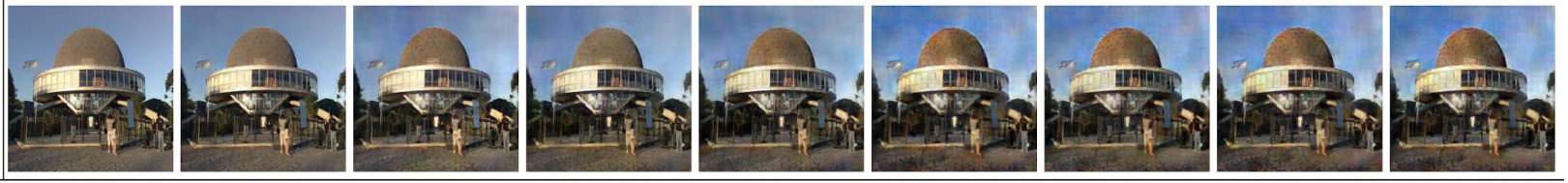
Higher layers are less sensitive to changes in color, texture, and shape



Mahendran and Vedaldi, “Understanding Deep Image Representations by Inverting Them”, CVPR 2015

Feature Inversion

BTW, photorealistic inversion is possible!

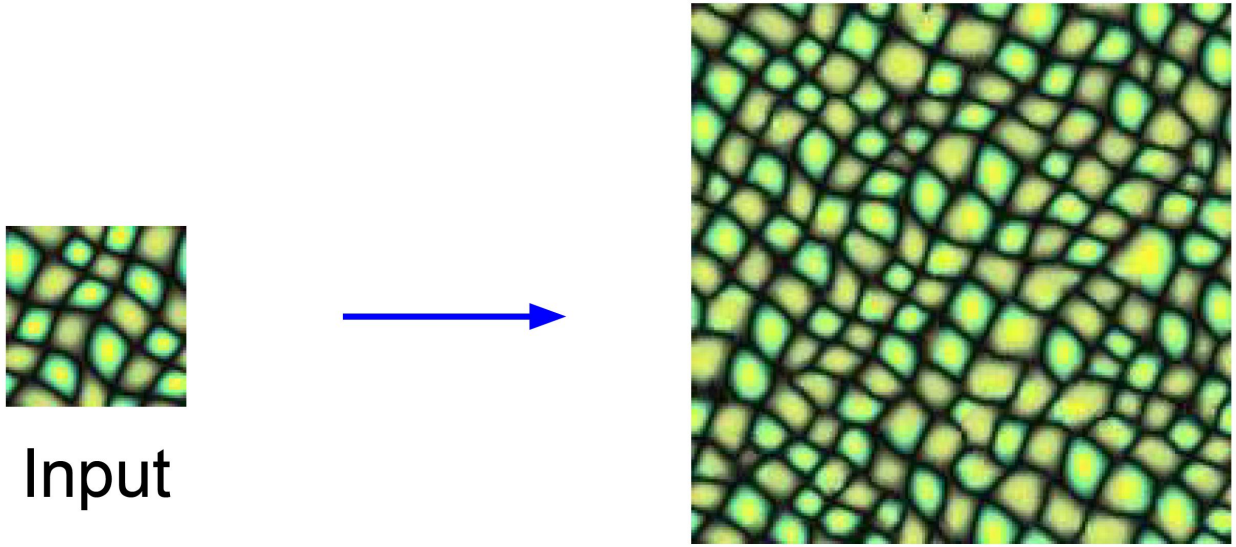
Layer	image	pool1	pool2	conv3	conv4	pool5	fc6	fc7	fc8
SAE Dosovitskiy & Brox (2016)									
SWWAE									

- Using the encoder pooling switches for unpooling, the information loss due to max-pooling can be better recovered.
- The extremely good reconstruction quality of SWWAE indicates the "convolutional filters + ReLU" cause **very minor** information losses.

Zhang, Y., Lee, K., & Lee, H. Augmenting supervised neural networks with unsupervised objectives for large-scale image classification. ICML 2016

Texture Synthesis

- Given a sample patch of some texture, can we generate a bigger image of the same texture?

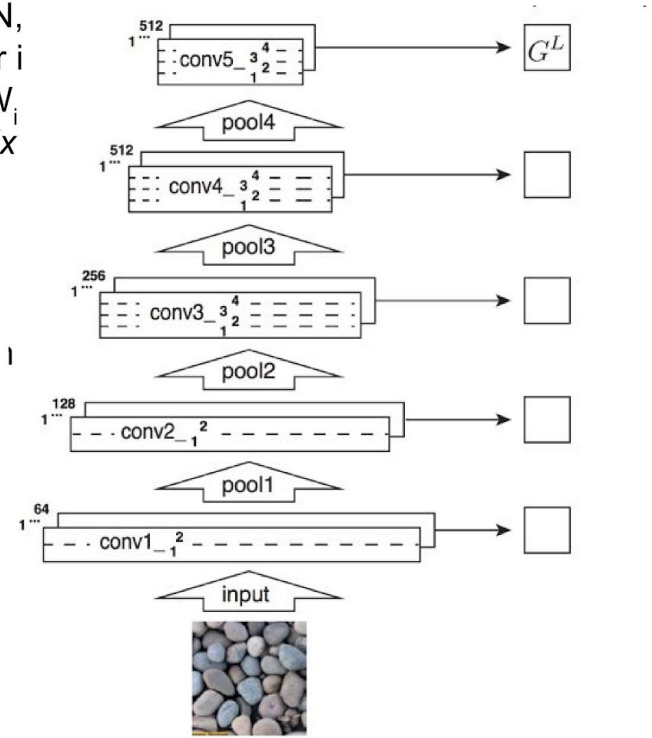


Texture Synthesis

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l\right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \text{ (shape } C_i \times C_i\text{)}$$



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Gatys et al, "Texture Synthesis using Convolutional Neural Networks", NIPS 2015

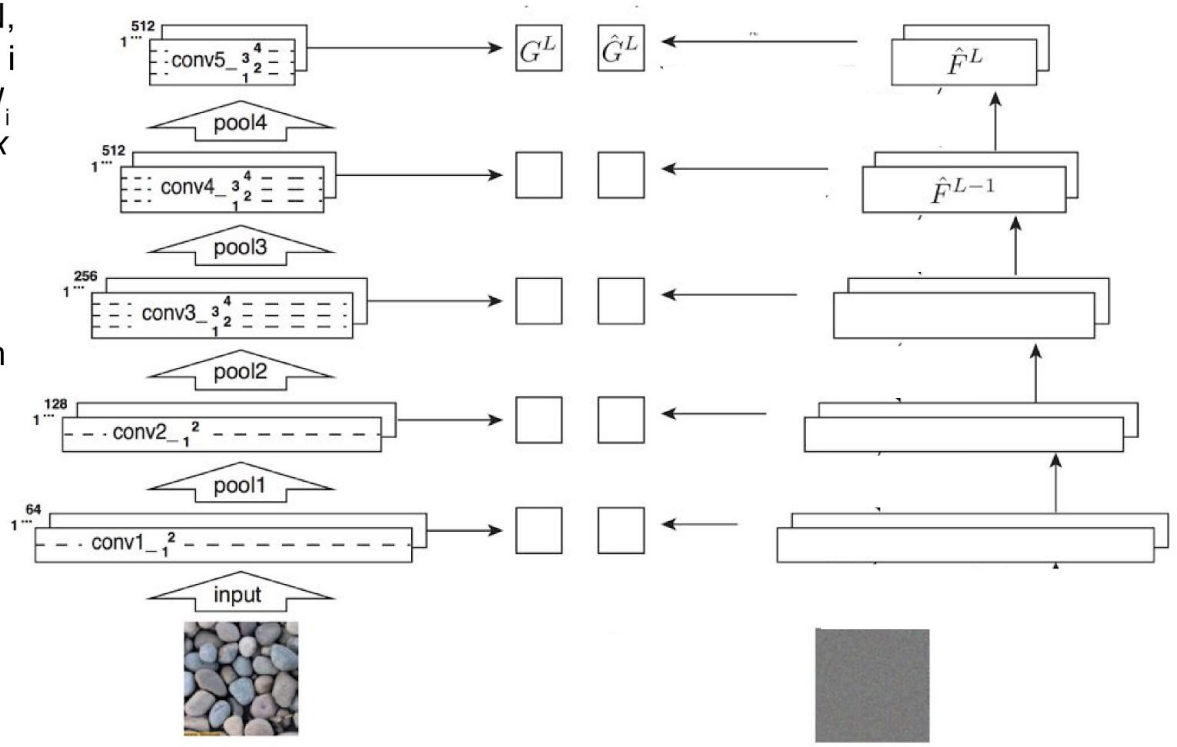
Texture Synthesis

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^l - \hat{G}_{ij}^l\right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \text{ (shape } C_i \times C_i\text{)}$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer



Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Gatys et al, "Texture Synthesis using Convolutional Neural Networks", NIPS 2015

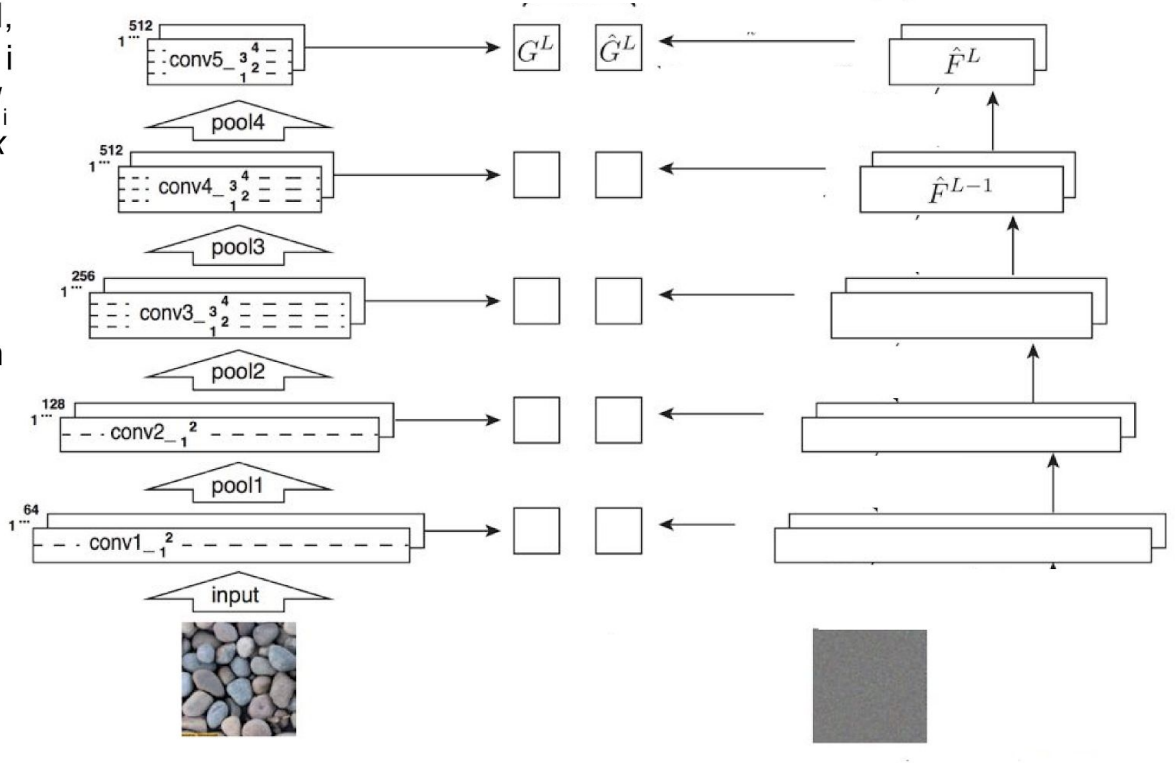
Texture Synthesis

1. Pretrain a CNN on ImageNet (VGG-19)
2. Run input texture forward through CNN, record activations on every layer; layer i gives feature map of shape $C_i \times H_i \times W_i$
3. At each layer compute the *Gram matrix* giving outer product of features:

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk} \text{ (shape } C_i \times C_i)$$

4. Initialize generated image from random noise
5. Pass generated image through CNN, compute Gram matrix on each layer
6. Compute loss: weighted sum of L2 distance between Gram matrices

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G^l_{ij} - \hat{G}^l_{ij}\right)^2 \qquad \mathcal{L}(\vec{x}, \hat{\vec{x}}) = \sum_{l=0}^{L} w_l E_l$$

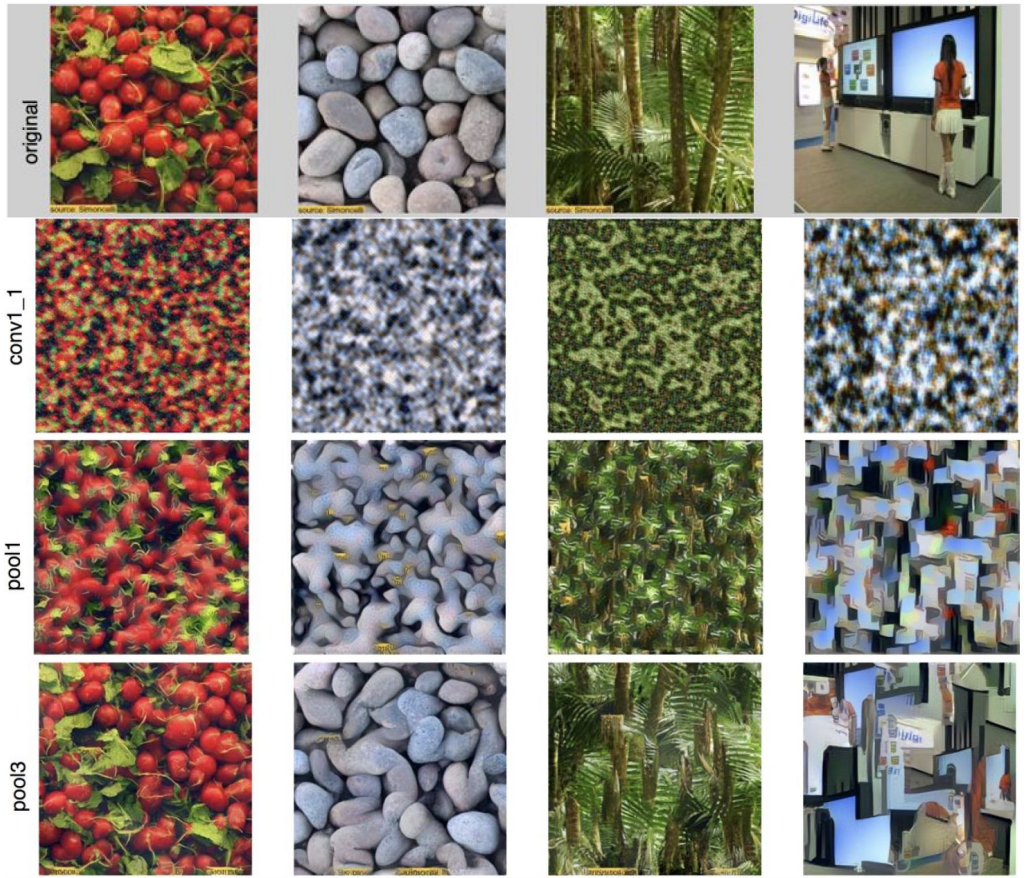


Gatys, Ecker, and Bethge, "Texture Synthesis Using Convolutional Neural Networks", NIPS 2015

Gatys et al, "Texture Synthesis using Convolutional Neural Networks", NIPS 2015

Texture Synthesis

- Reconstructing from higher layers recovers larger features from the input texture



Gatys et al, “Texture Synthesis using Convolutional Neural Networks”, NIPS 2015

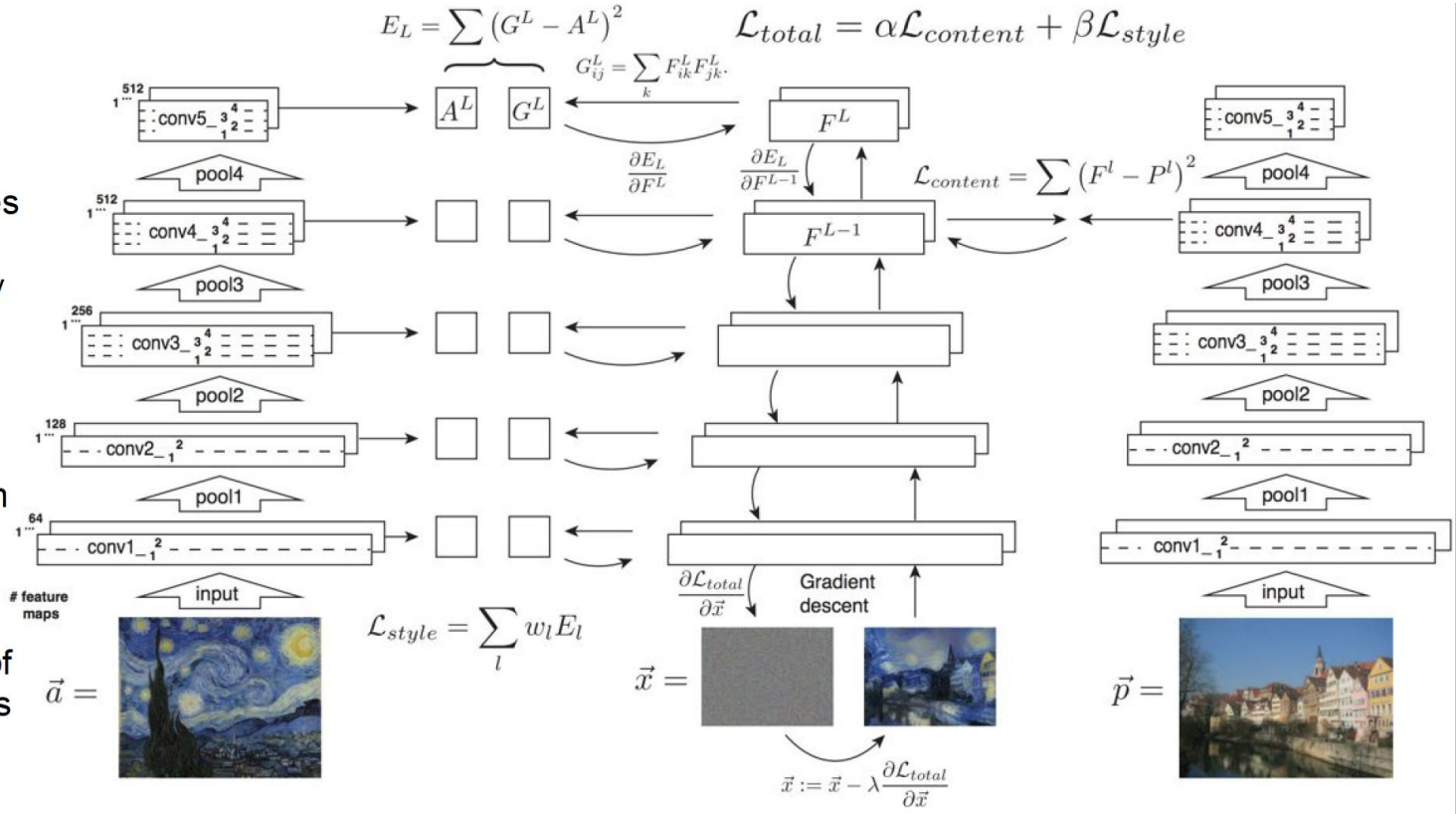
Neural Style Transfer

- Given a content image and a style image, find a new image that
  - Matches the CNN features of the content image (feature reconstruction)
  - Matches the Gram matrices of the style image (texture synthesis)
- Combine feature reconstruction from Mahendran et al with Neural Texture Synthesis from Gatys et al, using the same CNN!

Gatys et al, “A Neural Algorithm of Artistic Style”, arXiv 2015
Gatys et al, “Image Style Transfer using Convolutional Neural Networks”, CVPR 2016

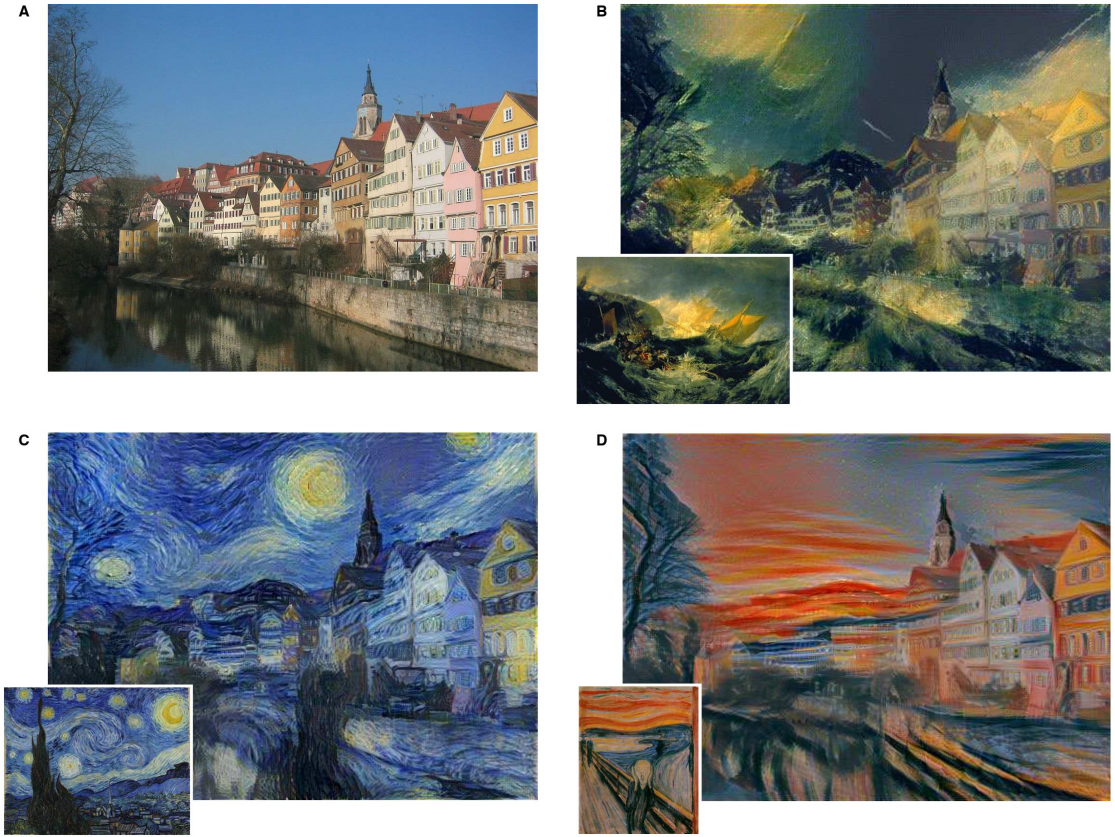
Neural Style Transfer

1. Pretrain CNN
2. Compute features for content image
3. Compute Gram matrices for style image
4. Randomly initialize new image
5. Forward new image through CNN
6. Compute style loss (L2 distance between Gram matrices) and content loss (L2 distance between features)
7. Loss is weighted sum of style and content losses
8. Backprop to image
9. Take a gradient step
10. GOTO 5



Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016

Neural Style Transfer



Gatys et al, “Image Style Transfer using Convolutional Neural Networks”, CVPR 2016

Neural Style Transfer

More weight to content loss

More weight to style loss

Gatys et al, “Image Style Transfer using Convolutional Neural Networks”, CVPR 2016

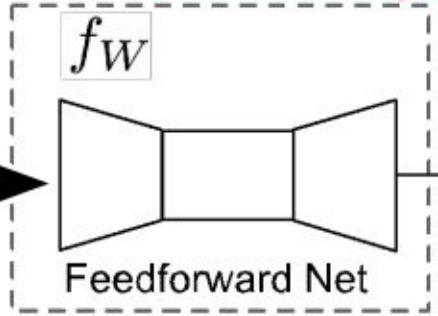
Fast Style Transfer

Problem: Style transfer is slow; need hundreds of forward + backward passes of VGG

Solution: Train a feedforward network to perform style transfer!

Fast Style Transfer

(1) Train a feedforward network for each style
(2) Use pretrained CNN to compute same losses as before
(3) After training, stylize images using a single forward pass



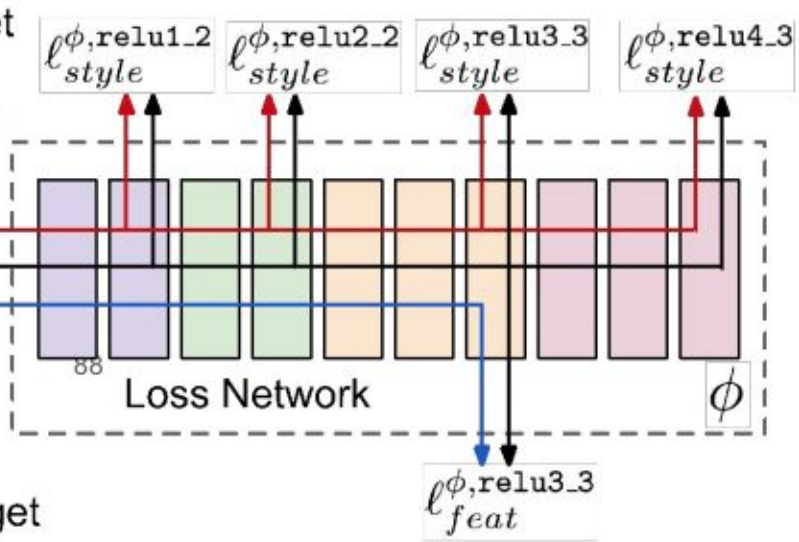
Style Target

y_s

$\hat{y}$

y_c

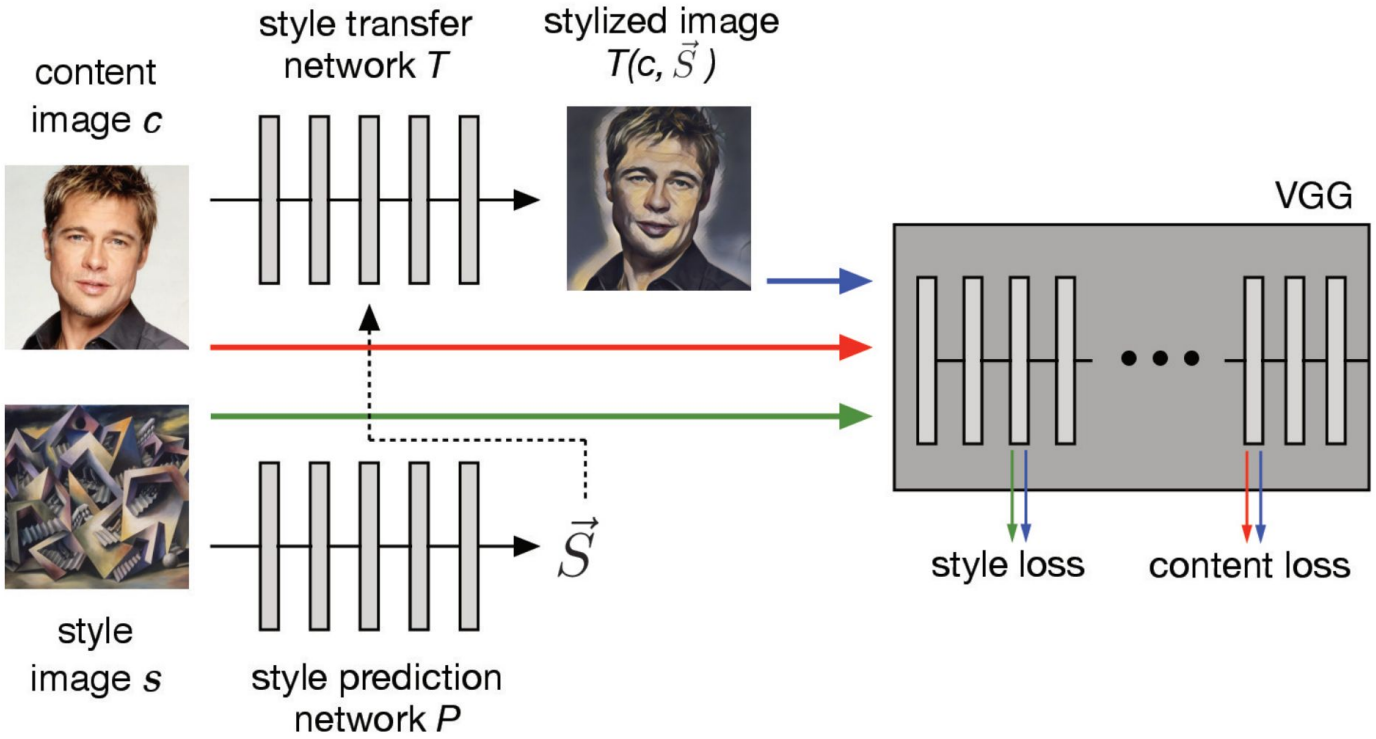
Content Target



Works real-time at test-time!

Johnson et al, "Perceptual Losses for Real-Time Style Transfer and Super-Resolution", ECCV 2016

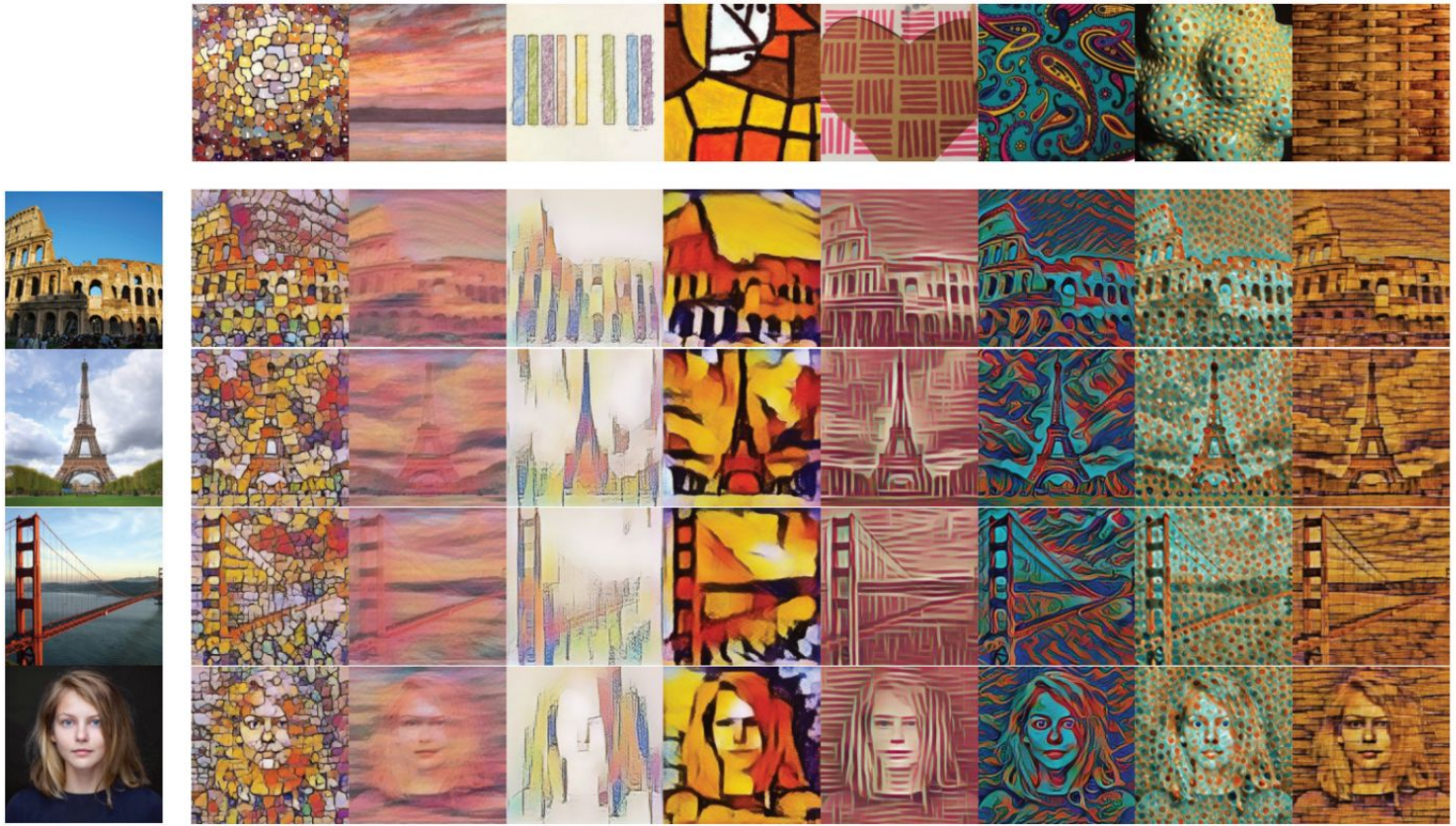
Real-time Arbitrary Style Transfer



Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., & Shlens, J. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *BMVC 2017*

Huang, X., & Belongie, S. Arbitrary style transfer in real-time with adaptive instance normalization. *ICCV 2017*

Real-time Arbitrary Style Transfer



Ghiasi, G., Lee, H., Kudlur, M., Dumoulin, V., & Shlens, J. Exploring the structure of a real-time, arbitrary neural artistic stylization network. *BMVC*

Summary

- Basic Principles of NNs in Supervised Learning
  - Define architecture by considering input/outputs
  - Define loss function properly
- Case studies
  - Image Classification: AlexNet, InceptionNet, ResNet, etc.
  - Object Detection: YOLO, RCNN, Faster RCNN
  - Segmentation: FCN, DeconvNet, DeepLab, Mask RCNN
  - Pose Estimation: DeepPose, Mask-RCNN
  - Style Transfer