# Vision

Arthur J. Redfern
arthur.redfern@utdallas.edu

# Outline

- Motivation
- Images
- Image classification
- Pixel classification
- Object detection
- Object segmentation
- Depth estimation
- Motion estimation
- References

# Motivation

# A Brief History Of Classical Vision

- Pre processing
- Feature extraction
  - Hand engineered for region candidates
  - Hand engineered for objects
  - Hand engineered for depth
  - Hand engineered for motion
- Prediction
  - Trainable classifier
- Post processing

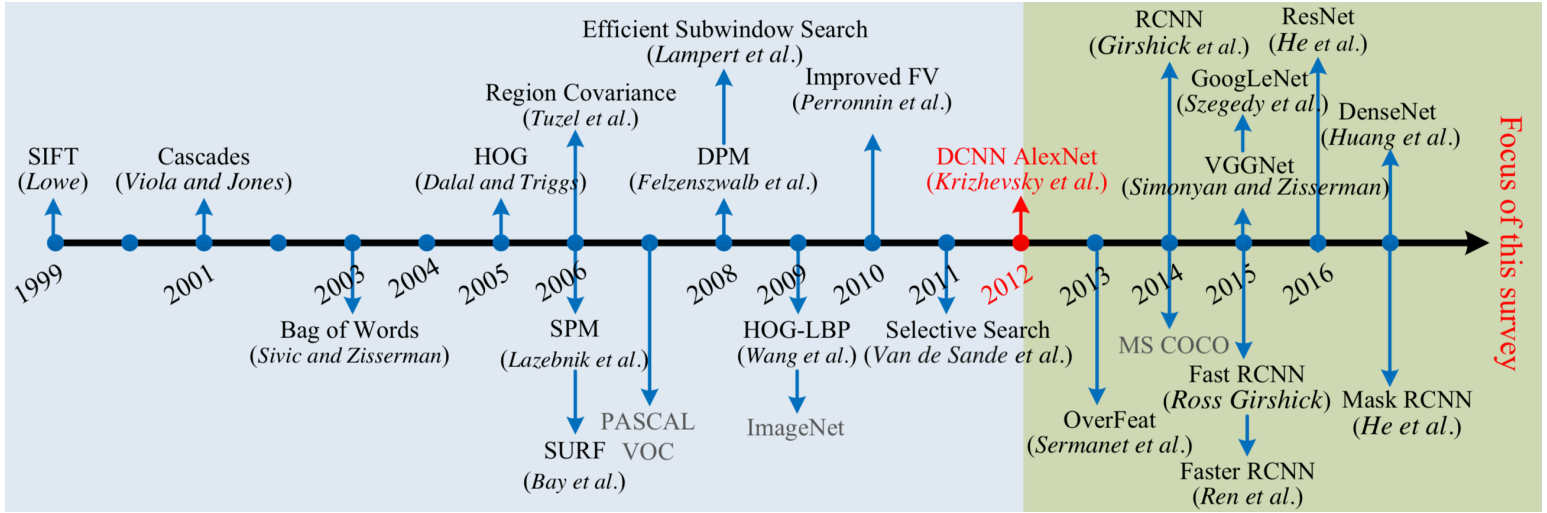



Figure from https://arxiv.org/abs/1809.02165

# Image Understanding Is A Classification Problem

## Object detection

- Input: Image
- Box each object
- Classify the object in each box


## Object segmentation

- Input: Image
- Segment each object

Figures from Pascal VOC (http://host.robots.ox.ac.uk:8080/pascal/VOC/) and Microsoft COCO (http://mscoco.org)

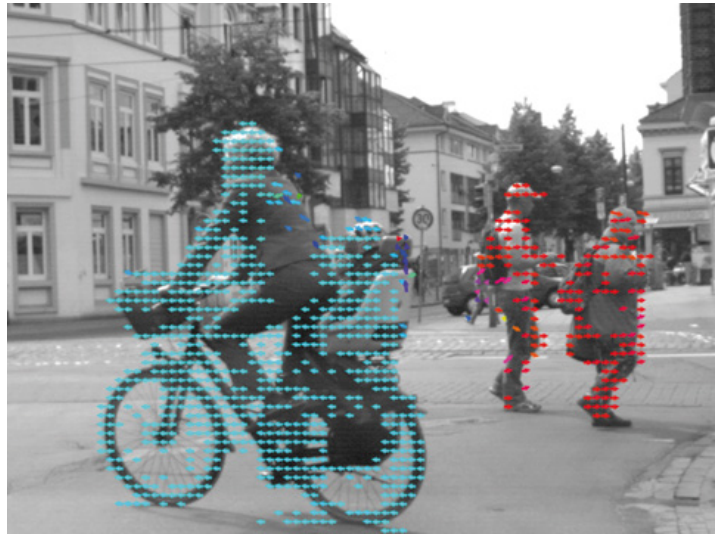
# Structure Estimation Is A Classification Problem

## Depth estimation

- Input:  2+ images with spatial separation
- Preprocessing (rectification)
- Cost (find the same point in 2+ images = classification)
- Depth estimation (smoothing, SGM, ...)

## Motion estimation

- Input:  2+ images at separate time instants
- Preprocessing (rectification)
- Cost (find the same point in 2+ images = classification)
- Motion estimation (refinement, ...)



Figures https://ieeexplore.ieee.org/document/6327192 and https://www.commonvisionblox.com/en/products/series/cvb-optical-flow/

# The Strategy Described Here

- Replace hand engineered feature extraction and prediction with xNN based methods
  - xNNs are universal approximators
  - Most vision problems can be cast as approximating a mapping from pixels to classes
  - Typically use CNNs to take advantage of spatial invariance to reduce compute
  - Train end to end
- Network design and training
  - Frequently start from a base classification network trained on ImageNet
  - Remove the classification head, add a head appropriate for the problem, restart training

# Disclaimer

- There's a lot of vision related stuff not included here
  - Different methods within the categories of problems included here
  - Problems that are not included here
- Possibly some of this will be addressed in future versions of the slides
- Regardless of whether it is or not, hopefully these slides provide enough of a base from which to branch off and learn more on your own

Figure from https://papers.nips.cc/paper/8061-deep-network-for-the-integrated-3d-sensing-of-multiple-people-in-natural-images.pdf

# Images

# Understand The Data Available For Processing

- Understanding how images are captured is helpful for understanding vision
  - It's the data the vision algorithm has to work with
  - Apertures, lens, sensors, image signal processing pipelines, ...

- But the realities of a 1 semester course, the content we've covered so far and the content we have left to cover makes it impractical to spend a lot of time on this

- So the following are a few of many excellent links for more information on the image capture process
  - Brown CSCI 1290: computational photography and image manipulation
    - http://cs.brown.edu/courses/csci1290/
    - Links to additional courses are at the bottom of the page
  - Digital photography
    - YouTube links: https://sites.google.com/site/marclevoylectures/
    - PDFs: https://sites.google.com/site/marclevoylectures/schedule
    - Lenses and optics: https://drive.google.com/file/d/0B6w2XfPHEcZYT0dwaFF4emd6Vzg/view
      https://drive.google.com/file/d/0B6w2XfPHEcZYTnZIR2VpWThHM0U/view
    - Post processing: https://drive.google.com/file/d/0B6w2XfPHEcZYeTFKZWhndVBOYTA/view

# For Humans Or Machines?

- One thing to think about is that the majority of the design and processing of image capture devices is to create images that are pleasing in some way to humans
  - It's also the majority of the data that we have to work with
- But for vision we want images that are optimized for algorithms to extract information
  - Is there a better tuning of the image capture process for this?

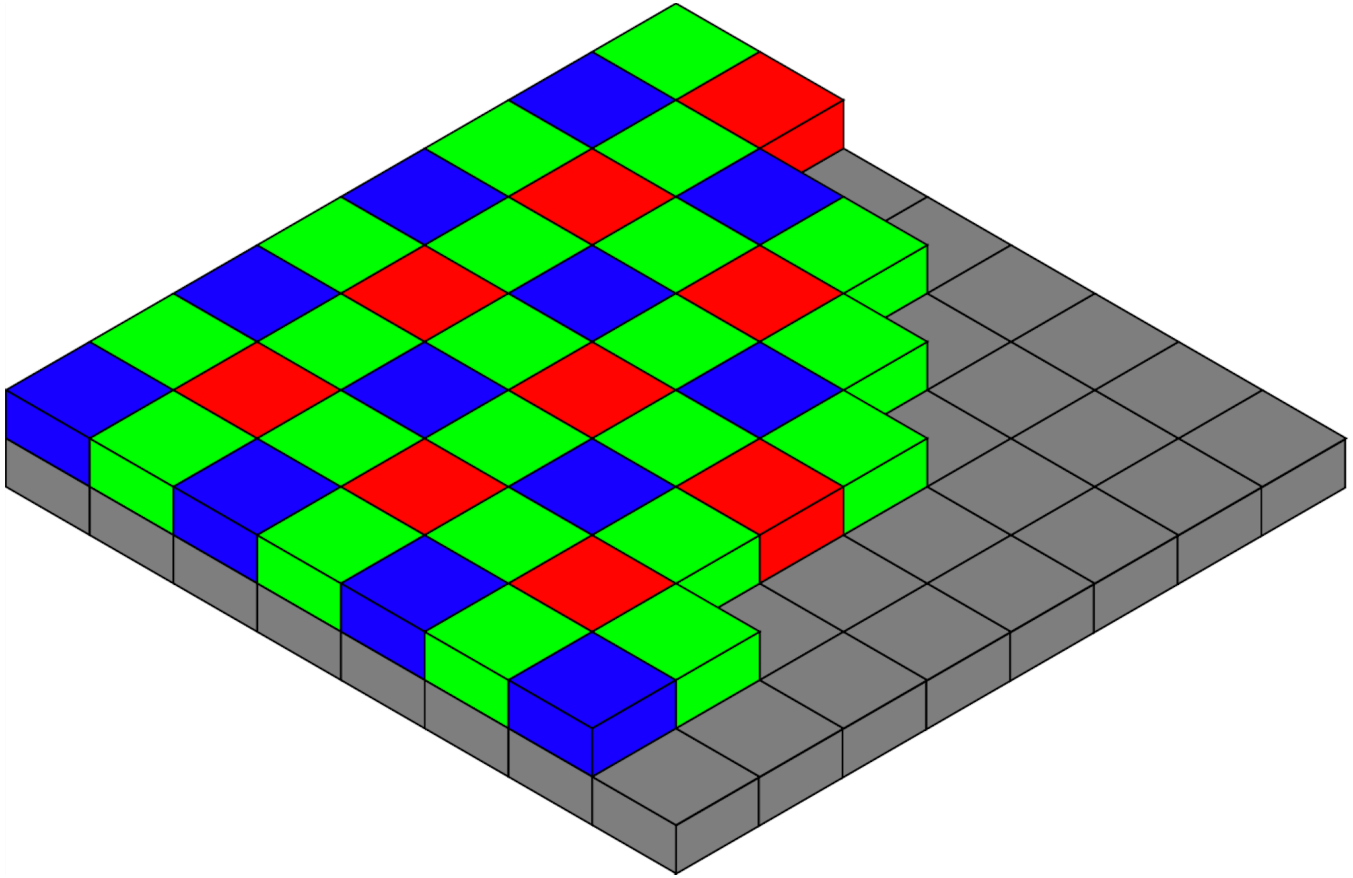


Figure from https://en.wikipedia.org/wiki/Bayer_filter

# Example ISP: ARM Mali Camera

Configurable ASIC style

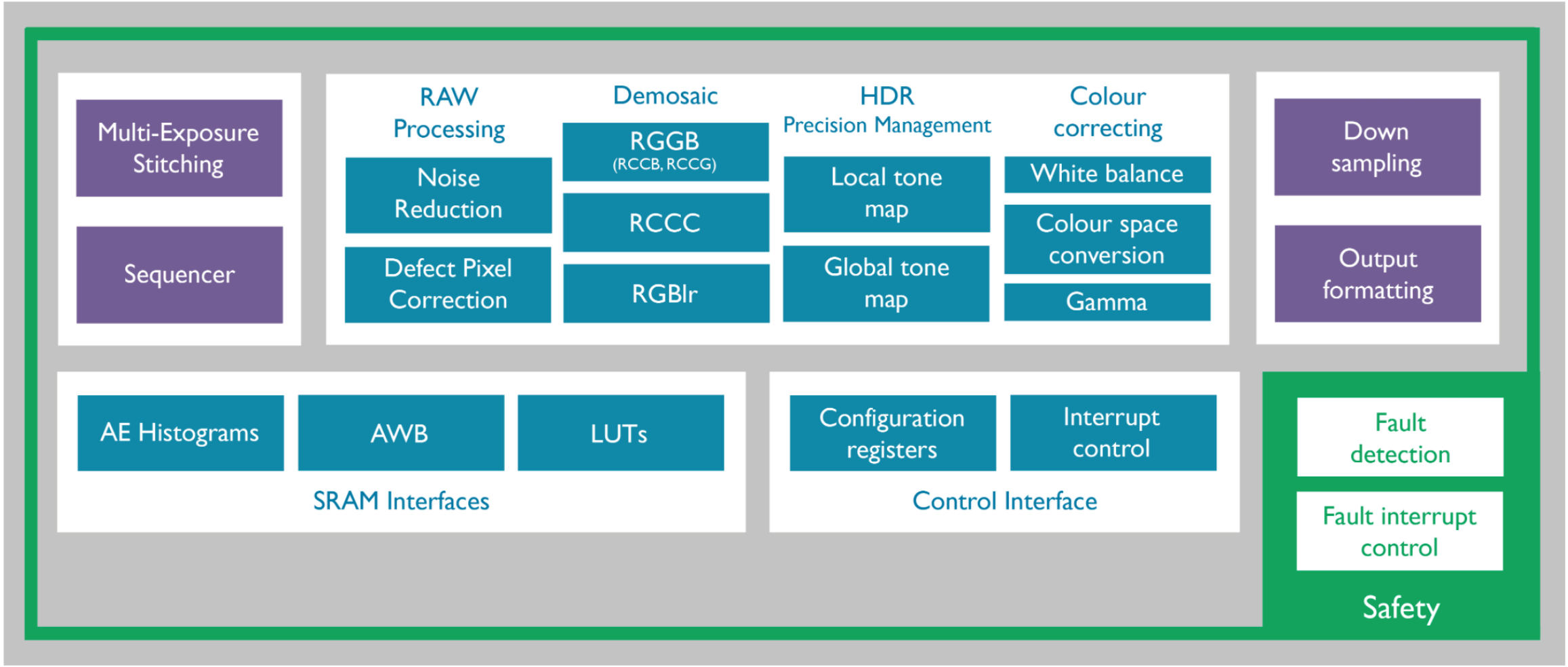



Figure from https://developer.arm.com/products/graphics-and-multimedia/mali-camera

# Example ISP: Google Pixel Visual Core

- Domain specific architecture (DSA)
  - Inputs and outputs are in DRAM
  - Ring network between IPU cores and DMA
  - IPU is a line buffer + a stencil processors
  - Internal data is in line buffers
  - Stencil processors act on 16x16 compute regions within a 20x20 array (excess is halo) and perform 16b vector MAC operations

- Domain specific software (no abbreviation)
  - Program in a subset of Halide (http://halide-lang.org)
  - Halide backend creates an architecture independent intermediate representation
  - Final compilation into architecture specific instructions
  - Includes explicit memory movement

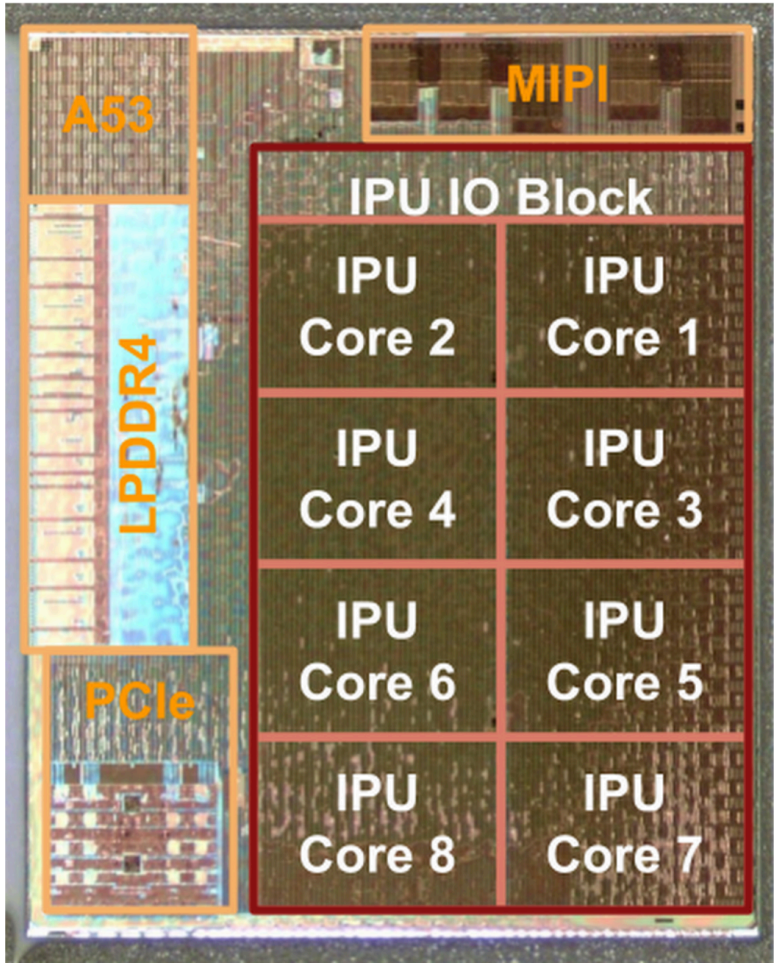



Figure from https://www.blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/

# Human Understanding

- Will perhaps be added to a future version

- While there's no need to copy what the human visual system does, at the same time it could be useful to understand it a little better as an existence proof of what's possible with that architecture

# Image Classification

# Goal

- Image classification assigns a single label to a whole image corresponding to the dominant object in the image

Bald eagle

Figure from http://image-net.org

# Data

- General
  - MNIST
  - Fashion MNIST
  - CIFAR
  - Street view house numbers
  - Tiny ImageNet
  - ImageNet
  - Open images dataset V4
  - Google landmarks V2
  - …
- Proprietary
  - Note: assume that all large companies with an AI focus have internal classification datasets that are much much larger than ImageNet

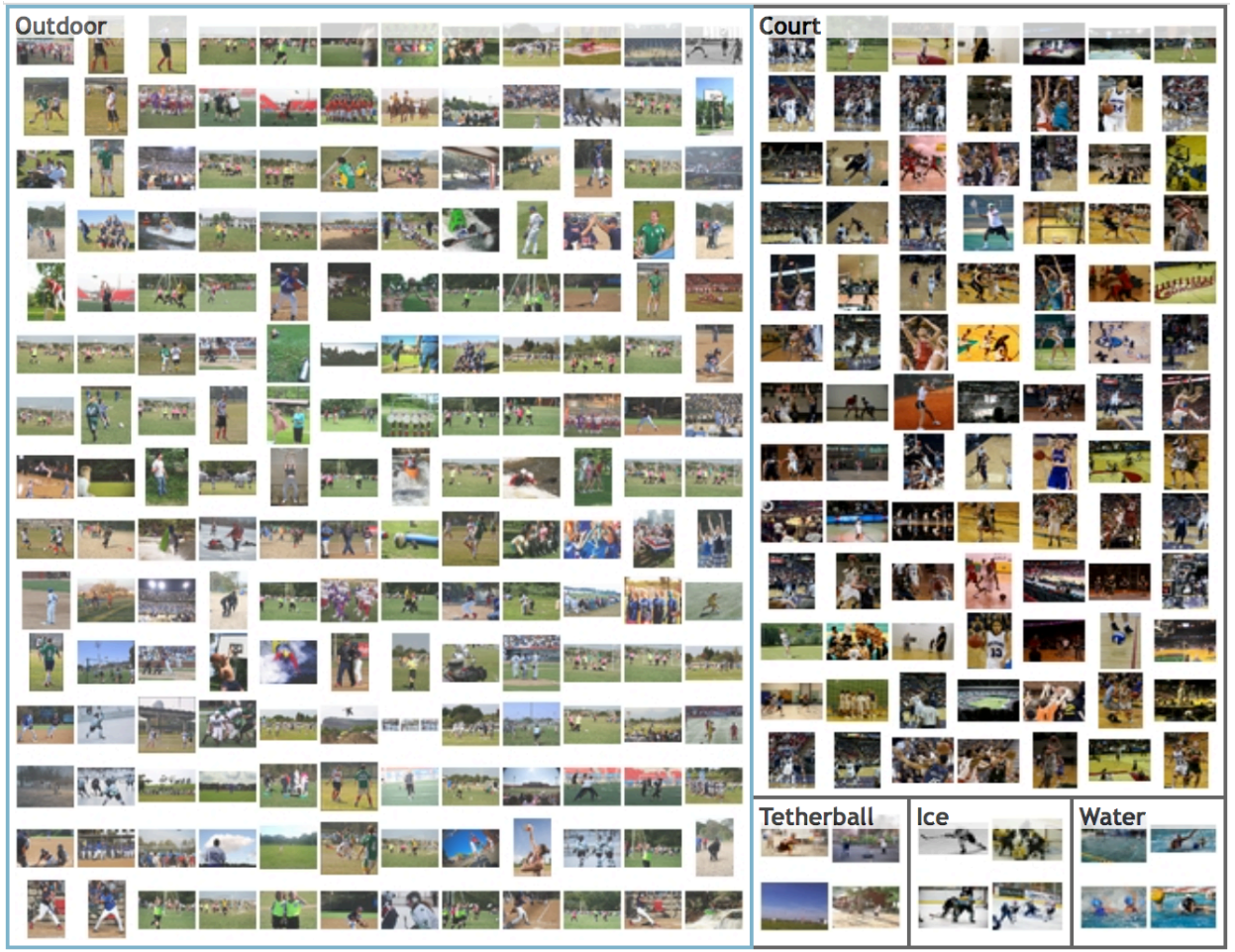



Figure from http://image-net.org/explore.php

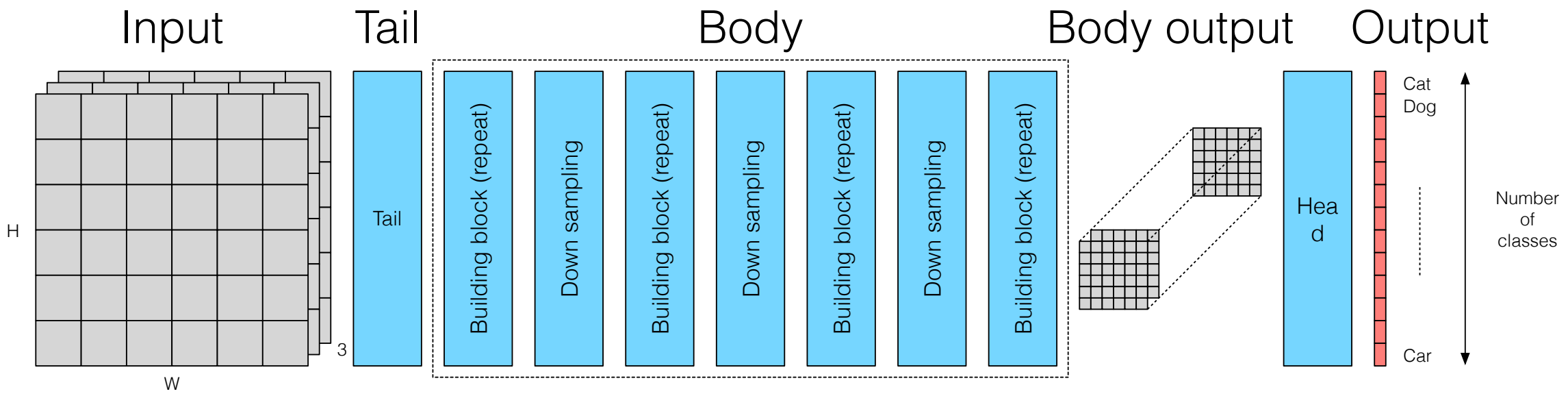
# Pre Processing

- Common operations
  - Resizing
    - Not because of convolutional layers
    - Because of fully connected layers
  - Cropping
  - Normalizing to 0 mean and 1 variance

- Example pre processing used by PyTorch for networks applied to ImageNet
  - Resize image such that the shorter side is 256
  - Crop to 224 x 224
  - Flip BGR to RGB
  - Normalize each channel by diving by 255
  - Subtract mean
    - [0.485, 0.456, 0.406]
  - Divide by standard deviation
    - [0.229, 0.224, 0.225]

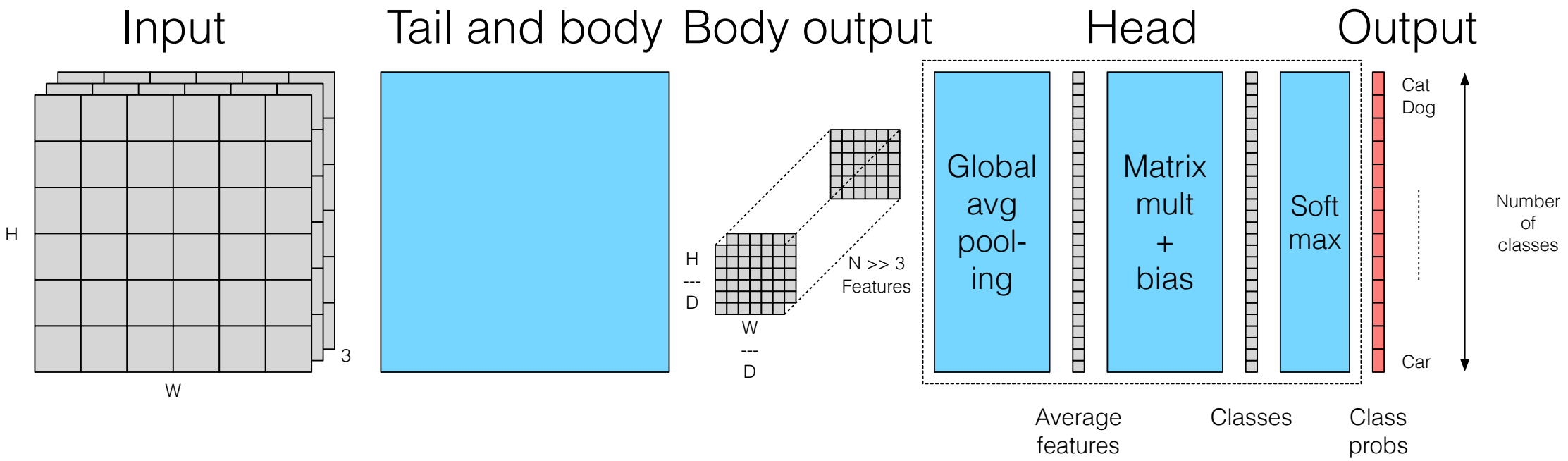
# Feature Extraction

See the network design slides for tail and body design examples



# Classification

See the network design slides for head design examples



# Post Processing

- To win a contest use an ensemble of
  - Different networks or
  - The same network trained different ways
- Distillation is a method for compressing the knowledge of multiple networks into a single network
  - Distilling the knowledge in a neural network
  - https://arxiv.org/abs/1503.02531
- Other than using ensembles and / or distillation, there's not really a lot to do

# Evaluation

- Top N accuracy
  - Network (if softmax was included) predicts a pmf for the different classes
  - Top N accuracy means the correct class was in the top N highest pmf values

- Uses
  - Top N = 1 is frequently used for datasets with a small number of classes
  - Top N = 5 is frequently also considered for datasets with a large number of classes with a high level of similarity (e.g., 2 dogs that look alike)

# Pixel Classification

# Goal

- Pixel classification: label every pixel in the image with the class that it belongs to
- Usually called semantic segmentation
- For a 3 x 512 x 1024 input image, that would mean labeling 512*1024 = 524288 pixels
- Note: sometimes the goal is modified to label every pixel in a lightly down sampled version of the image
  - Ex: (512/4) x (1024/4) down sampled image requiring 128*256 = 32768 pixels to be labeled

Figure from https://www.cityscapes-dataset.com overlaid with coloring

# Data

- Laundry list
  - Cityscapes
  - Pascal VOC
  - NYUDv2
  - SIFT Flow
  - CamVid
  - SUN RGB-D
  - ADE20K
  - ScanNet
- Note
  - It's possible to use the object segmentation datasets for generating labeled data for pixel segmentation
  - It's also possible to use object detection bounding boxes to improve training

Figure from https://www.cityscapes-dataset.com overlaid with coloring

# Feature Extraction

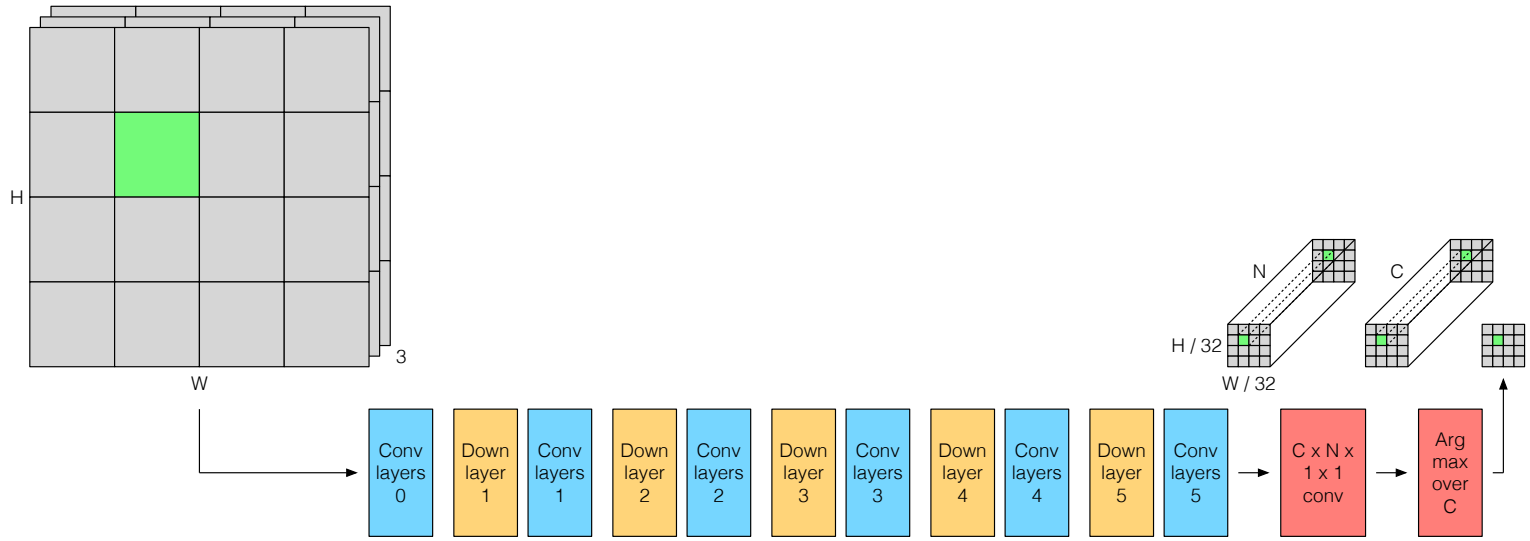
- How does feature extraction for pixel classification compare to feature extraction for classification?
  - Want strong features for prediction (like classification)
  - Want spatially accurate features for localization (new requirement)
  - Note: this same question also applies to object detection, object segmentation, depth estimation, motion estimation, ...

- Problem
  - Deep strong features that are good for prediction tend to be bad for localization
  - Shallow weak features that are good for localization tend to be bad for prediction

- The receptive field size is 1 more thing to keep track of
  - Shallower better localized features tend to have smaller receptive field sizes
  - Deeper worse localized features tend to have larger receptive field sizes
  - Scaling the size of the input image scales the size of the objects relative to the receptive field sizes
  - Classification typically has a global average pooling operation at the end to partially alleviate this problem, methods requiring good localization typically do not

# A Conceptual Starting Point

- Pixel classification at 1/32 row and 1/32 col spatial resolution
  - Start with a classification network with 5 levels of down sampling probably pre trained on ImageNet
  - Remove the head (global avg pool and fully connected layer)
  - Add a CNN style 2D convolution layer with C x N x 1 x 1 filters and a soft / arg max in place of ReLU (C = number of output classes)
  - Effectively this layer is going to learn a linear transformation from a N x 1 vector to a C x 1 vector that's the same for every pixel in the feature map
  - Resume training on the new data set with labels at 1/32 the resolution in each dimension
  - Need to decide if the labels should be 1 hot based on the dominant object in the 32 x 32 region in the original image (so leaning a mapping to a 1 hot vector) or a pmf representing the class probability of all objects in the 32 x 32 region (so learning a mapping to a pmf)

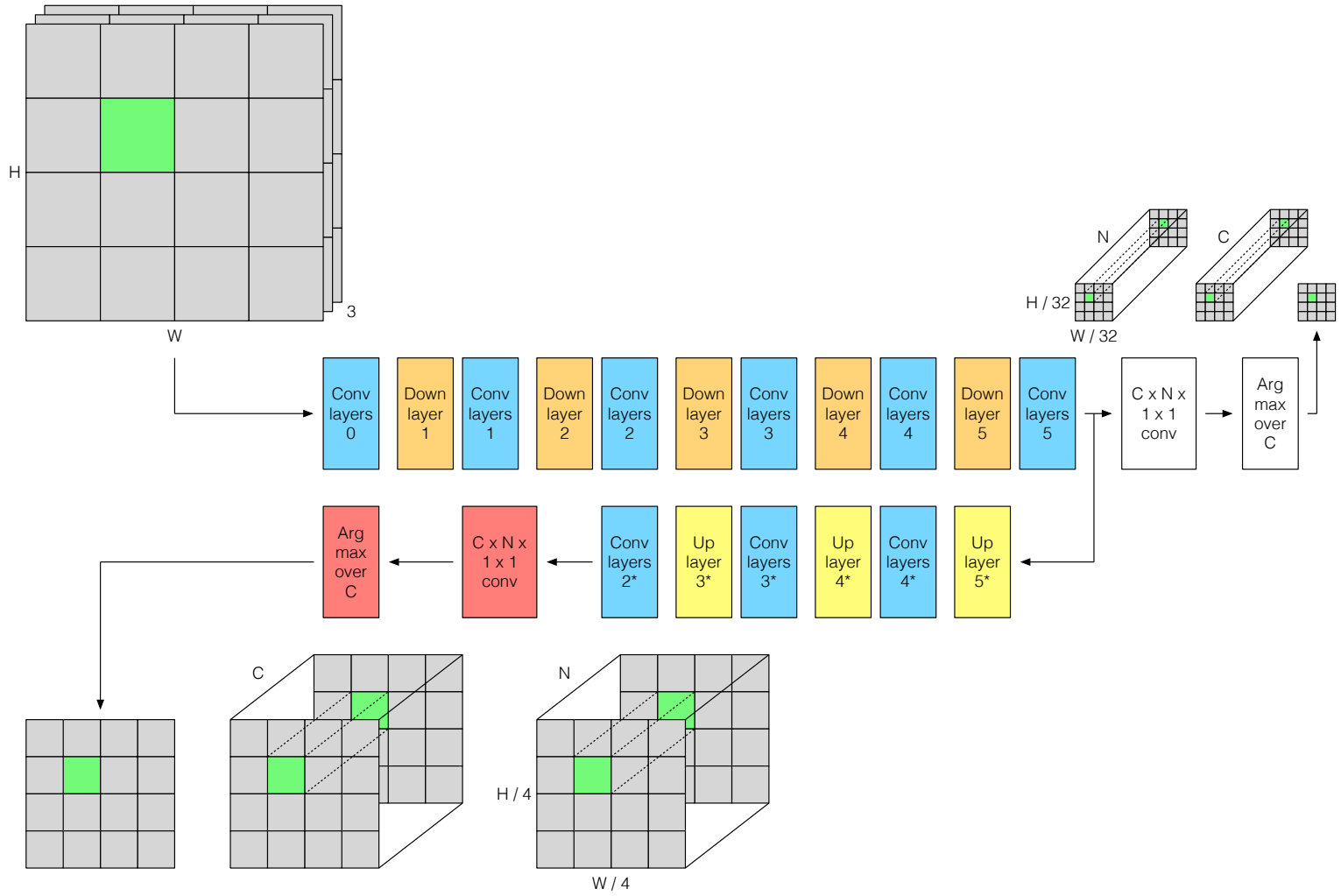


# A First Idea For Improving Localization

- So the question is how do you improve the localization of pixel classes from 1/32 in each dimension

- One option would be to up sample the final output feature map before applying the new head
  - Perhaps do this all at once
  - Perhaps do this in stages with extra transformations in between
  - Perhaps reduce N in the process of the transformations
  - Maybe not to the full image resolution but 1/4 the image resolution
  - Maybe all of this helps a little

- But is there a better way to create strong spatially localized features?



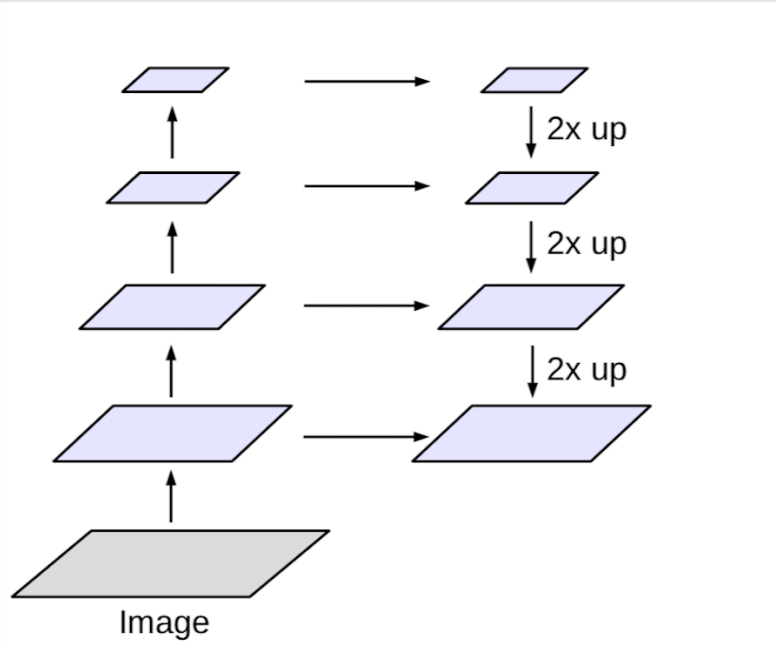
# Strong Spatially Localized Features

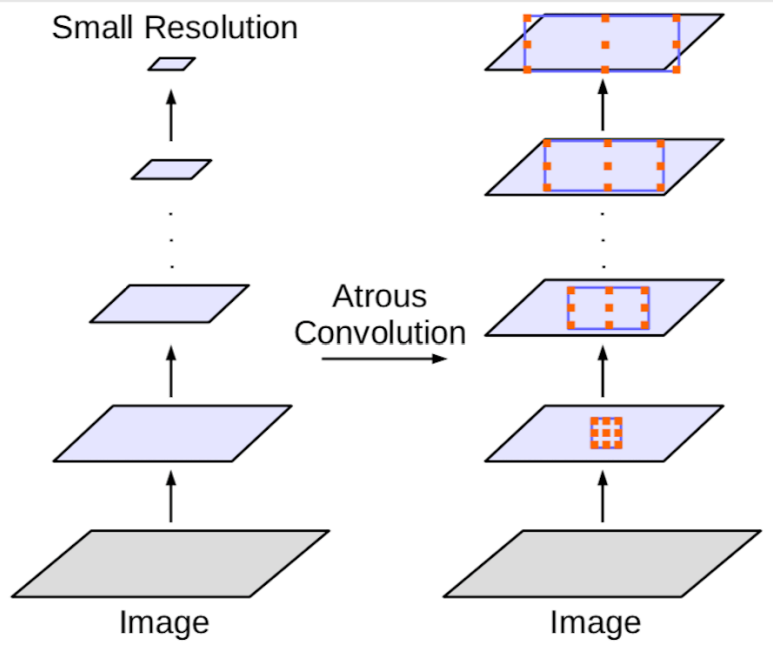
- Strategies for creating strong spatially localized features tend to fall into 1 of 2 classes
  - Create strong features without down sampling
  - Combine strong poorly localized features with weak well localized features

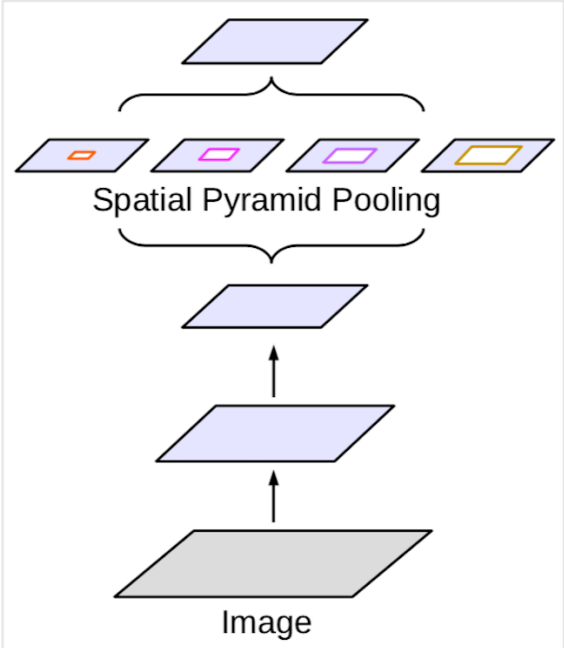
- In practice networks can use both

# Strong Spatially Localized Feature Examples

The encoder decoder with skip connections combines strong feature




## Encoder decoder with skip connections

(combine strong poorly localized features with weak well localized features via concat (typically) or add (occasionally) then convolve to combine)

## Atrous convolution

(instead of down sampling the image, up sample the filter to increase the receptive field size)

## Spatial pyramid pooling

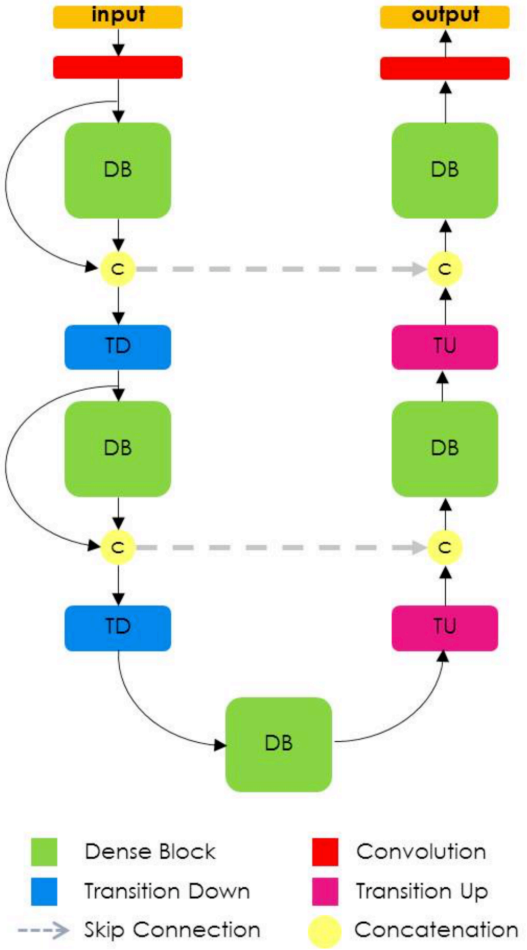
(instead of down sampling the image, use pooling to increase the receptive field size)

Figure from https://arxiv.org/abs/1706.05587

# Example: The One Hundred Layers Tiramisu

Encoder decoder with skip connections

| Architecture |
|---|
| Input, $m = 3$ |
| $3 \times 3$ Convolution, $m = 48$ |
| DB (4 layers) + TD, $m = 112$ |
| DB (5 layers) + TD, $m = 192$ |
| DB (7 layers) + TD, $m = 304$ |
| DB (10 layers) + TD, $m = 464$ |
| DB (12 layers) + TD, $m = 656$ |
| DB (15 layers), $m = 896$ |
| TU + DB (12 layers), $m = 1088$ |
| TU + DB (10 layers), $m = 816$ |
| TU + DB (7 layers), $m = 578$ |
| TU + DB (5 layers), $m = 384$ |
| TU + DB (4 layers), $m = 256$ |
| $1 \times 1$ Convolution, $m = c$ |
| Softmax |



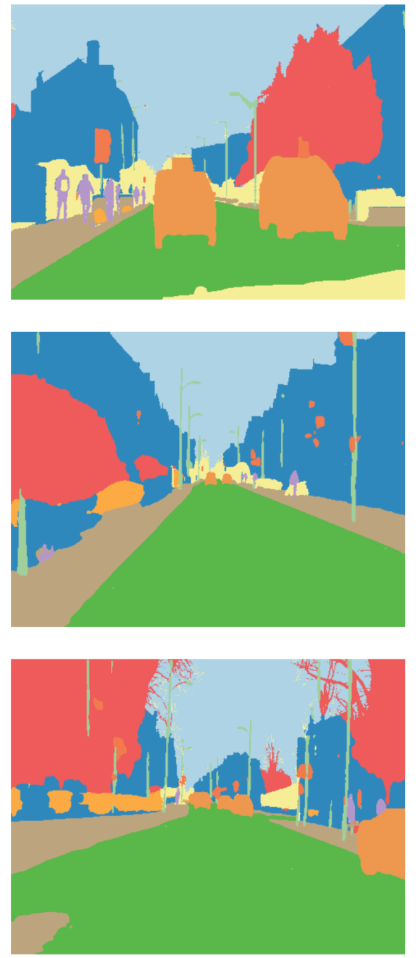



Figure from https://arxiv.org/abs/1611.09326

# Example: Pyramid Scene Parsing Network

Uses atrous convolution and pyramid pooling

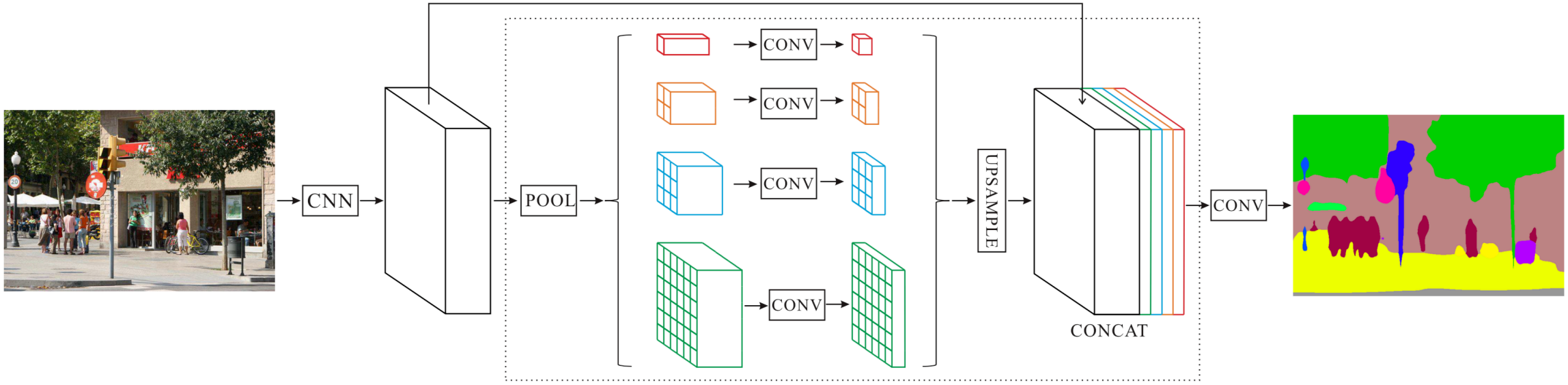



Figure from https://arxiv.org/abs/1612.01105

# Example: The Kitchen Sink

Encoder decoder with skip connections, atrous convolution and spatial pyramid pooling

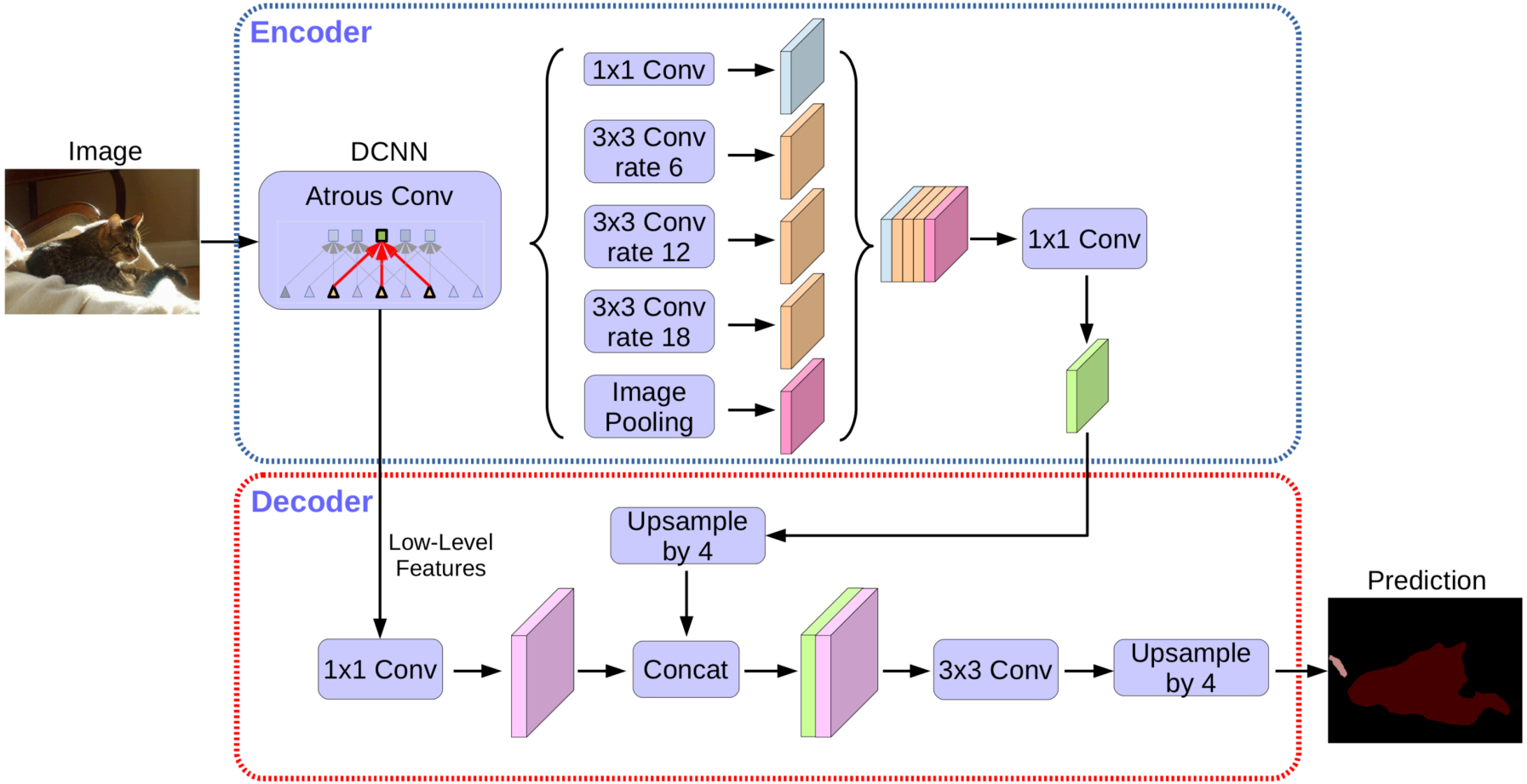



Figure from https://arxiv.org/abs/1802.02611

# Demo: Cityscapes Pixel Classification

A modified ResNeXt encoder + custom sequential class based decoder with skip connections

# Extending SE Style Improvement To Spatial

- Thought train
  - The SE module (see the network design lecture) re weighted channels based on global information to improve classification
  - For pixel classification, object detection, object segmentation, ... we ideally need a more spatially diverse re weighting
  - A number of papers have ideas along these lines
  - The below is a non comprehensive laundry list

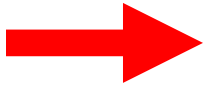
- Concurrent spatial and channel squeeze & excitation in fully convolutional networks
  - https://arxiv.org/abs/1803.02579
- CBAM: convolutional block attention module
  - https://arxiv.org/abs/1807.06521

# Some Additional References

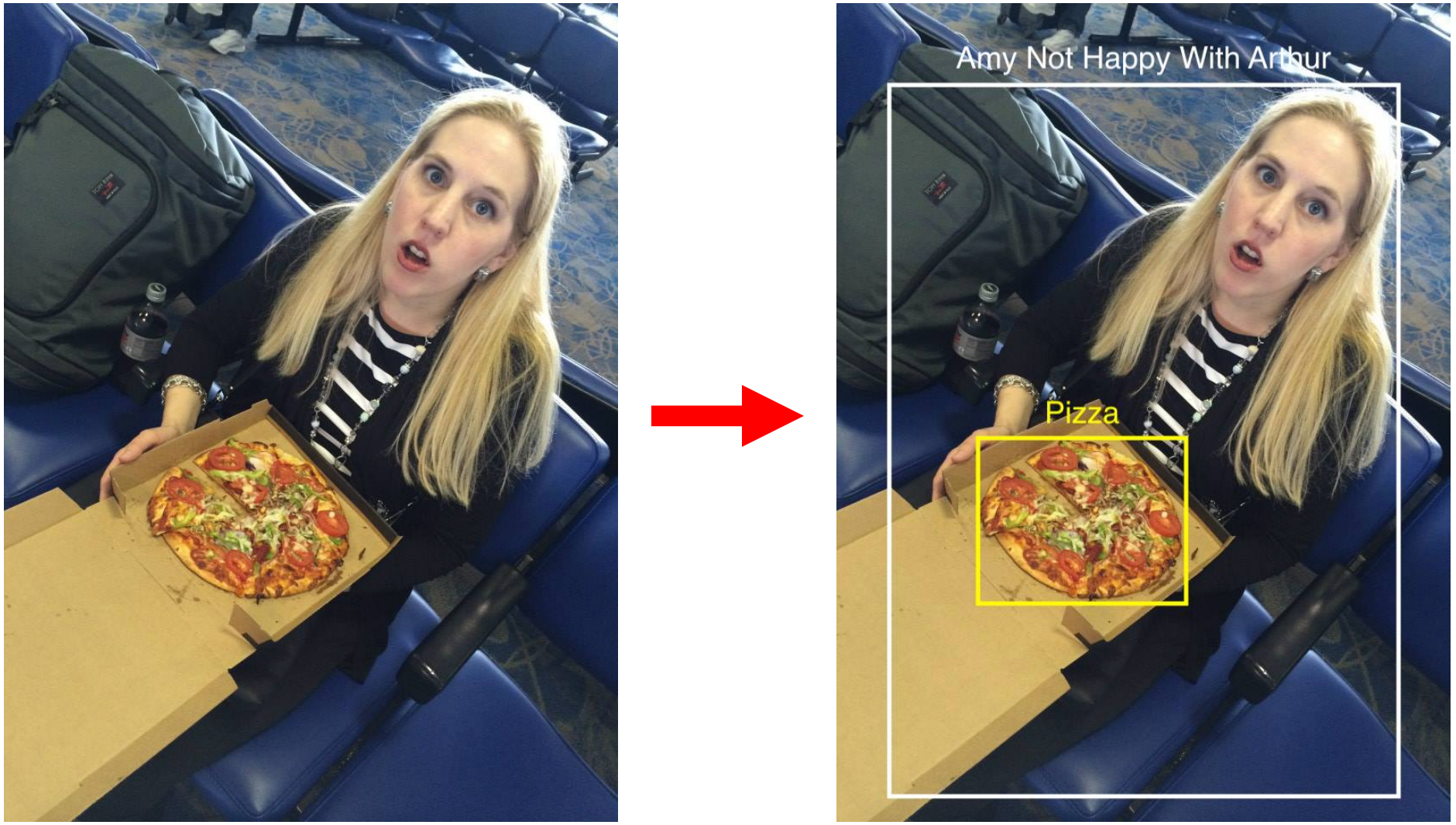
- Rethinking Atrous convolution for semantic image segmentation
  - https://arxiv.org/abs/1706.05587
  - DeepLab V3
- Decoders matter for semantic segmentation: data-dependent decoding enables flexible feature aggregation
  - https://arxiv.org/abs/1903.02120
- Searching for MobileNetV3
  - https://arxiv.org/abs/1905.02244
  - See figure 10 for Atrous convolution with SE style multiplicative re weighting and skip connections

# Object Detection

# Disclaimer: Not All Labels Are This Descriptive



# When You’re Eating Pizza In The Charlotte Airport Before Flying To Italy



# Goal

- Draw a box around every object in the image and classify the object in the box

- Localization and classification
  - Localization can be cast as a classification problem
    - Cover an image with many potential (anchor) boxes at different locations with different shapes and sizes
    - Classification problem: does a potential (anchor) box contain an object or not
    - Regression problem: refine the sizing of the potential (anchor) box
  - Classification is classifying the objects in the boxes


Figure from https://storage.googleapis.com/openimages/web/index.html

# Data

- Laundry list
  - Pascal VOC
  - Open images dataset V4
  - Street view house numbers


Figure from https://storage.googleapis.com/openimages/web/index.html

# Again Need Strong Spatially Localized Features

- Can use all of the previously mentioned methods for generating features
- Will also add a new approach: feature pyramid networks
  - Give the ability of making predictions from feature maps at different scales
  - Either separately at each scale then post process or you can think of variants that do it at all scales in a combined manner then post process

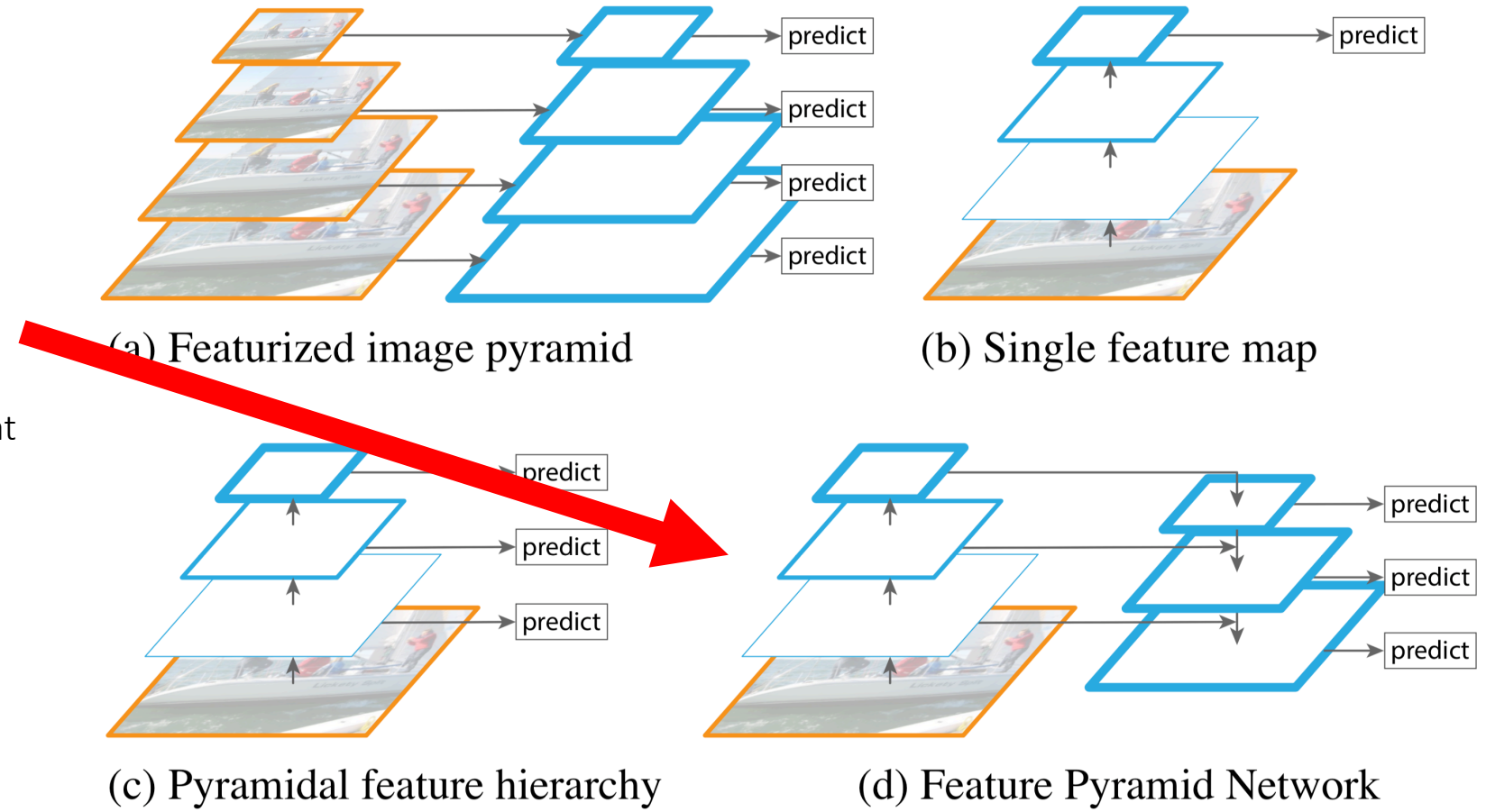
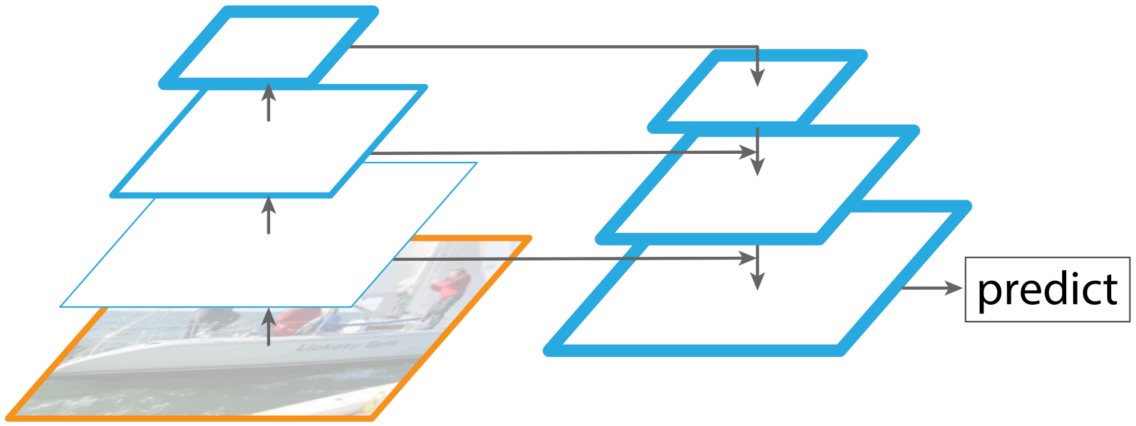


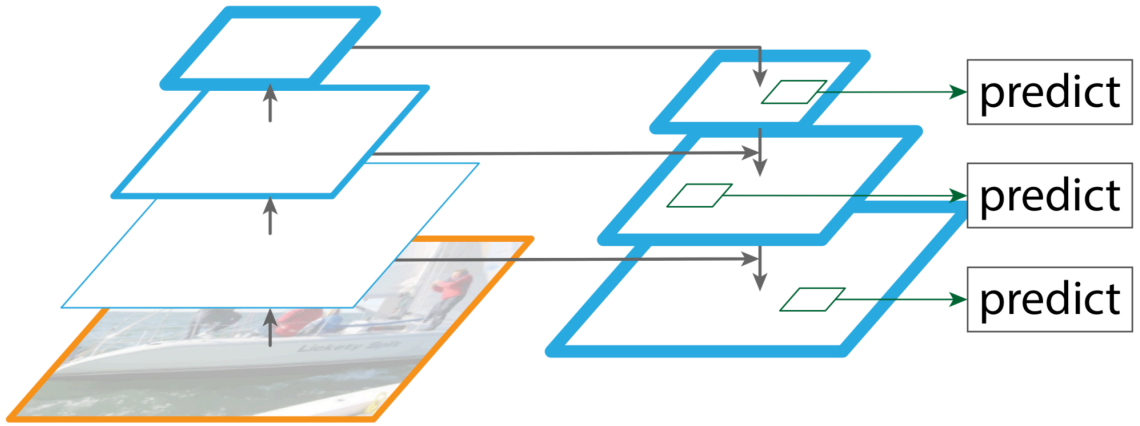

Figure from https://arxiv.org/abs/1612.03144

# Feature Pyramid Vs Skip Connections

Being able to predict boxes and classes at multiple feature map scales helps object detection handle a wide variety of object sizes; previously, in pixel classification, the purpose was only to classify at the pixel level so only the highest resolution feature map was used for prediction




Encoder decoder with skip connections




Feature pyramid

Figures from https://arxiv.org/abs/1612.03144

# 2 Strategies For Object Detection

- Strategy 1: a two stage approach (+ post processing)
  - 1. Classify anchor boxes {contains object, does not contain object} and refine / regress {Δx, Δy, Δw, Δw} anchor box locations
  - 2. Classify objects in the refined anchor boxes that contain objects (and possibly refine / regress their location again)
  - Then use post processing to combine / merge / prune results

- Strategy 2: a single stage approach (+ post processing)
  - 1. Classify anchor boxes {contains object, does not contain object}, refine / regress {Δx, Δy, Δw, Δw} anchor box locations and classify the objects in those boxes at the same time
  - Then use post processing to combine / merge / prune results

- Commentary
  - Post processing is ignored in the counting of stages but is critical for good performance
  - Typically, single stage approaches are faster than two stage approaches
  - Typically, two stage approaches are more accurate than single stage approaches
  - But there are lots of caveats in that statement that I'm glossing over

# Two Stage Object Detection History

- A historical arch of some key object detection methods (this is a really beautiful arc of papers)
  - R-CNN: rich feature hierarchies for accurate object detection and semantic segmentation
    - Selective search for region proposals, warp to constant size, conv layers, SVM head
  - SPPNet: spatial pyramid pooling in deep convolutional networks for visual recognition
    - Spatial pyramid pooling so can process full image at once covering all regions via extracting arbitrary region sizes to fixed length vectors, multi stage training that's unable to update conv layers before the SPP layer
  - Fast R-CNN
    - Differentiable RoI pooling layer (single SPP level) with end to end training, no SVM head
  - Faster R-CNN: towards real-time object detection with region proposal networks
    - Replace selective search with a region proposal network built off of the same conv layers used for classification
  - R-FCN: object detection via region-based fully convolutional networks
  - FPN:  feature pyramid networks for object detection

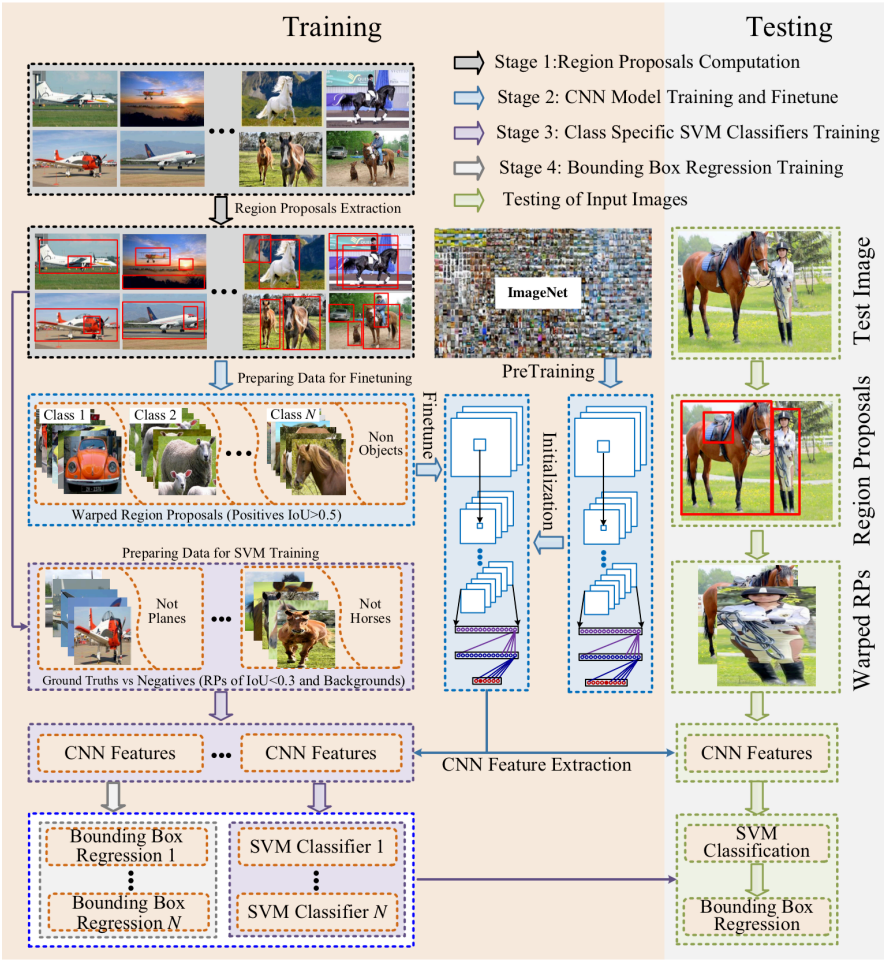



Figure from https://arxiv.org/abs/1809.02165

# Two Stage Object Detection

- Use a CNN encoder to generate strong features from an input image
  - ImageNet style feature encoders are popular
  - Perhaps stop just before the 5th level of down sampling (i.e., output feature maps resolution at 1/16 of input image)
- Find regions (boxes) in the original image that contain objects
  - Use the CNN encoded features with a region proposal network head to classify and refine anchor boxes
  - Boxes are classified as {contains object, does not contain object} and locations {Δx, Δy, Δw, Δw} are refined via regression
- Map corresponding CNN encoded features in boxes that contain objects to a single size vector
  - In general, boxes will be all sorts of different sizes and in all different locations in the input image
  - But fully connected layers and linear classifier heads work on fixed length inputs
  - So a pooling approach will be used to map each different CNN encoder region to vectors of the same size
  - The vectors can be stacked shoulder to shoulder as a matrix and processed as a batch in the next step
- Classify the vectors corresponding to pooled regions containing objects
  - This is a standard classification head working on a batch of inputs where elements of the batch correspond to different regions from the same image
  - Regression can also be done to further refine the locations

# Highlighting Efficiency

- A CNN encoder is only used once per input image
  - Encoded features are used for identifying regions of interest via a small region of interest head
  - Encoded features are used for classifying regions of interest via a small classification head

- The classification head works on all regions of interest at the same time
  - Allows batch processing for the classification of regions of interest via matrix matrix multiplication

# Anchor Boxes

Candidate boxes that enclose objects, can be uniform or optimized in size and location distribution for a specific data set; object detection methods determine if an object is present in the box, what class the object belongs to how to modify the box location and size to better enclose the object

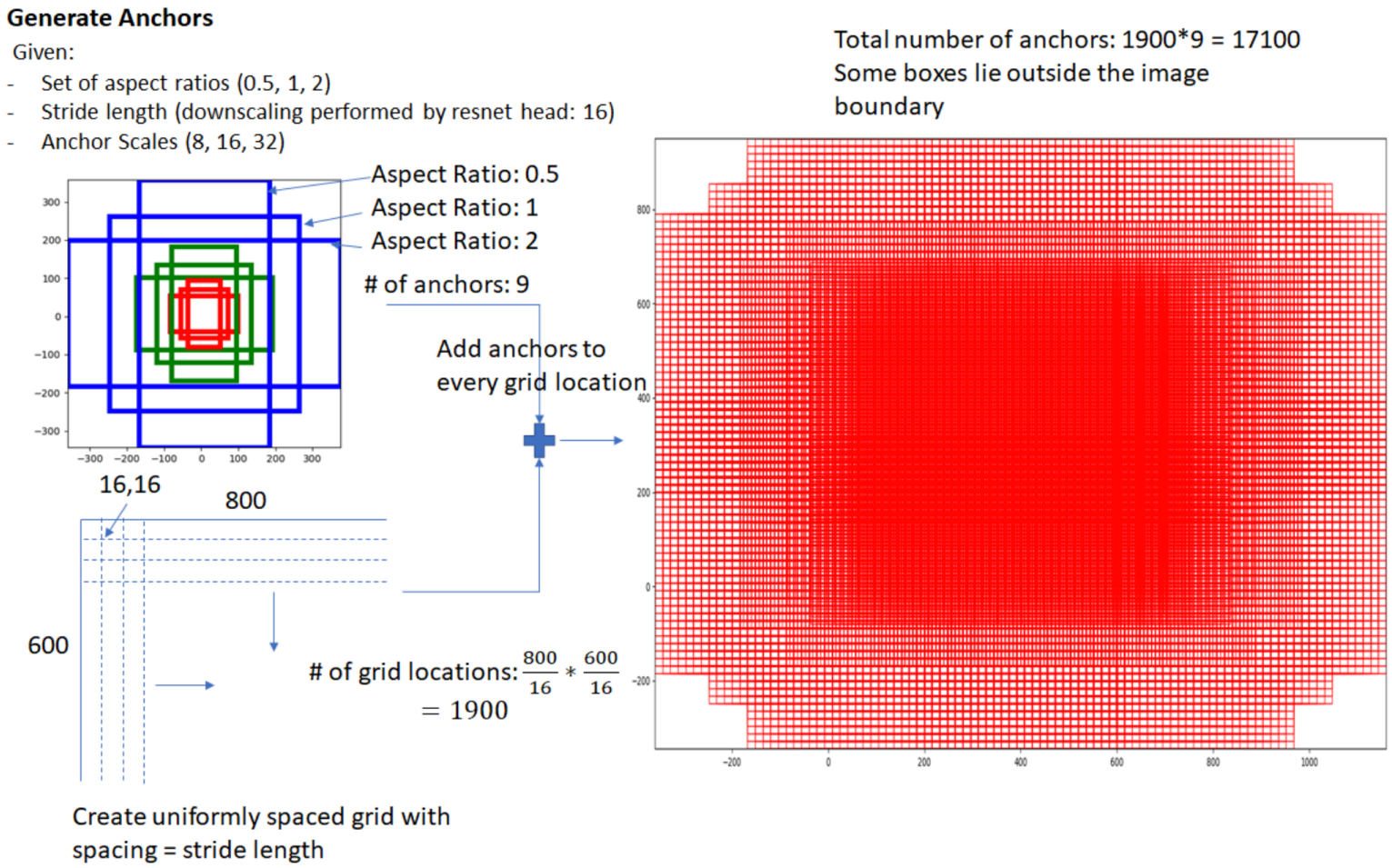



Figure from http://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/

# Region Proposal Network

Input is feature maps after convolution; output is a per anchor box classification {object in box, no object in box} and regression (x, y, w and h relative to the anchor box); separate weights are used for each of the k anchor classifiers (?) and regressors (yes)

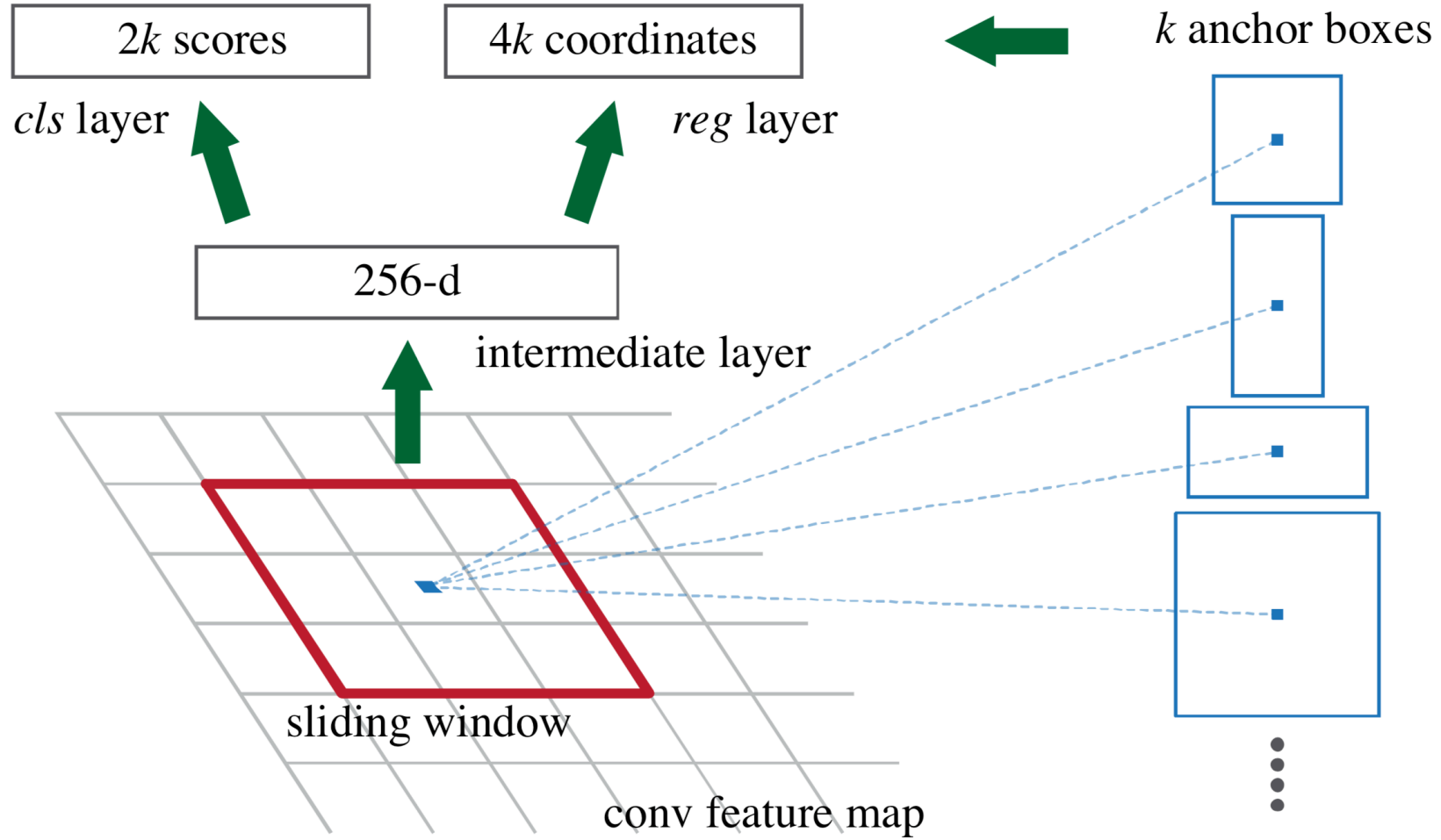



Figure from https://arxiv.org/abs/1506.01497

# Spatial Pyramid Pooling And RoI Pooling

SPP input is arbitrary size feature map regions and output is 1 fixed length vector per region appropriate for fully connected layers (FC can batch process multiple regions via matrix mult); RoI pooling is SPP with only 1 level and each output region a fixed size N x W x H tensor instead of a N*W*H x 1 vector; note that applying an integer number of bins to an arbitrary region size results in some bin boundary quantization affects on the resulting output

Output is fixed length vectors irrespective of region size that are appropriate for fully connected layer classification

Allows convolutional features to be generated for the full image all at once instead of for each region separately (a huge amount of overhead when there's overlapping regions)

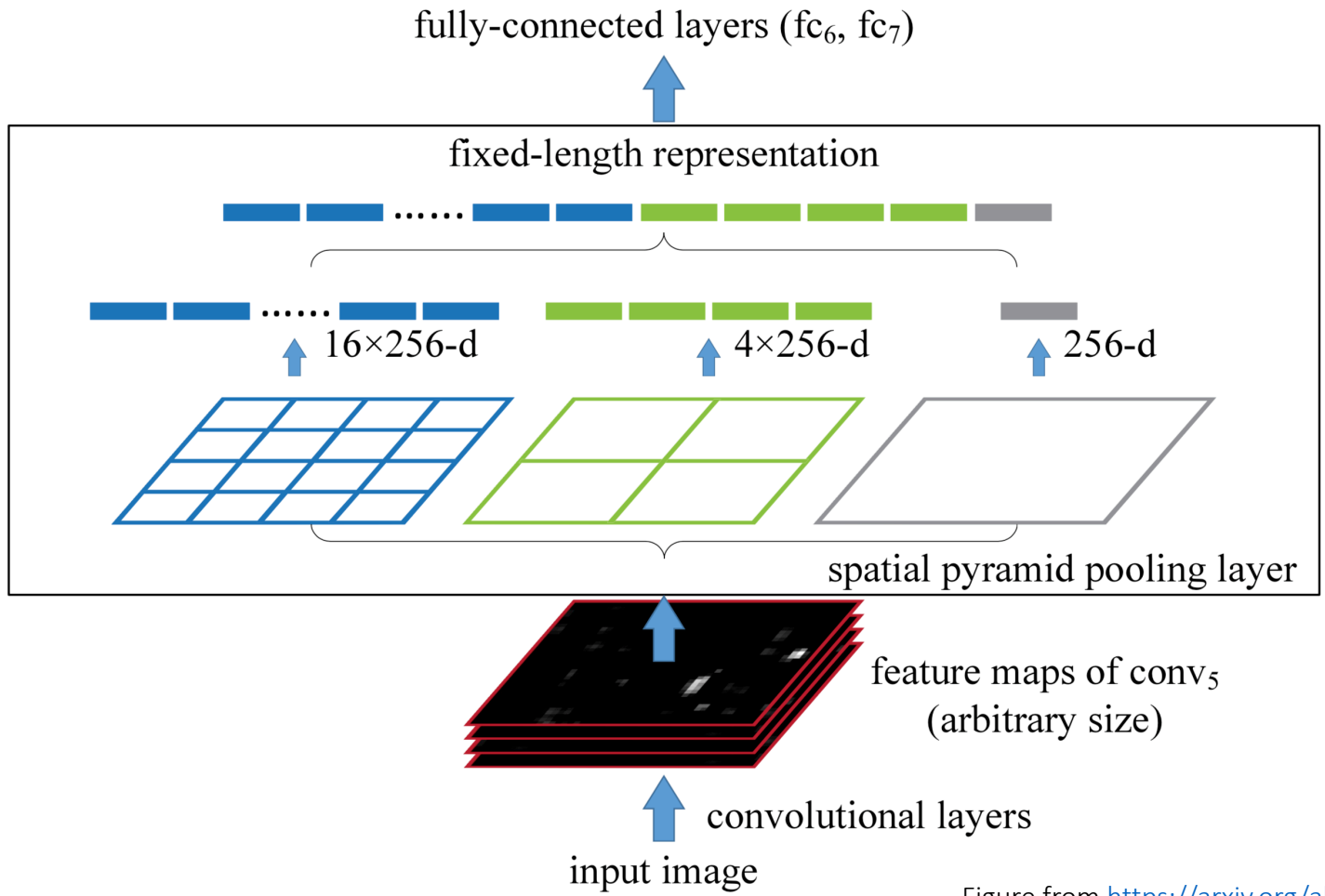
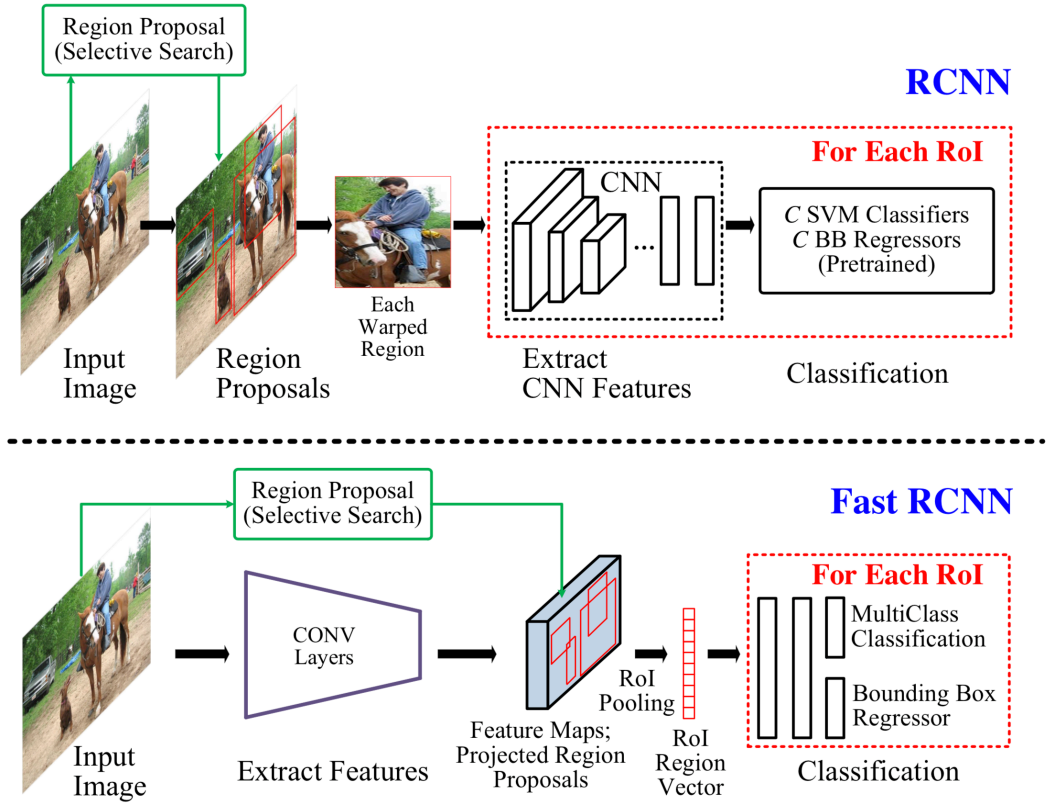



Figure from https://arxiv.org/abs/1406.4729

# Example: Two Stage Approach History

While R-FCN is included I prefer stopping at Faster R-CNN (effectively what was described)



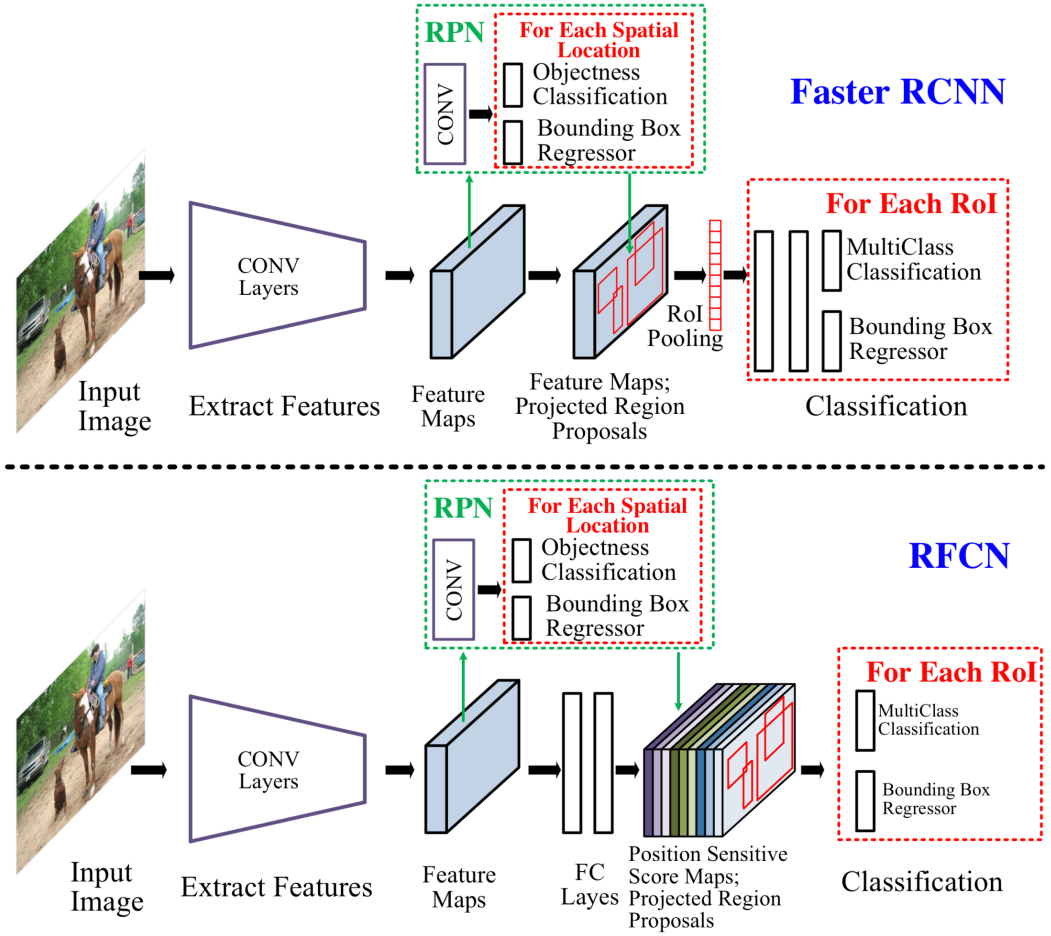




Figure from https://arxiv.org/abs/1809.02165

# Example: Faster R-CNN

After RoI pooling each region of interest, irrespective of size, is mapped to a constant size feature map or constant length vector; these are batched together for all regions of interest and become the input to the classification layers

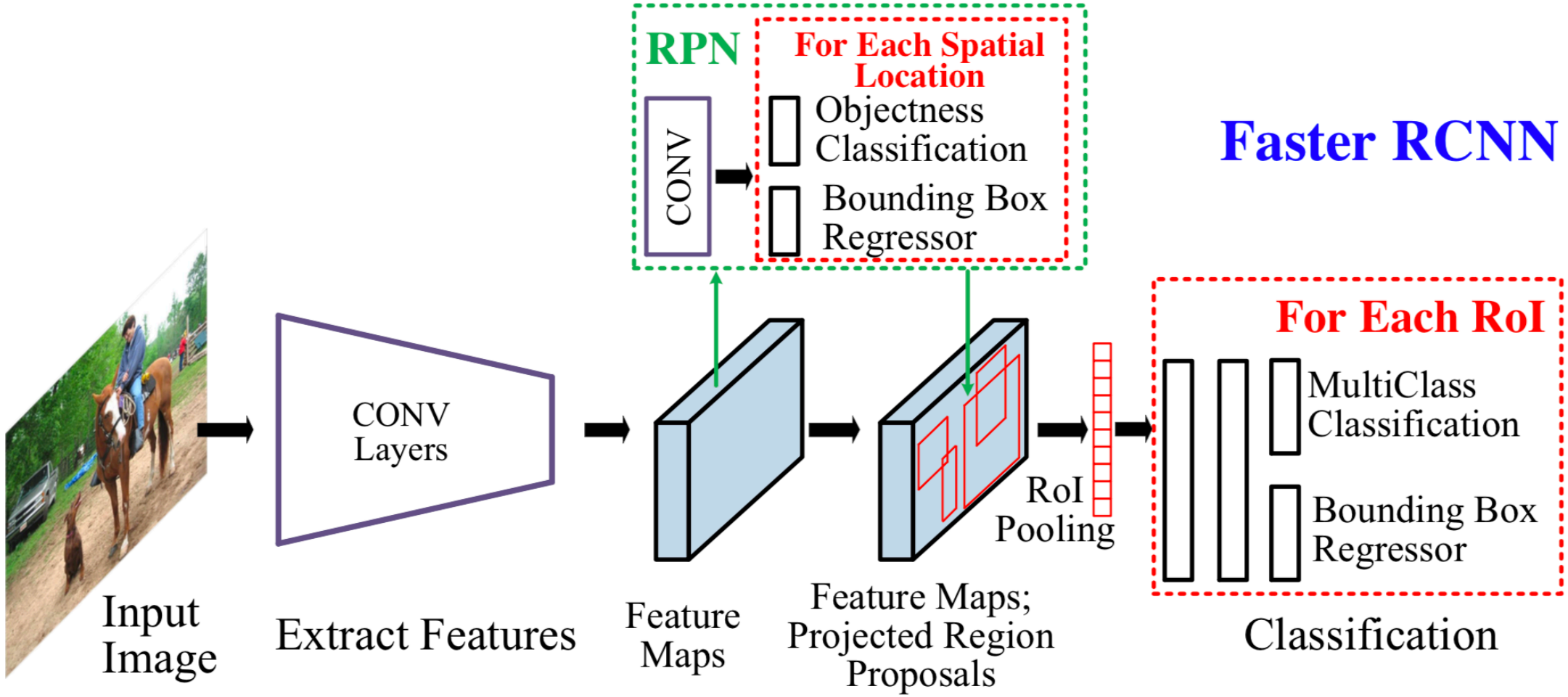
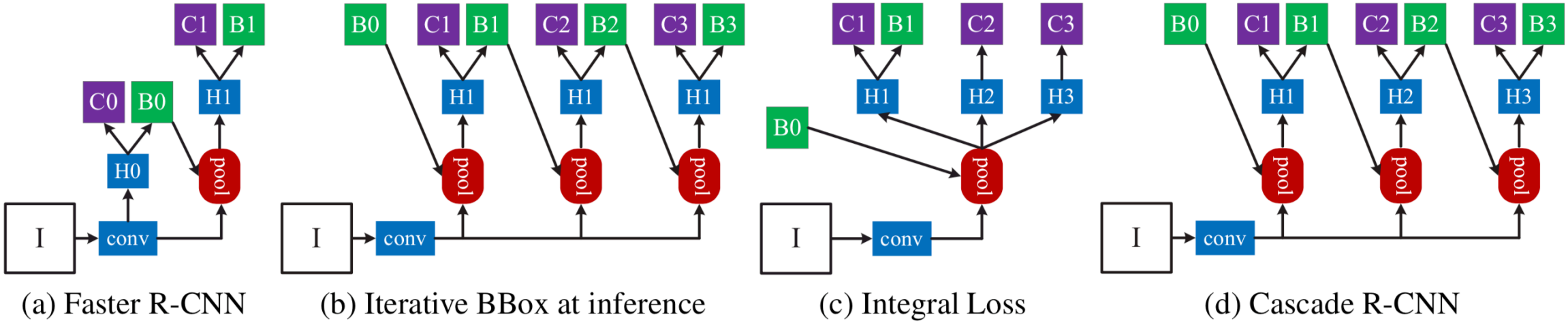



Figure from https://arxiv.org/abs/1809.02165

# Example: Cascade R-CNN

A method for improving accuracy where bounding box localization is trained sequentially with increasing IoU thresholds; I = image, conv = shared convolutional layers, pool = region pooling, H = network head, Bx = bounding box stage x proposal and Cx = classification stage x proposal



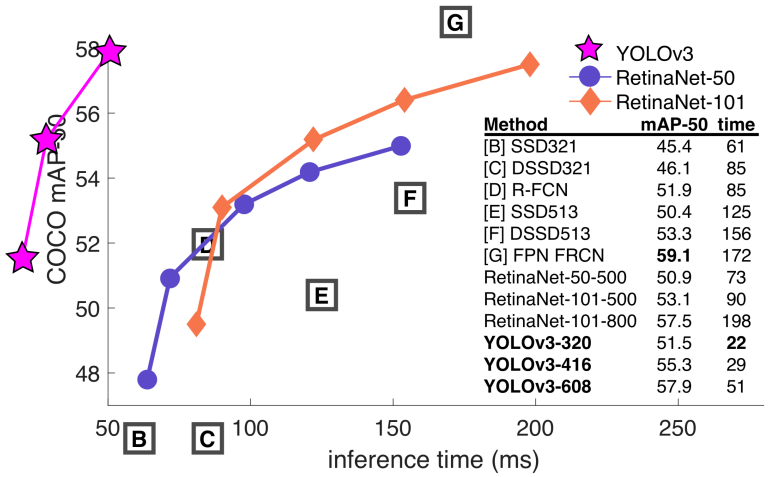

(a) Faster R-CNN (b) Iterative BBox at inference (c) Integral Loss (d) Cascade R-CNN

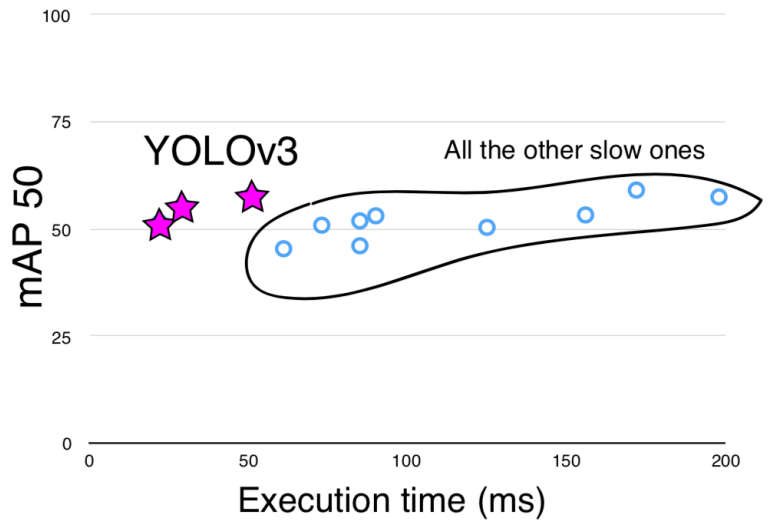
Figure from https://arxiv.org/pdf/1712.00726.pdf

# Single Stage Object Detection

- Popular methods that estimate bounding boxes and classes in a single stage (then assemble the results with post processing)
  - YOLO
    - YOLO
    - YOLO9000
    - YOLO V3
  - SSD
  - RetinaNet

If you're ever feeling stressed and could use a laugh to take your mind off things, I highly recommend reading the YOLO V3 paper




## 1. Introduction

Sometimes you just kinda phone it in for a year, you know? I didn't do a whole lot of research this year. Spent a lot of time on Twitter. Played around with GANs a little. I had a little momentum left over from last year [12] [1]; I managed to make some improvements to YOLO. But, honestly, nothing like super interesting, just a bunch of small changes that make it better. I also helped out with other people's research a little.

Actually, that's what brings us here today. We have a camera-ready deadline [4] and we need to cite some of the random updates I made to YOLO but we don't have a source. So get ready for a TECH REPORT!

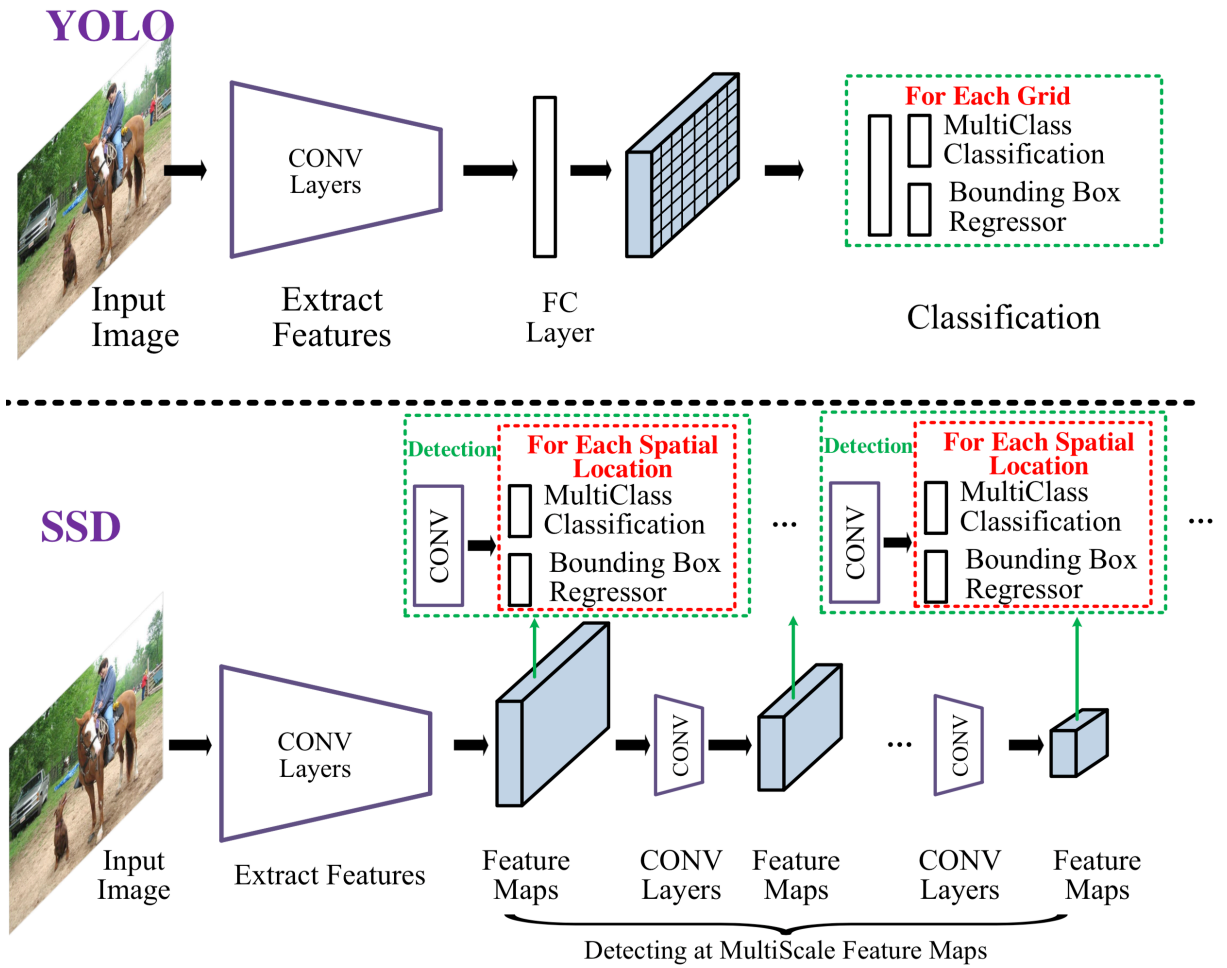
The great thing about tech reports is that they don't need intros, y'all know why we're here. So the end of this introduction will signpost for the rest of the paper. First we'll tell you what the deal is with YOLOv3. Then we'll tell you how we do. We'll also tell you about some things we tried that didn't work. Finally we'll contemplate what this all means.

## 2. The Deal

So here's the deal with YOLOv3: We mostly took good ideas from other people. We also trained a new classifier network that's better than the other ones. We'll just take you through the whole system from scratch so you can understand it all.

Text from https://arxiv.org/abs/1804.02767

# Examples: YOLO And SSD




Detection heads are similar to RPNs except that multi class classification is predicted instead of a RPN predicting class / no class

Figure from https://arxiv.org/abs/1809.02165

# Example: RetinaNet

K = number of classes and A = number of anchor boxes per feature map pixel

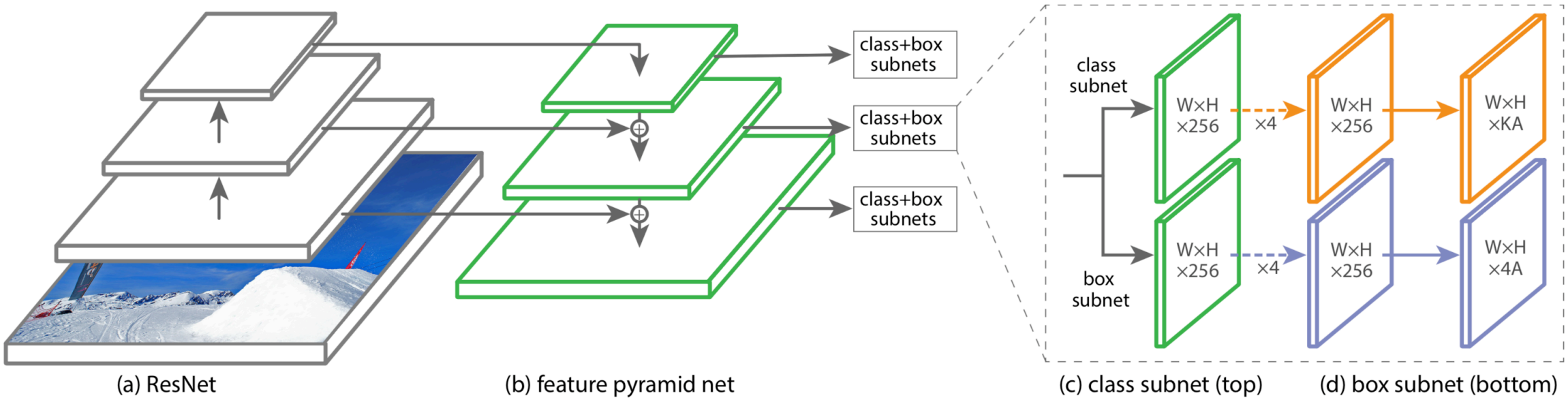



Figure from https://arxiv.org/abs/1708.02002

# Revisiting Anchor Boxes

- Though train
  - Both the 1 and 2 stage methods started from a pre determined set of potential anchor boxes
  - Q: Is it possible to determine a set of potential anchor boxes from the image itself? A: Yes
  - Ideas for doing this include predicting anchor boxes (a) centers and (b) upper left and lower right corners from the image itself
  - Below is a non comprehensive laundry list

- DeNet: scalable real-time object detection with directed sparse sampling
  - https://arxiv.org/abs/1703.10295
- Point linking network for object detection
  - https://arxiv.org/abs/1706.03646
- CornerNet: detecting objects as paired keypoints
  - https://arxiv.org/abs/1808.01244
- CornerNet-Lite: efficient keypoint based object detection
  - https://arxiv.org/abs/1904.08900
- Objects as points
  - https://arxiv.org/abs/1904.07850
- CenterNet: keypoint triplets for object detection
  - https://arxiv.org/abs/1904.08189

# Some Additional References

- Single-shot refinement neural network for object detection
  - https://arxiv.org/abs/1711.06897
- Light-head R-CNN: in defense of two-stage object detector
  - https://arxiv.org/abs/1711.07264
- Pelee: a real-time object detection system on mobile devices
  - https://arxiv.org/abs/1804.06882
- M2Det: a single-shot object detector based on multi-level feature pyramid network
  - https://arxiv.org/abs/1811.04533
- AutoFocus: efficient multi-scale inference
  - https://arxiv.org/abs/1812.01600

# Some Additional References

- Fully convolutional one-stage object detection
  - https://arxiv.org/abs/1904.01355
- NAS-FPN: learning scalable feature pyramid architecture for object detection
  - https://arxiv.org/abs/1904.07392
- Learning data augmentation strategies for object detection
  - https://arxiv.org/abs/1906.11172
- Cascade RPN: delving into high-quality region proposal network with adaptive convolution
  - https://arxiv.org/abs/1909.06720
- EfficientDet: scalable and efficient object detection
  - https://arxiv.org/abs/1911.09070

# Post Processing

- Non maximal suppression
  - Inputs contain scores and box locations
  - Create a score list by sorting the entries based on the score
  - Repeat the following until no more entries with scores above a threshold are in the score list
    - Start with the entry with the best score on the score list
    - Remove other entries (suppress non maximals) from the score list with significant overlap
    - Add the best entry from the score list to the prediction list
    - Remove the best entry from the score list

- Variants
  - Can play games like averaging a few together, instead of removing others penalize them by reducing their score, ...
  - A difficulty is finding balance between suppressing windows and detecting close together objects (overlap parameter choice)

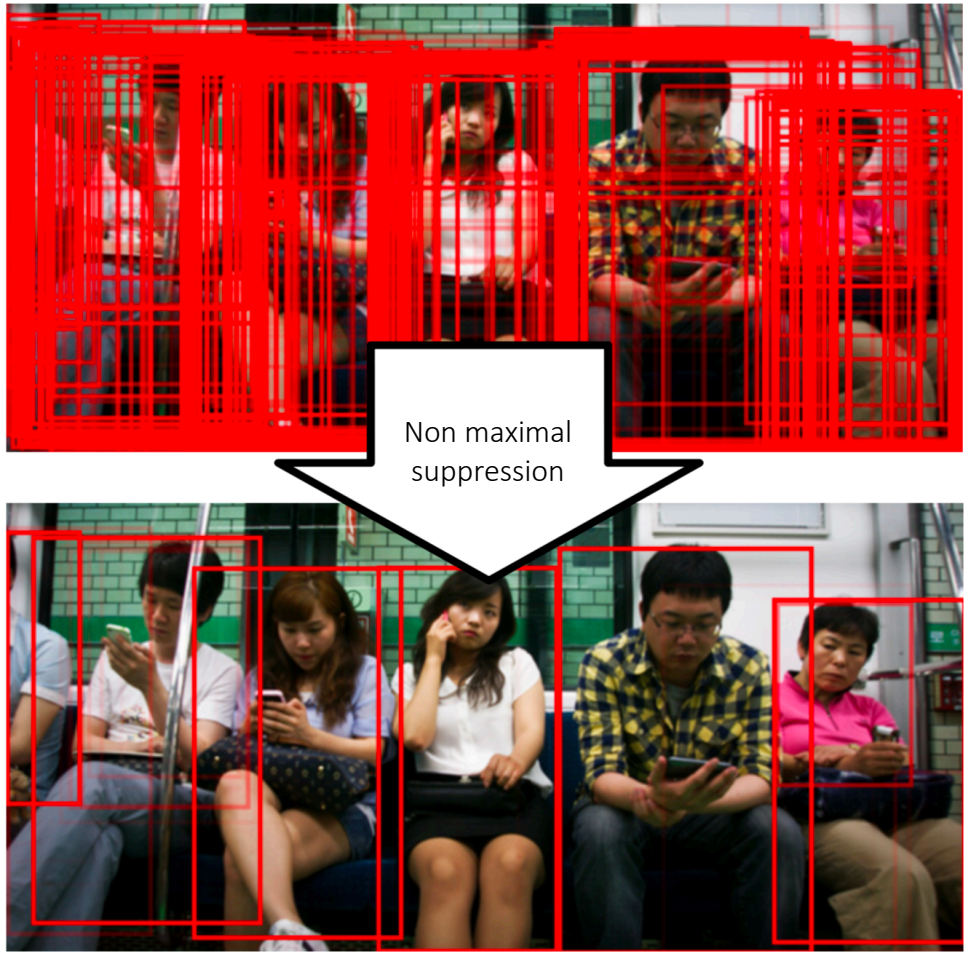



Figure from https://arxiv.org/pdf/1705.02950.pdf

# Post Processing

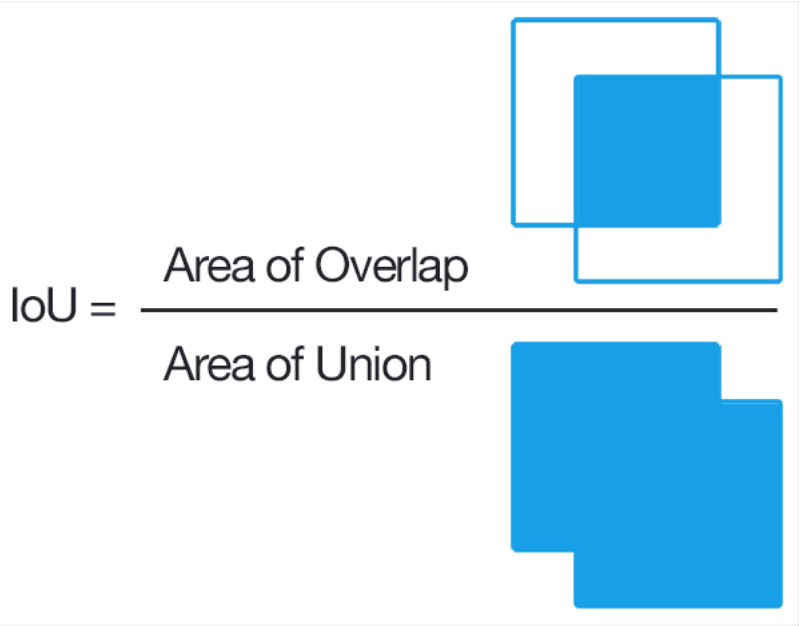

Figure from http://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/

# Evaluation – Confidence Threshold

- The network output can be summarized as
  - Class 0
    - Confidence 0, bounding box 0
    - Confidence 1, bounding box 1
    - …
  - Class 1
    - Confidence 0, bounding box 0
    - Confidence 1, bounding box 1
    - …
  - …
- Non maximal suppression (or another algorithm) is run on the network output to merge / remove / etc. predictions
- Remaining predictions are further reduced by only keeping predictions for a class above a confidence threshold for the class

# Evaluation – Intersection Over Union

- For each of the remaining predictions the intersection over union with closest ground truth object is computed which allows the prediction to be marked as {0 == not true positive, 1 == true positive} if it is above a given IoU threshold
  - Class 0
    - Confidence 0, bounding box 0, TP for IoU > 0.50?, TP for IoU > 0.55?, …, TP for IoU > 0.95?
    - Confidence 1, bounding box 1, TP for IoU > 0.50?, TP for IoU > 0.55?, …, TP for IoU > 0.95?
    - …
  - Class 1
    - Confidence 0, bounding box 0, TP for IoU > 0.50?, TP for IoU > 0.55?, …, TP for IoU > 0.95?
    - Confidence 1, bounding box 1, TP for IoU > 0.50?, TP for IoU > 0.55?, …, TP for IoU > 0.95?
    - …
  - …
- Typically only 1 prediction is allowed to be a true positive for a given ground truth object and the other predictions are marked as not true positives




Historical contests used a single IoU value of 0.50; more recent contests have used a spectrum of IoU values from 0.50, 0.55, …, 0.95 to encourage better localization

Figure from https://commons.wikimedia.org/wiki/File:Intersection_over_Union_-_visual_equation.png

# Evaluation – Precision

- Precision
  - How precise are the predictions for a particular class
  - True class positives / (true class positives + false class positives)
  - Correctly predicted objects in a class / All objects predicted in a class
  - Penalizes you for making too many predictions that aren't correct of a class
- Average precision (AP)
  - Precision averaged over multiple IoU thresholds
- Mean average precision (mAP)
  - Average precision averaged over all classes

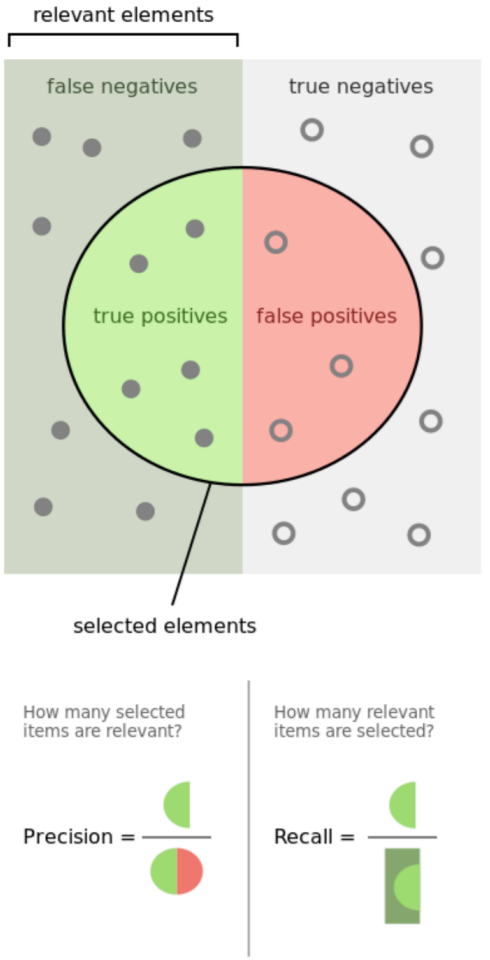



Figure from https://en.wikipedia.org/wiki/Precision_and_recall

# Evaluation – Recall

- Recall
  - How complete are the predictions for a particular class
  - True class positives / (true class positives + false class negatives)
  - Correctly predicted objects in a class / All objects that should have been predicted of a class
  - Penalizes you for missing objects of a class
- Average recall (AR)
  - Recall averaged over multiple IoU thresholds
- Mean average recall (mAR)
  - Average recall averaged over all classes

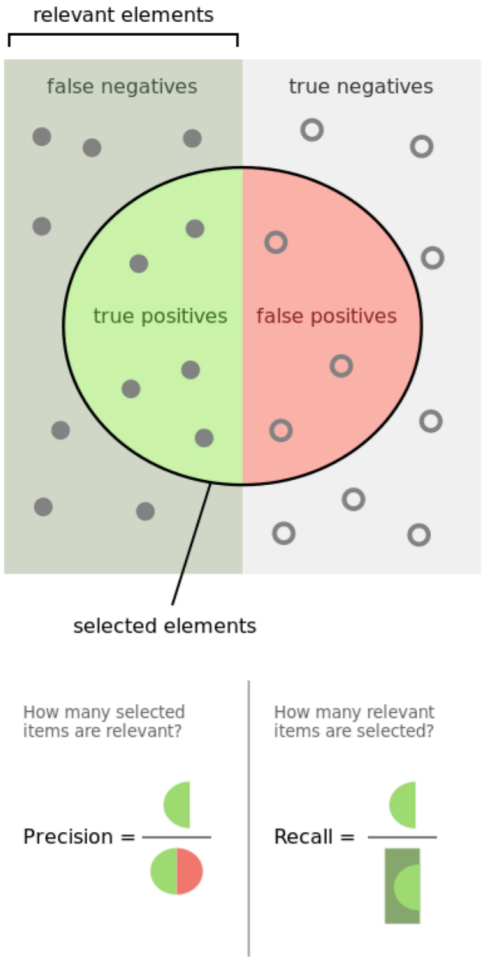



Figure from https://en.wikipedia.org/wiki/Precision_and_recall

# Evaluation – Precision Recall Curve

- So remember a few slides back when we got rid of predictions below a specified confidence threshold?

- You can create a precision recall curve by varying this confidence threshold and plotting the resulting precision vs recall
  - Increasing the confidence threshold (fewer predictions) tends to lead to smaller values of recall and higher values of precision
  - Decreasing the confidence threshold (more predictions) tends to lead to larger values of recall and smaller values of precision

- You can plot a precision recall curve for a single class at a single IoU; or you can plot multiple precision recall curves for a single class each at a different IoU

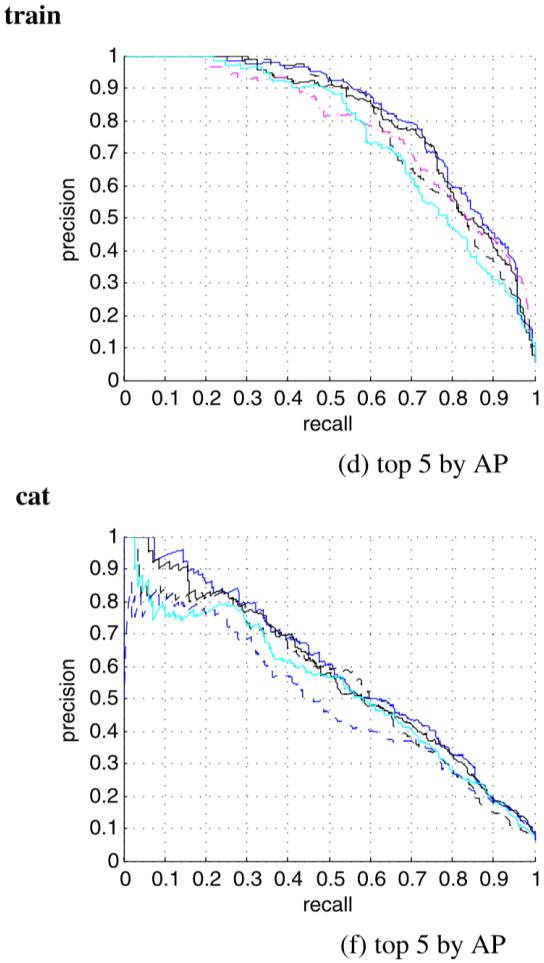



Figure from http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf

# Extending From Images To Video

- Frequently each frame is treated as a completely separate image for the CNN to perform MoD on
  - Feels suboptimal
- Research is directed at exploiting the temporal structure to improve both accuracy and performance
  - Just mentioned here
  - But details won't be discussed in these slides
- For reference, here's a paper that proposes a method for achieving the same accuracy as CNN style 3D convolution with ~ CNN style 2D convolution performance
  - TSM: temporal shift module for efficient video understanding (https://arxiv.org/abs/1811.08383)

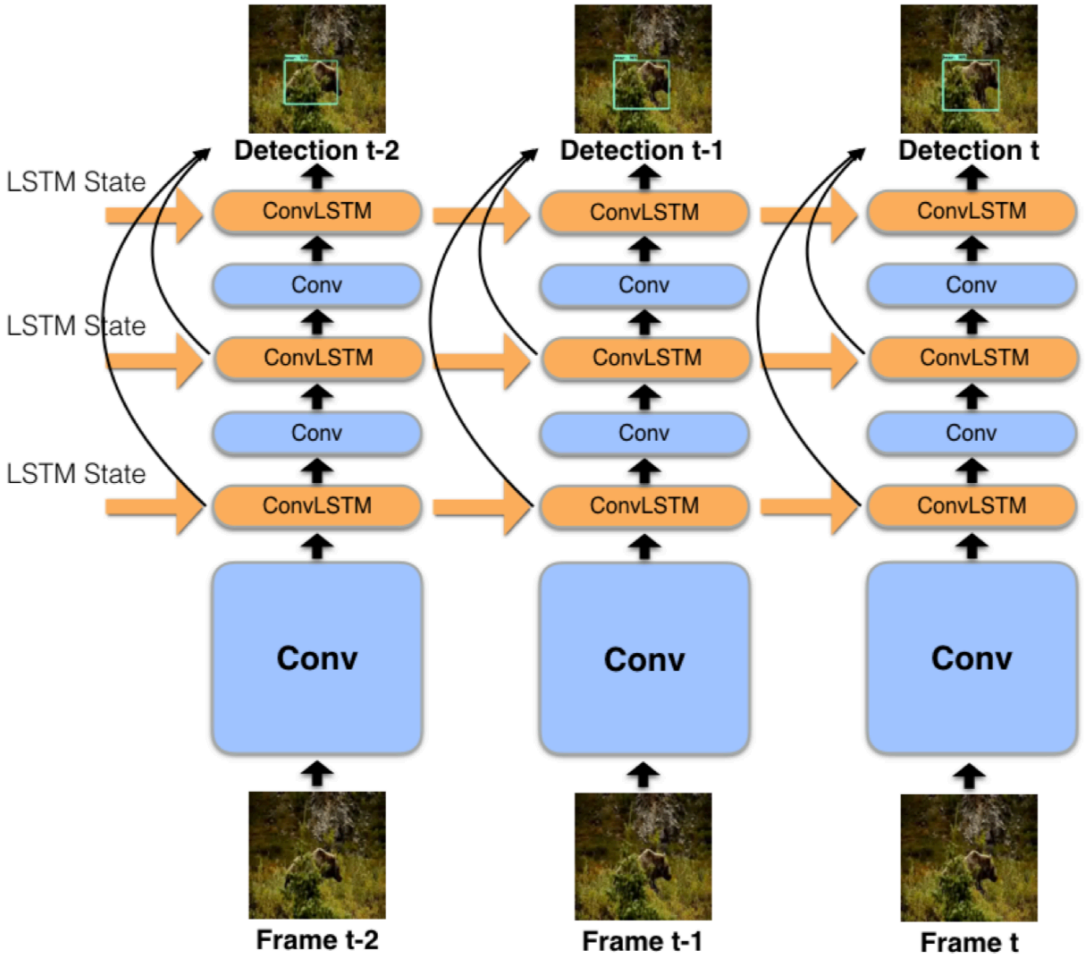



Figure from https://arxiv.org/abs/1711.06368

# Object Segmentation

# Goal

- Trace a contour around the boundary of every object in the image and classify the object in the contour


Figure from https://www.mapillary.com/dataset/vistas

# Data

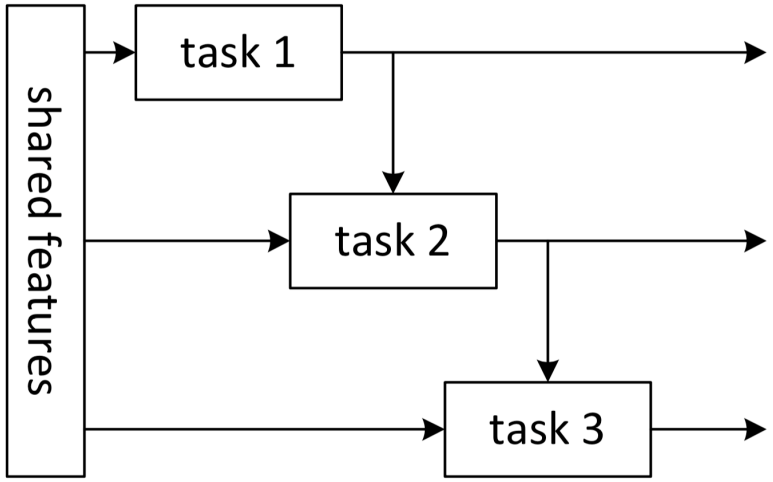
- Laundry list
  - COCO
  - Mapillary Vistas

- The high complexity of labeling makes object segmentation a good candidate for synthetic data sets
  - Generating image segmentation datasets with Unreal Engine 4
  - https://medium.com/@jeff_97181/generating-image-segmentation-datasets-with-unreal-engine-4-2b5b9f75da34

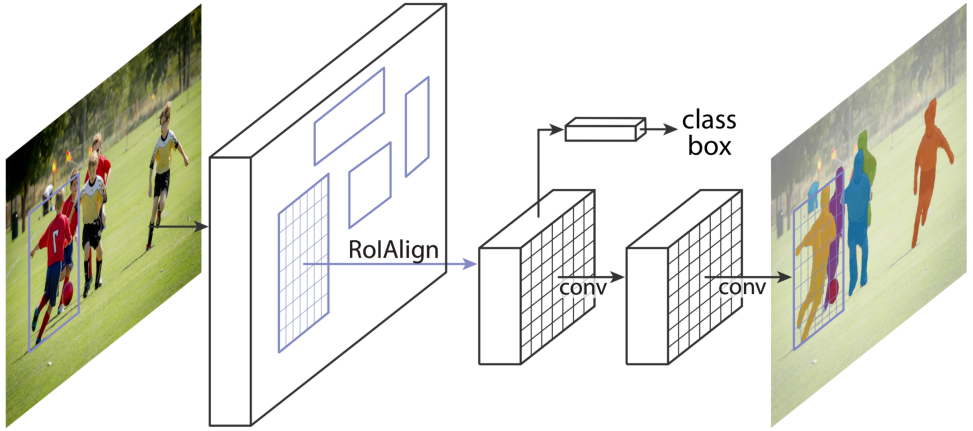

Figure from https://www.mapillary.com/dataset/vistas

# Multistage Methods

- Three stage (+ post processing)
  - Classify anchor boxes {contains object, does not contain object} and refine {Δx, Δy, Δw, Δh} anchor box locations
  - Then divide the refined anchor box into a grid and classify grid cells {part of the object, not part of the object} to determine a mask
  - Then classify objects in the grid cells that are part of the object for refined anchor boxes that contain objects (and possibly refine their location again)




- Two stage (+ post processing)
  - Classify anchor boxes {contains object, does not contain object} and refine {Δx, Δy, Δw, Δh} anchor box locations
  - Then predict a mask and classify objects in the refined anchor boxes that contain objects (and possibly refine their location again)




Figures from https://arxiv.org/abs/1512.04412 and https://arxiv.org/abs/1703.06870

# Ex: Instance Aware Semantic Segmentation

3 stage method; RoI warping is a differentiable transformation allowing end to end training that crops and warps a RoI to a target size via interpolation followed by pooling

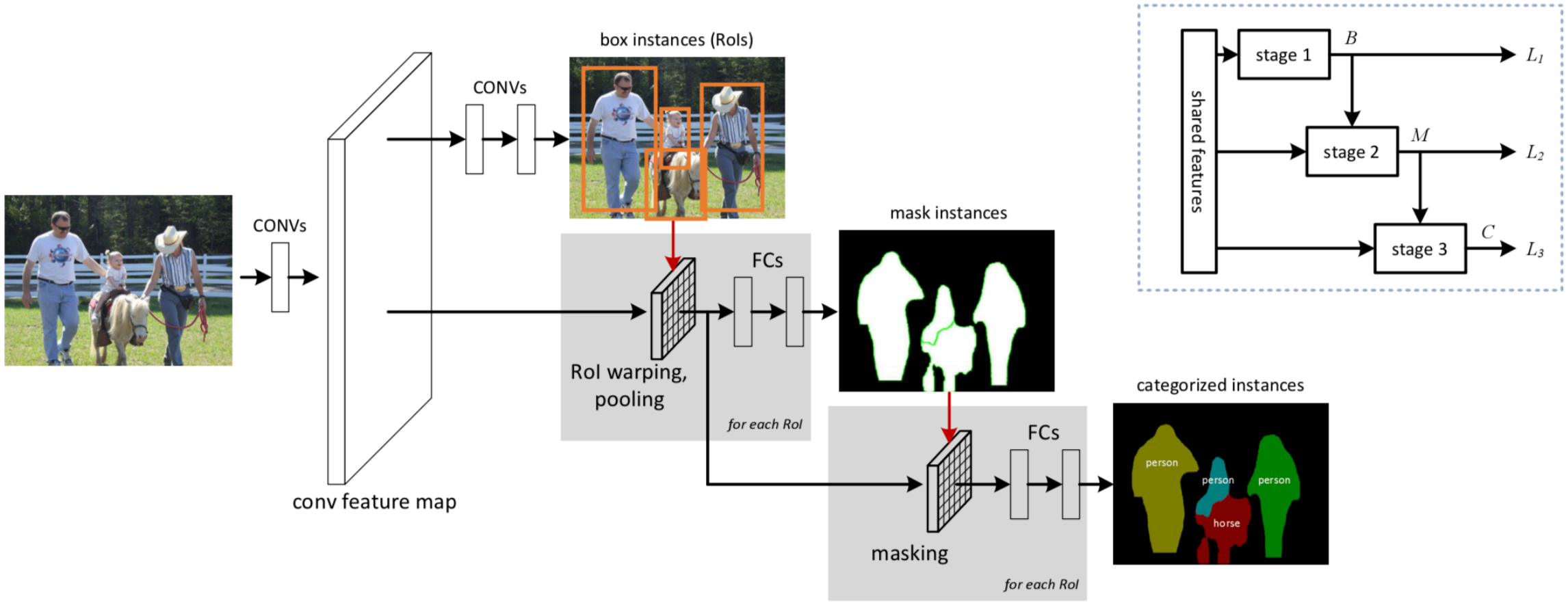



Figure from https://arxiv.org/abs/1512.04412

# RoI Align

Input is N x h x w feature maps from convolutional layer and box coordinates for an arbitrary sized RoI, output is fixed size N x H x W feature maps with bilinear interpolation used to avoid bin boundary quantization (a key for improving performance relative to RoI pooling)

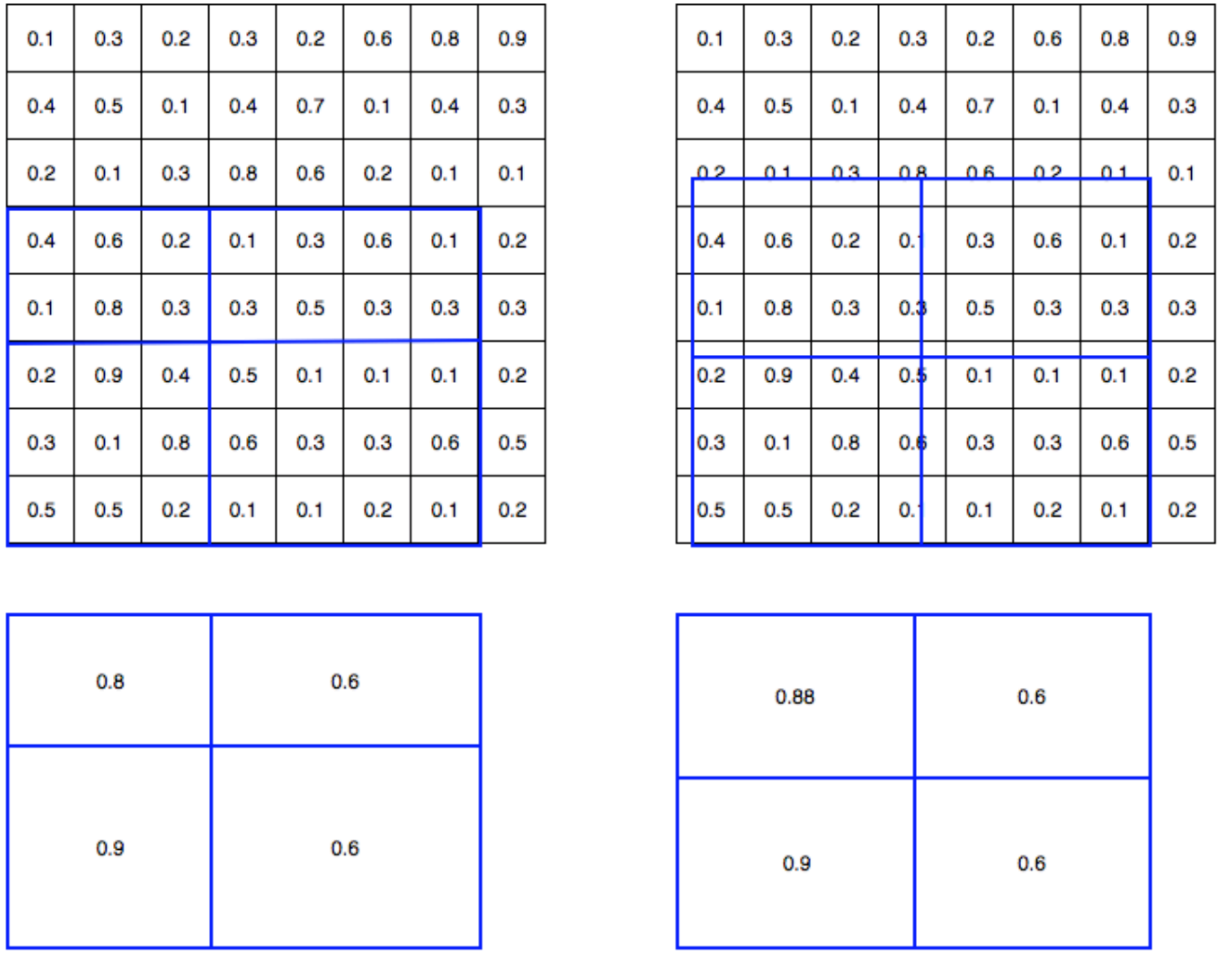



Figure from https://medium.com/@jonathan_hui/image-segmentation-with-mask-r-cnn-ebe6d793272

# Example: Mask R-CNN

2 stage method

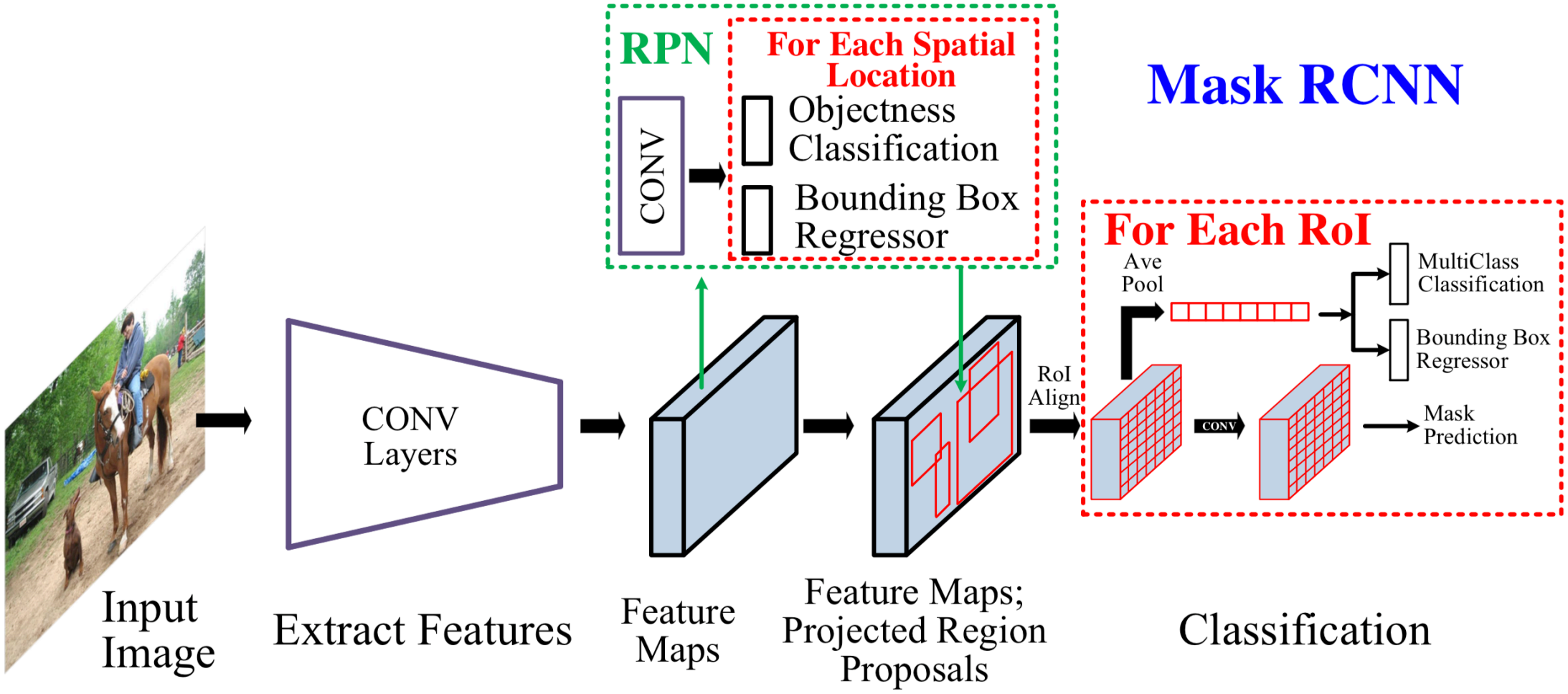



Figure from https://arxiv.org/abs/1809.02165

# Some Additional References

- Fully convolutional instance-aware semantic segmentation
  - https://arxiv.org/abs/1611.07709
- Path aggregation network for instance segmentation
  - https://arxiv.org/abs/1803.01534
- RetinaMask: learning to predict masks improves state-of-the-art single-shot detection for free
  - https://arxiv.org/abs/1901.03353
- Mask scoring R-CNN
  - https://arxiv.org/abs/1903.00241
- CenterMask: real-time anchor-free instance segmentation
  - https://arxiv.org/abs/1911.06667
- YOLACT: real-time instance segmentation
  - https://arxiv.org/abs/1904.02689
  - https://github.com/dbolya/yolact
- YOLACT++: better real-time instance segmentation
  - https://arxiv.org/abs/1912.06218

# Depth Estimation

# Goal

- Given a stereo image pair separated in space determine the distance of every pixel from the cameras

Figure from https://arxiv.org/pdf/1512.02134.pdf

# Goal

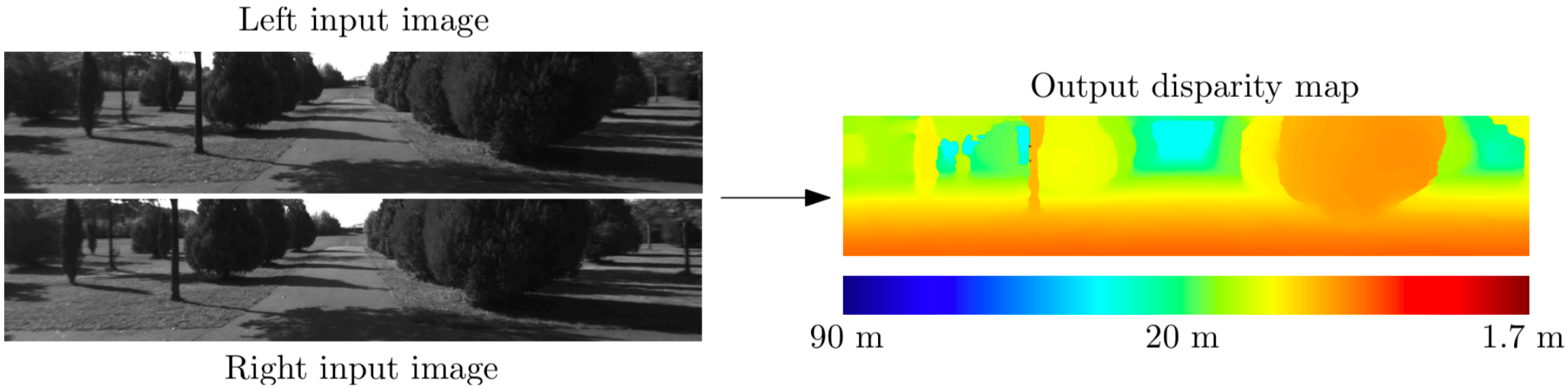



Figure from https://arxiv.org/abs/1409.4326

# Fundamentals

- Pre processing
  - Start from a left and right image
  - Transform to emulate having a common image plane with focal length f and distance between cameras T
- Disparity
  - Search along horizontal (epipolar) lines to find the same point in both images
  - The disparity is difference in location of the same point in both images d = $x_R$ - $x_L$
- Depth
  - The depth can be computed from the disparity as Z = f T / d
  - The smaller the disparity the farther the point is away (and vice versa)

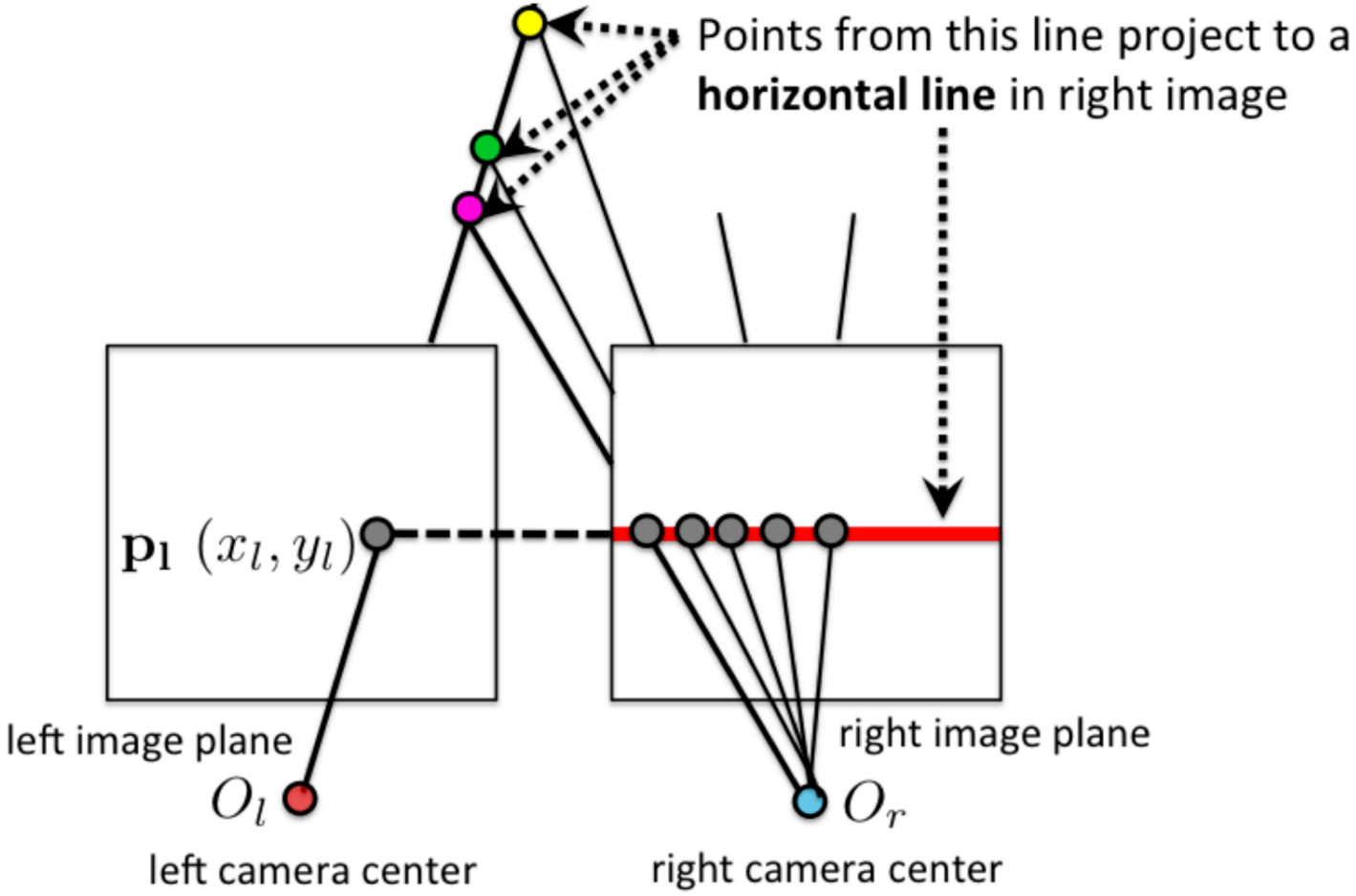
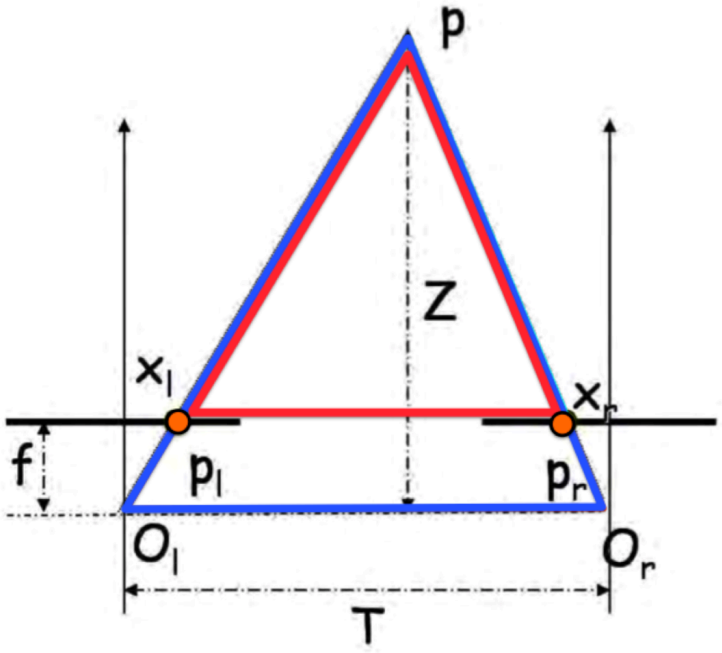



Figure from http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

# Fundamentals

- Pre processing
  - Start from a left and right image
  - Transform to emulate having a common image plane with focal length f and distance between cameras T
- Disparity
  - Search along horizontal (epipolar) lines to find the same point in both images
  - The disparity is difference in location of the same point in both images d = $x_R$ - $x_L$
- Depth
  - The depth can be computed from the disparity as Z = f T / d
  - The smaller the disparity the farther the point is away (and vice versa)




**Similar triangles:**

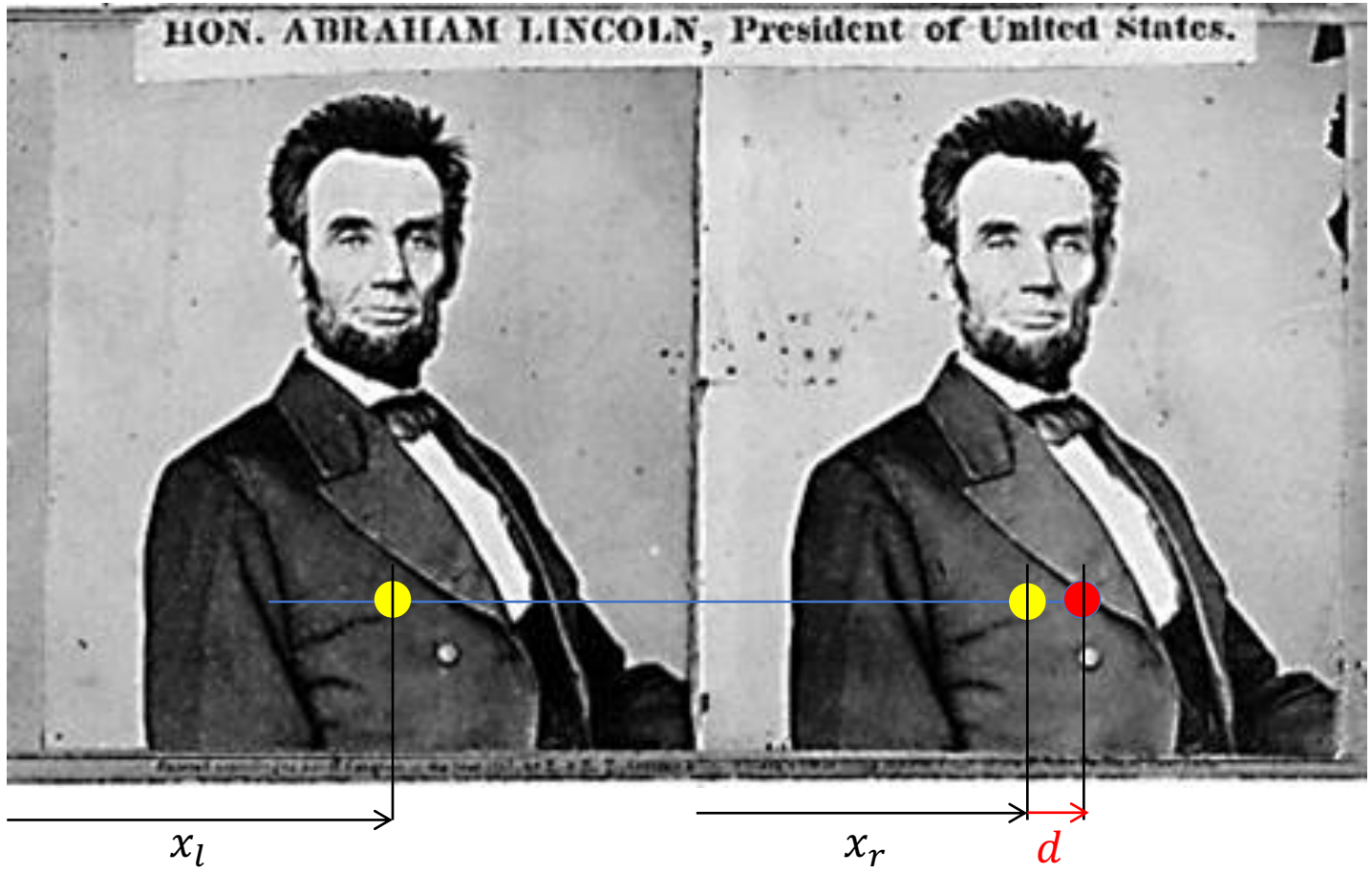
$$\frac{T}{Z} = \frac{T + x_l - x_r}{Z - f}$$

$$Z = \frac{f \cdot T}{x_r - x_l}$$

So if I know $x_l$ and $x_r$, then I can compute Z!

Figure from http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

# Fundamentals



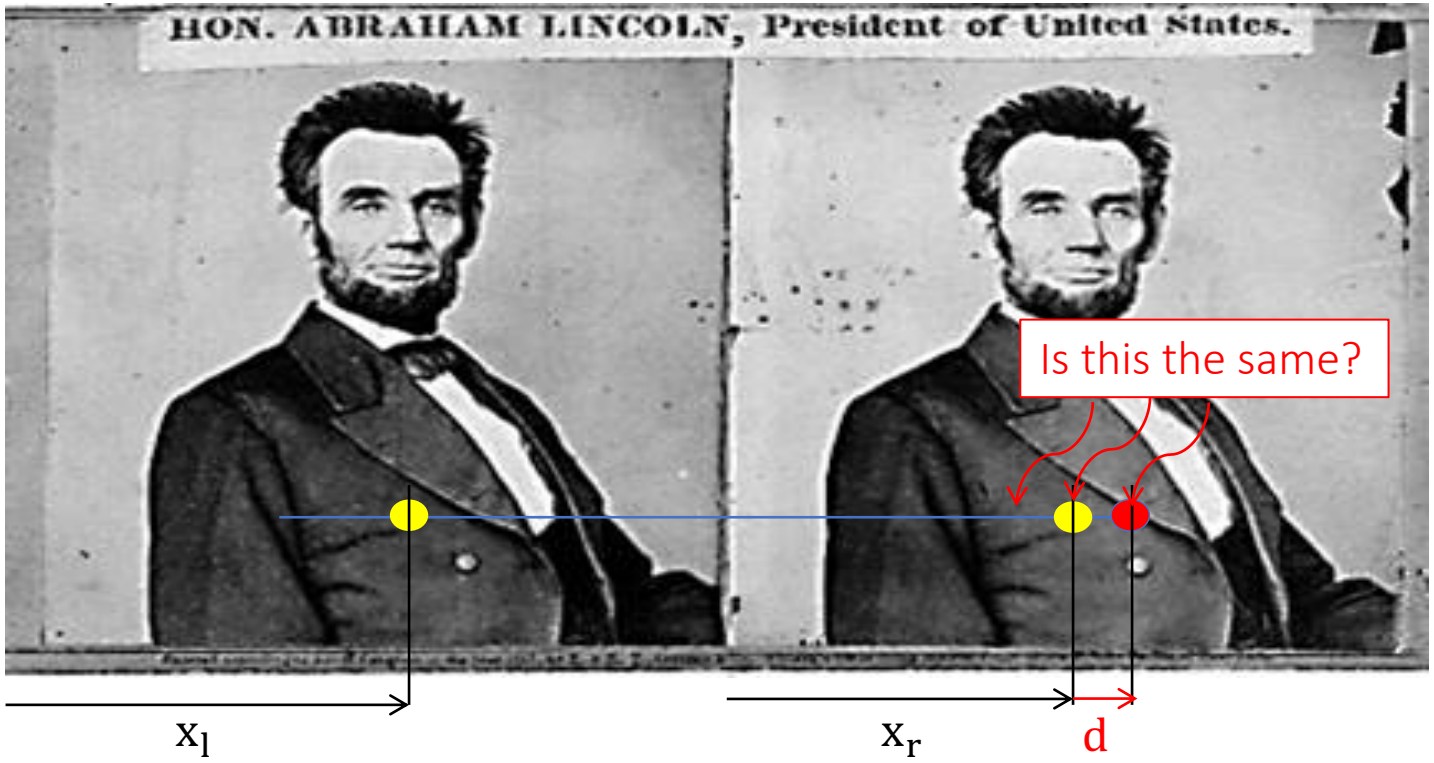

Apologies to the creator of this figure (I'm not sure where it originally came from)

# Classical Stereo Disparity Estimation

- Cost cube creation
  - Cube = {disparity 0, ..., D – 1} x rows x cols
  - Values represent left / right match cost at the specified disparity
  - Patch comparisons via L1, L2 and variants are common matching
- Cost cube smoothing
  - Smooth consistent with image expectations
  - Across pixels via filtering like cross based cost averaging
  - Across pixels and disparities via nonlinear methods like SGM
- Cost cube to disparity
  - Winner take all selection of best match which is lowest cost
- Disparity refinement
  - Left / right consistency checks
  - Median filtering, sub pixel enhancement

# What Role Can CNNs Play?

- Challenges of stereo depth estimation
  - Poor texture in local regions
  - Repeated structure
- CNNs can help with finding the same point in 2 or more images
  - This is a classification problem
  - CNNs are pretty good at a lot of image related classification problems
- CNNs can do direct regression or quantized classification from pixels to disparity or distance
- CNNs can do iterative refinement of existing estimates




Beyond these classical replacements, object segmentation via CNNs can be used as additional information to help improve stereo matching

Apologies to the creator of this figure (I'm not sure where it originally came from)

# Data

- Laundry list
  - KITTI
  - FlyingThings3D
  - SceneFlow
  - ETH3D
- Note
  - Difficulty of acquisition and often sparse
  - 1 of the best uses of computer graphics for generating training data

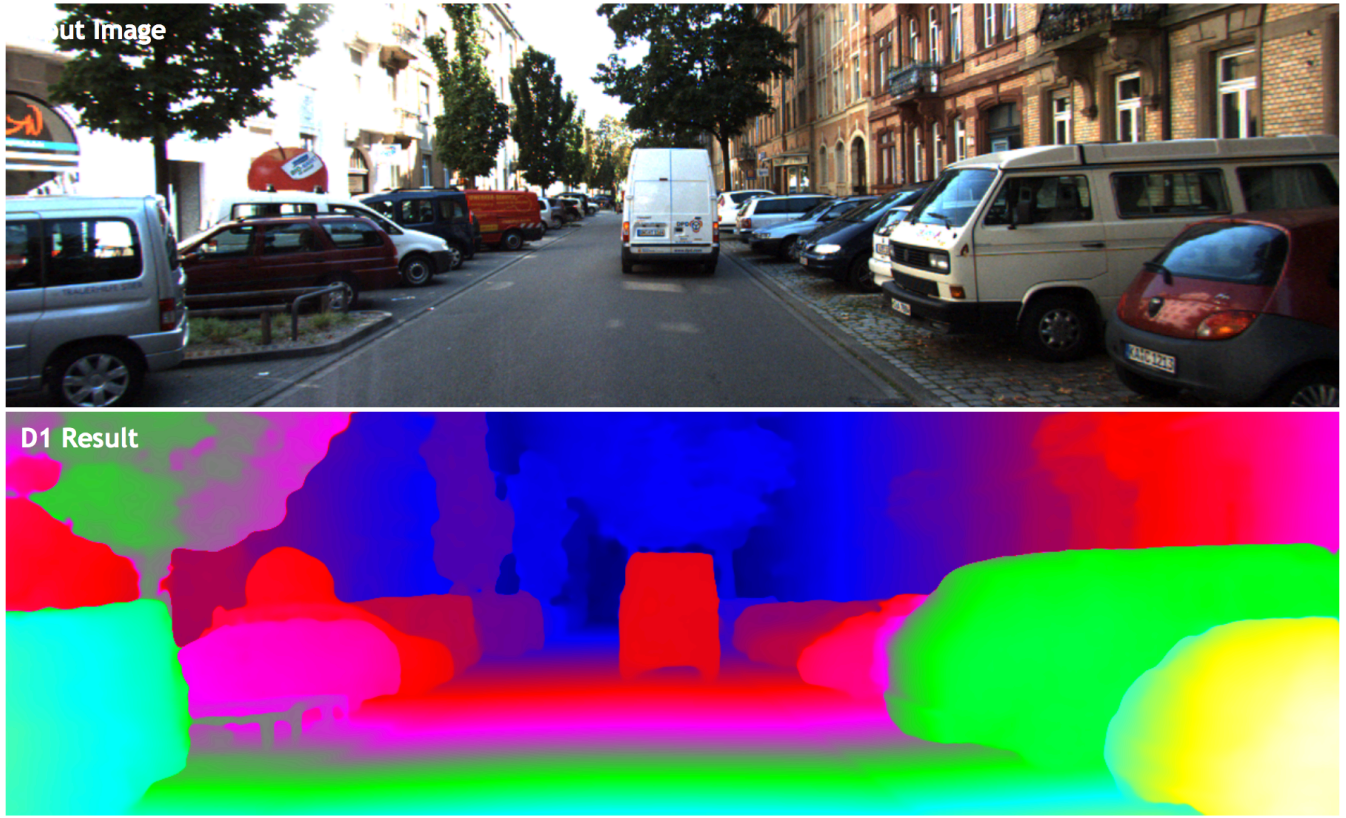


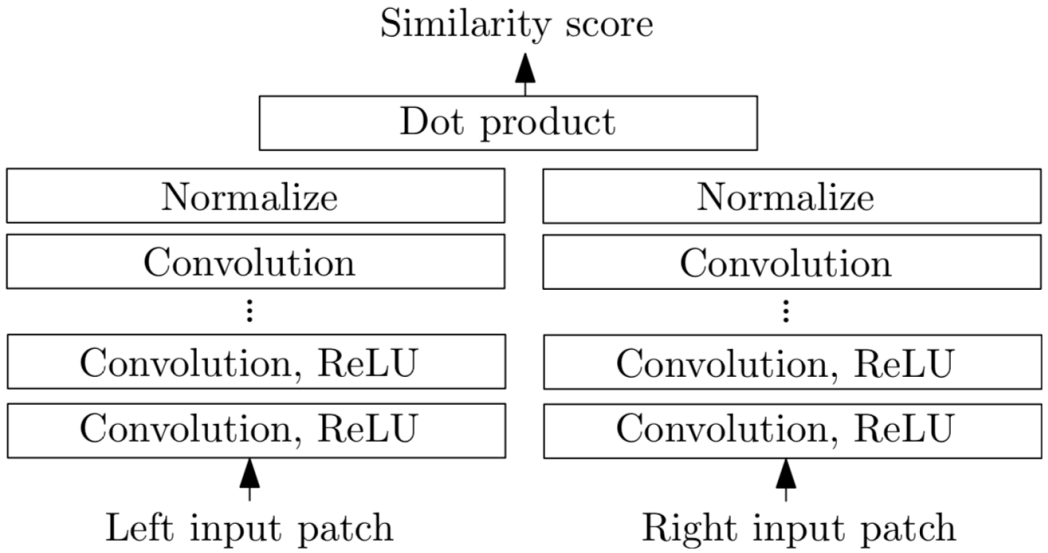

Figure from http://www.cvlibs.net/datasets/kitti/eval_scene_flow.php?benchmark=stereo

# Network Architecture Options

- Structures
  - Siamese network (typically with shared weights) then merge
  - Stacked feature maps in a single network (not as popular)

- Note
  - It is possible to estimate depth (to a scale factor) from a single image
  - However, performance tends to be much worse than stereo methods
  - As expected

# Image Patch Comparison



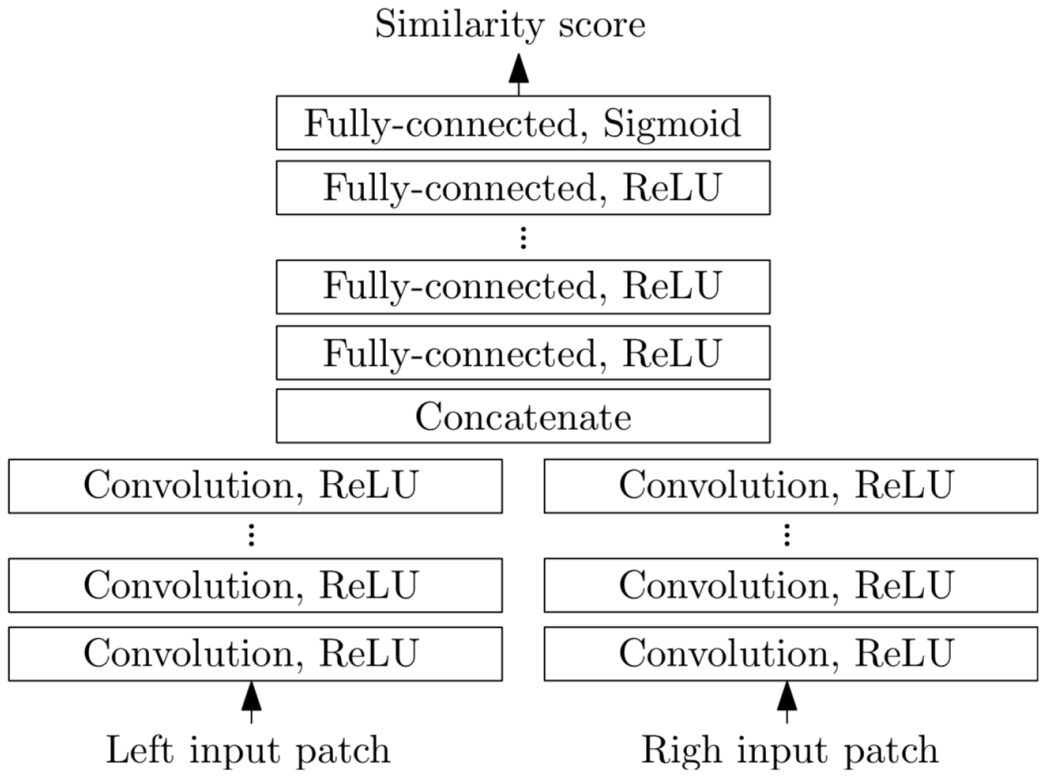
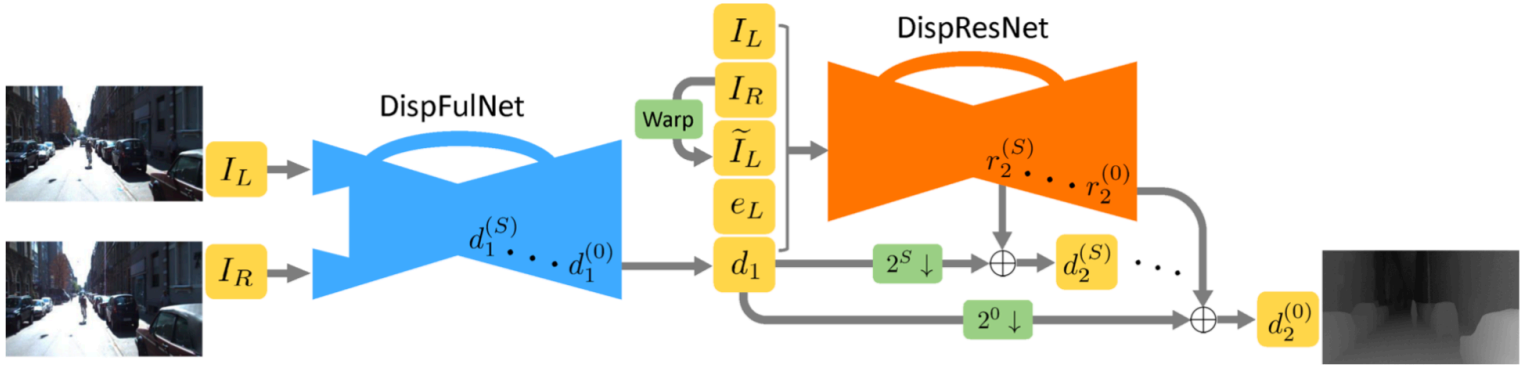




Figure from https://arxiv.org/abs/1510.05970

# Example: Cascade Residual Learning




| Name | Kernel | Str. | Ch I/O | InpRes | OutRes | Input |
|---|---|---|---|---|---|---|
| conv1 | 7×7 | 2 | 6/64 | 768×384 | 384×192 | Images |
| conv2 | 5×5 | 2 | 64/128 | 384×192 | 192×96 | conv1 |
| conv3a | 5×5 | 2 | 128/256 | 192×96 | 96×48 | conv2 |
| conv3b | 3×3 | 1 | 256/256 | 96×48 | 96×48 | conv3a |
| conv4a | 3×3 | 2 | 256/512 | 96×48 | 48×24 | conv3b |
| conv4b | 3×3 | 1 | 512/512 | 48×24 | 48×24 | conv4a |
| conv5a | 3×3 | 2 | 512/512 | 48×24 | 24×12 | conv4b |
| conv5b | 3×3 | 1 | 512/512 | 24×12 | 24×12 | conv5a |
| conv6a | 3×3 | 2 | 512/1024 | 24×12 | 12×6 | conv5b |
| conv6b | 3×3 | 1 | 1024/1024 | 12×6 | 12×6 | conv6a |
| pr6+loss6 | 3×3 | 1 | 1024/1 | 12×6 | 12×6 | conv6b |
| upconv5 | 4×4 | 2 | 1024/512 | 12×6 | 24×12 | conv6b |
| iconv5 | 3×3 | 1 | 1025/512 | 24×12 | 24×12 | upconv5+pr6+conv5b |
| pr5+loss5 | 3×3 | 1 | 512/1 | 24×12 | 24×12 | iconv5 |
| upconv4 | 4×4 | 2 | 512/256 | 24×12 | 48×24 | iconv5 |
| iconv4 | 3×3 | 1 | 769/256 | 48×24 | 48×24 | upconv4+pr5+conv4b |
| pr4+loss4 | 3×3 | 1 | 256/1 | 48×24 | 48×24 | iconv4 |
| upconv3 | 4×4 | 2 | 256/128 | 48×24 | 96×48 | iconv4 |
| iconv3 | 3×3 | 1 | 385/128 | 96×48 | 96×48 | upconv3+pr4+conv3b |
| pr3+loss3 | 3×3 | 1 | 128/1 | 96×48 | 96×48 | iconv3 |
| upconv2 | 4×4 | 2 | 128/64 | 96×48 | 192×96 | iconv3 |
| iconv2 | 3×3 | 1 | 193/64 | 192×96 | 192×96 | upconv2+pr3+conv2 |
| pr2+loss2 | 3×3 | 1 | 64/1 | 192×96 | 192×96 | iconv2 |
| upconv1 | 4×4 | 2 | 64/32 | 192×96 | 384×192 | iconv2 |
| iconv1 | 3×3 | 1 | 97/32 | 384×192 | 384×192 | upconv1+pr2+conv1 |
| pr1+loss1 | 3×3 | 1 | 32/1 | 384×192 | 384×192 | iconv1 |

Figure from https://arxiv.org/abs/1708.09204

# Example: Pyramid Stereo Matching Network

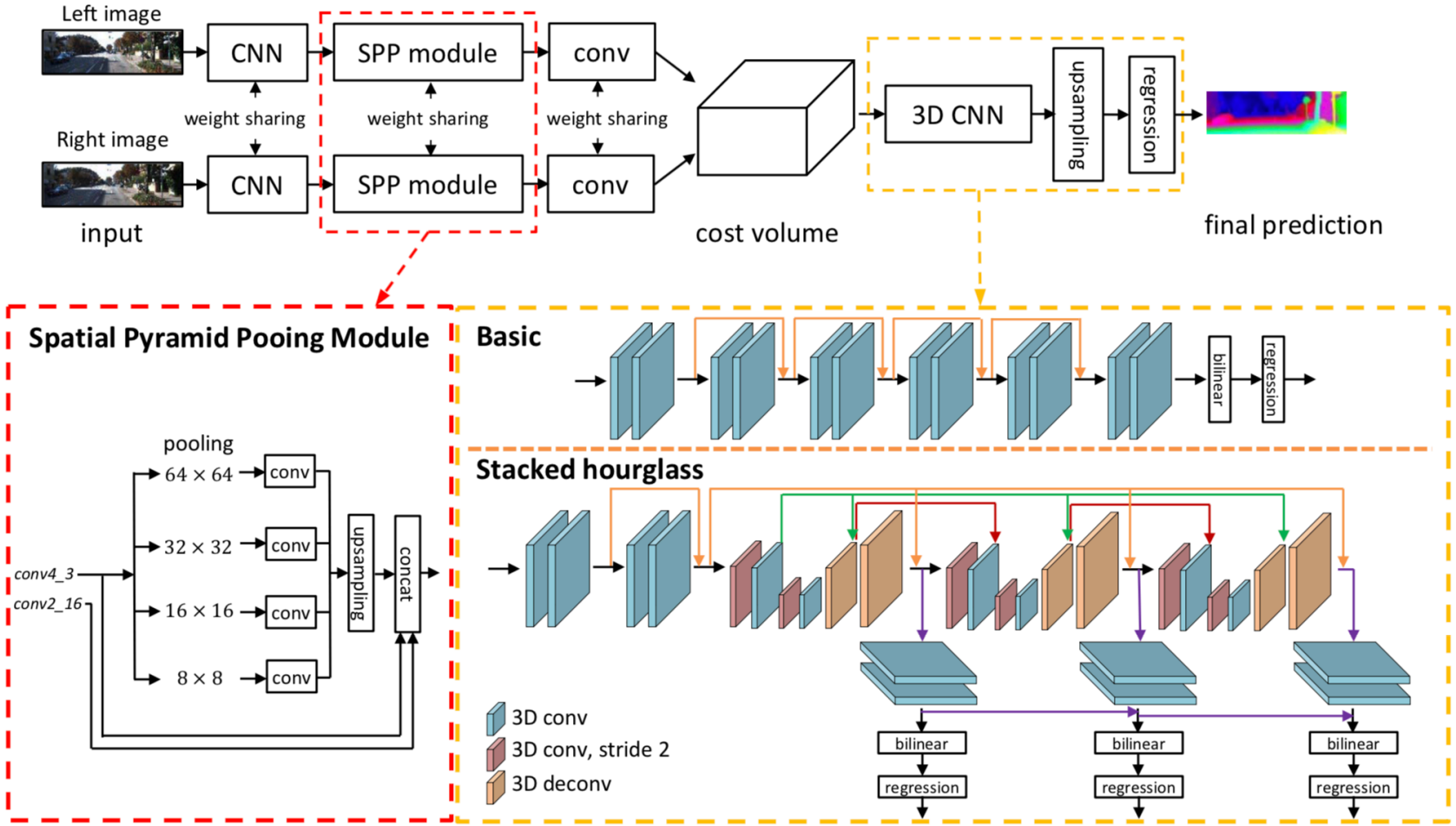



Figure from https://arxiv.org/abs/1803.08669

# Evaluation

- KITTI criteria
  - Errors exceed 3 pixels and 5 % of the true value
  - Errors can be categorized as
    - Foreground pixels
    - Background pixels
    - All pixels

# Depth Estimation From Monocular Video

For details see: Depth prediction without the sensors: leveraging structure for unsupervised learning from monocular videos

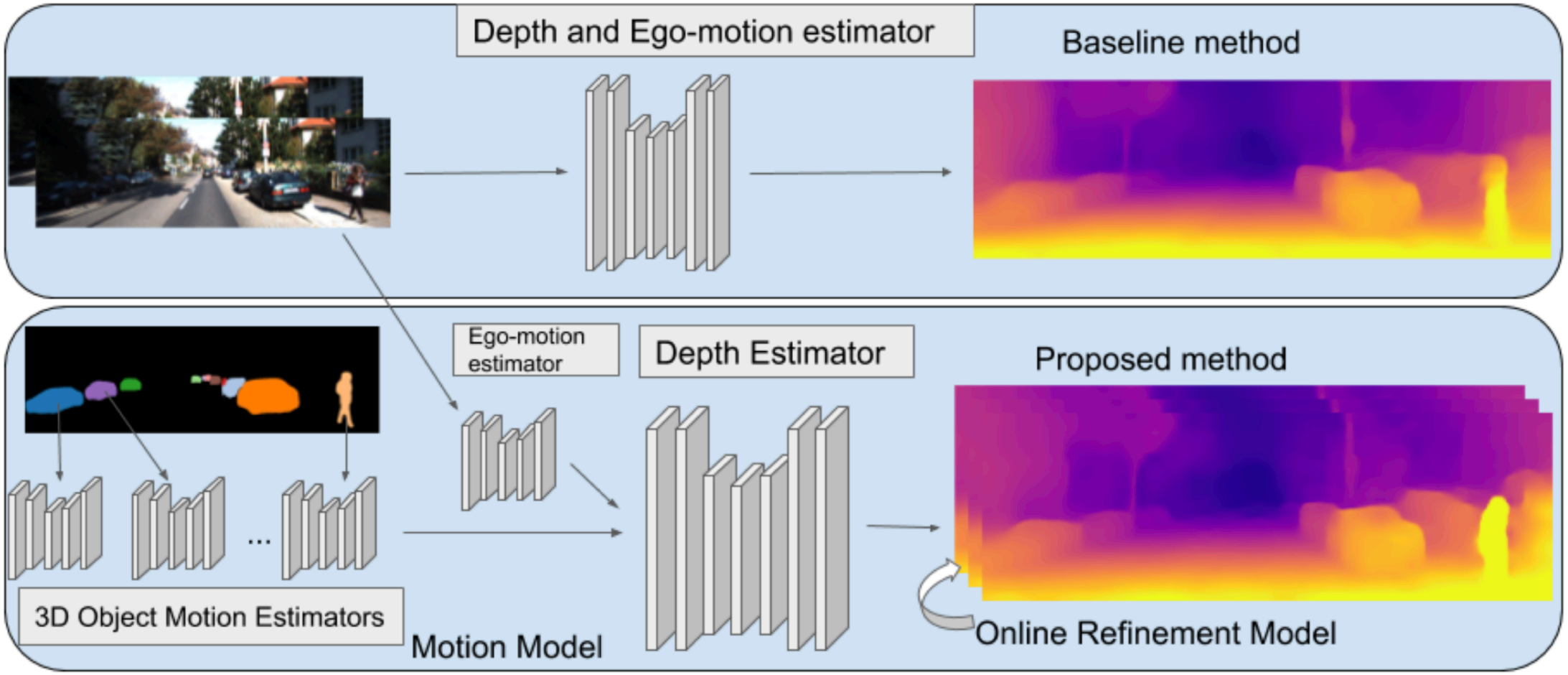



Figure from https://arxiv.org/abs/1811.06152

# Motion Estimation

# Goal

- Find the relative motion between pixels in 1 frame in time to the next
- This is referred to as optical flow (loosely)
- Note that there are single image motion estimation methods but these are not discussed here

Figure from http://sintel.is.tue.mpg.de

# Fundamentals

- 2 images separated in time, find the same pixel in both
  - Again, this is a classification problem
  - Though unlike stereo where the search is over a line, now it can be in 2D
- Can improve with the addition of object segmentation
  - Do object segmentation on both images
  - Figure out where an object moves from 1 image to the next
  - Doesn't fully handle deformations but also not bad if changes are small
- Can extend to more than 2 images
- Consistency between image at time 1, image at time 2 and motion estimation
  - Can use as an error estimate to potentially refine

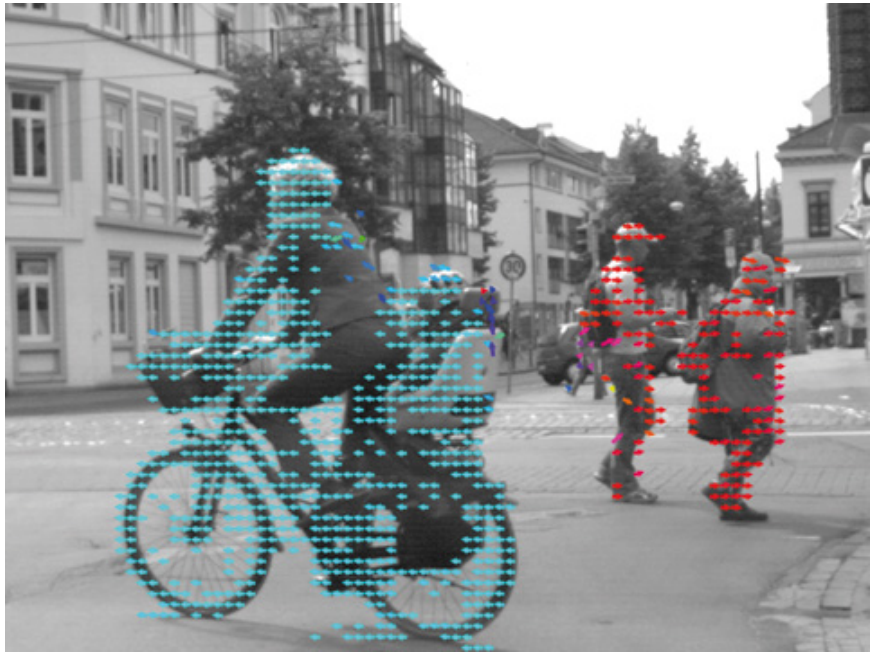


Figure from https://www.commonvisionblox.com/en/products/series/cvb-optical-flow/

# Data

- Laundry list
  - KITTI
  - MPI Sintel
  - Flying Chairs
  - Middlebury
  - HD1K

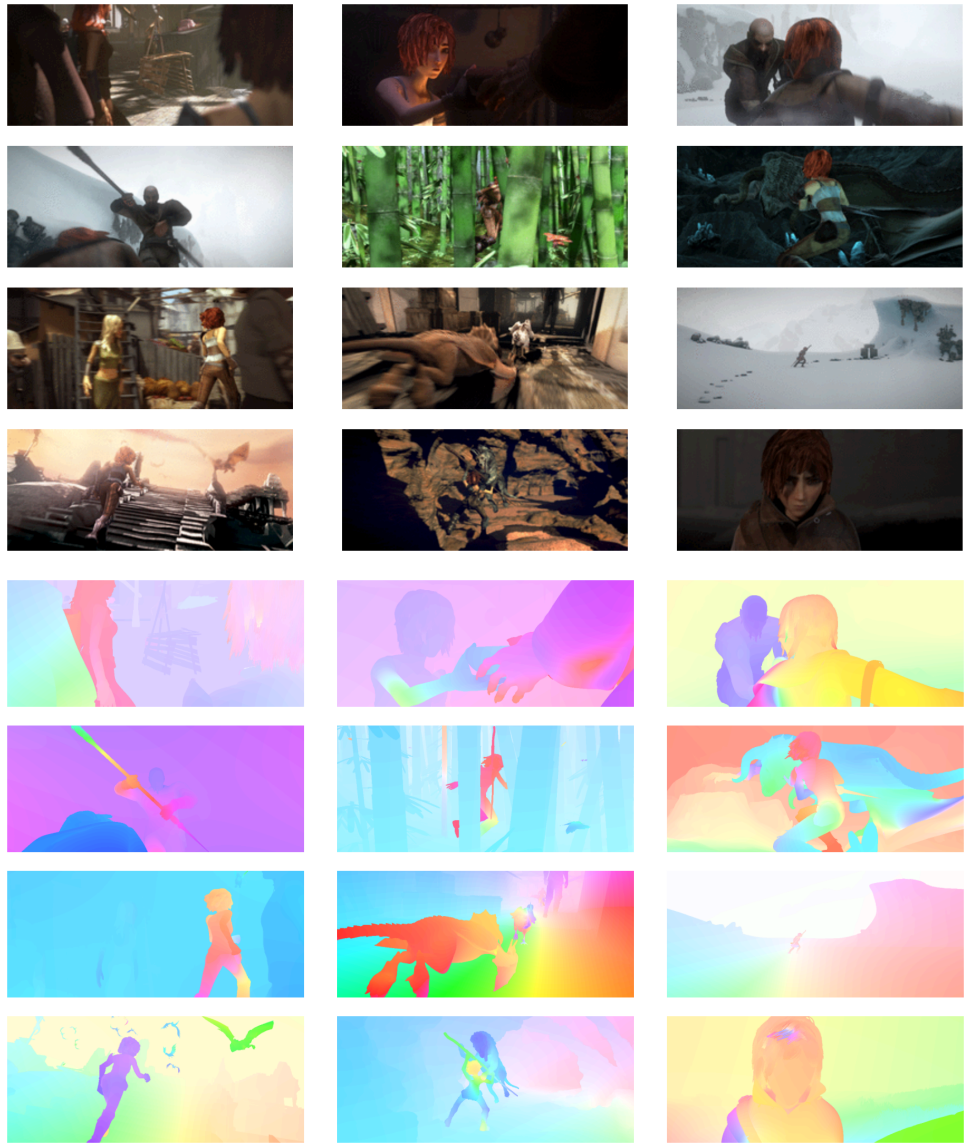


Figure from http://sintel.is.tue.mpg.de

# Example: FlowNet

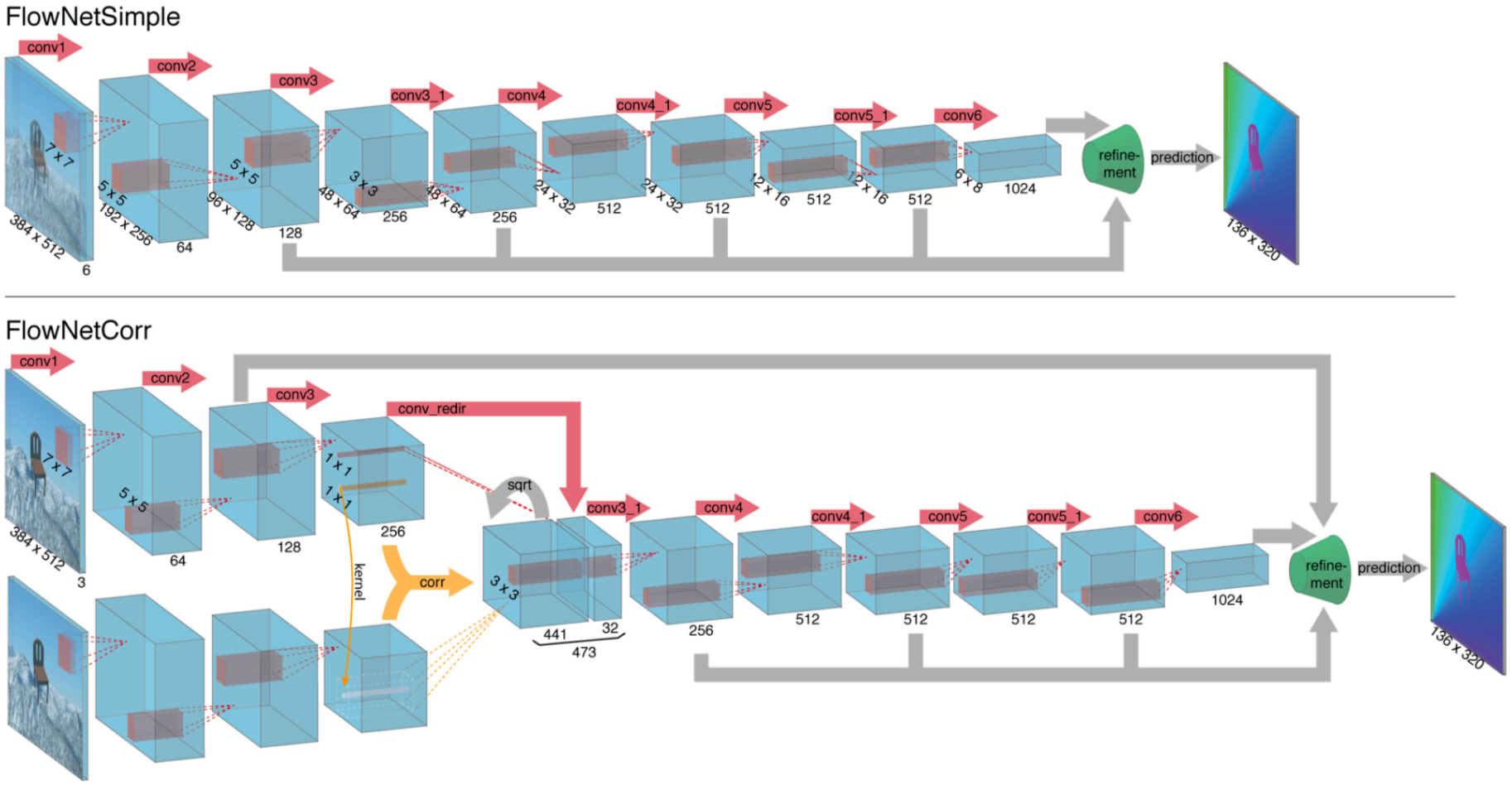



Figure from https://arxiv.org/abs/1504.06852

# Example: FlowNet

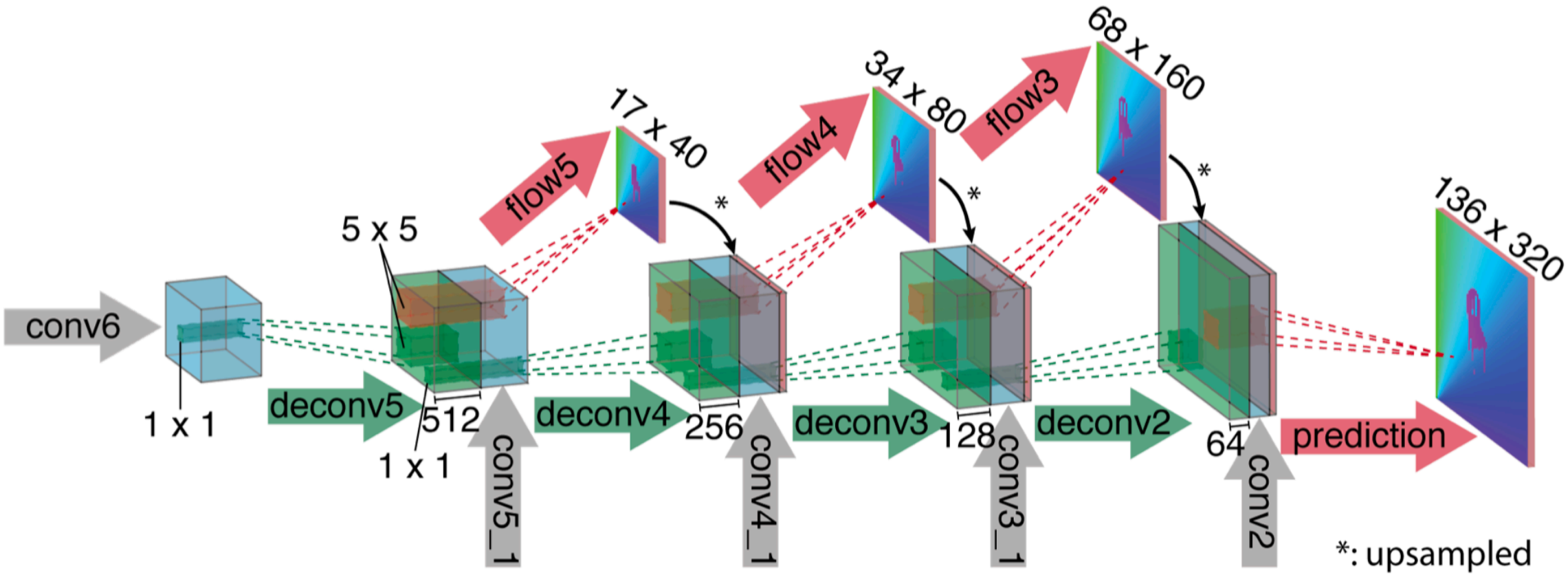



Figure from https://arxiv.org/abs/1504.06852

# Example: FlowNet 2.0

The sequel is better than the original (just like The Empire Strikes Back, don't argue with me on this, it's a fact)

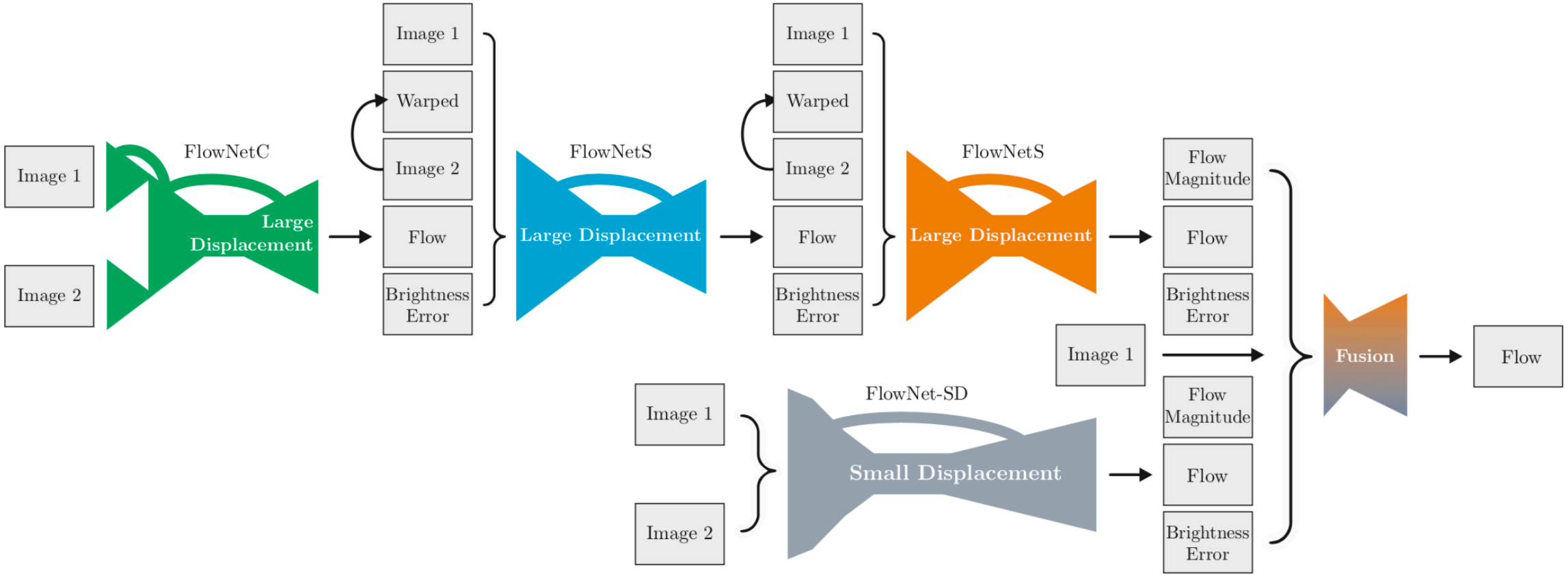



Figure from https://arxiv.org/abs/1612.01925

# Evaluation

- Basically the same as stereo depth estimation

# References

# Data

- Google AI datasets
  - https://ai.google/tools/datasets/
- Waymo open dataset
  - https://waymo.com/open/

- Yet another computer vision index to datasets (YACVID)
  - https://riemenschneider.hayko.at/vision/dataset/
- CV datasets on the web
  - http://www.cvpapers.com/datasets.html
- Computer vision online datasets
  - https://computervisiononline.com/datasets
- UC Irvine machine learning repository
  - https://archive.ics.uci.edu/ml/index.php
- 25 open datasets for deep learning every data scientist must work with
  - https://www.analyticsvidhya.com/blog/2018/03/comprehensive-collection-deep-learning-datasets/

# Data

- CIFAR
  - https://www.cs.toronto.edu/~kriz/cifar.html
- Cityscapes dataset
  - https://www.cityscapes-dataset.com
- Common objects in context
  - http://cocodataset.org
- ETH3D
  - https://www.eth3d.net
- Flying chairs
  - https://lmb.informatik.uni-freiburg.de/resources/datasets/FlyingChairs.en.html
- HD1K
  - http://hci-benchmark.org
- ImageNet
  - http://www.image-net.org

# Data

- The KITTI vision benchmark suite
  - http://www.cvlibs.net/datasets/kitti/
- Middlebury
  - http://vision.middlebury.edu/flow/
- Mapillary vistas
  - https://www.mapillary.com/dataset/vistas
- THE MNIST database of handwritten digits
  - http://yann.lecun.com/exdb/mnist/
- Fashion MNIST
  - https://github.com/zalandoresearch/fashion-mnist
- MPI Sintel flow dataset
  - http://sintel.is.tue.mpg.de
- Open images dataset V4
  - https://storage.googleapis.com/openimages/web/index.html

# Data

- The PASCAL visual object classes homepage
  - http://host.robots.ox.ac.uk/pascal/VOC/
- ScanNet
  - http://kaldir.vc.in.tum.de/scannet_benchmark/
- The street view house numbers (SVHN) dataset
  - http://ufldl.stanford.edu/housenumbers/

# Image Capture

- Brown CSCI 1290: computational photography and image manipulation
  - http://cs.brown.edu/courses/csci1290/
- Digital photography
  - https://sites.google.com/site/marclevoylectures/
  - https://sites.google.com/site/marclevoylectures/schedule

- ARM Mali camera
  - https://developer.arm.com/products/graphics-and-multimedia/mali-camera
- HotChips 2018: the Google pixel visual core live blog
  - https://www.anandtech.com/show/13241/hot-chips-2018-the-google-pixel-visual-core-live-blog
- Pixel visual core: image processing and machine learning on pixel 2
  - https://www.blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/

# Image Classification

- See design lecture notes

# Pixel Classification

- Fully convolutional networks for semantic segmentation
  - https://arxiv.org/abs/1411.4038
- Semantic image segmentation with deep convolutional nets and fully connected CRFs
  - https://arxiv.org/abs/1412.7062
- BoxSup: exploiting bounding boxes to supervise convolutional networks for semantic segmentation
  - https://arxiv.org/abs/1503.01640
- U-net: convolutional networks for biomedical image segmentation
  - https://arxiv.org/abs/1505.04597
- SegNet: a deep convolutional encoder-decoder architecture for image segmentation
  - https://arxiv.org/abs/1511.00561
- Bridging category-level and instance-level semantic image segmentation
  - https://arxiv.org/abs/1605.06885
- DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs
  - https://arxiv.org/abs/1606.00915

# Pixel Classification

- The one hundred layers tiramisu: fully convolutional DenseNets for semantic segmentation
  - https://arxiv.org/abs/1611.09326
- RefineNet: multi-path refinement networks for high-resolution semantic segmentation
  - https://arxiv.org/abs/1611.06612
- Full-resolution residual networks for semantic segmentation in street scenes
  - https://arxiv.org/abs/1611.08323
- Pyramid scene parsing network
  - https://arxiv.org/abs/1612.01105
- Large kernel matters - improve semantic segmentation by global convolutional network
  - https://arxiv.org/abs/1703.02719
- Not all pixels are equal: difficulty-aware semantic segmentation via deep layer cascade
  - https://arxiv.org/abs/1704.01344
- Rethinking atrous convolution for semantic image segmentation
  - https://arxiv.org/abs/1706.05587

# Pixel Classification

- Deep layer aggregation
  - https://arxiv.org/abs/1707.06484
- Encoder-decoder with atrous separable convolution for semantic image segmentation
  - https://arxiv.org/abs/1802.02611
- Searching for efficient multi-scale architectures for dense image prediction
  - https://arxiv.org/abs/1809.04184

# Object Detection

- DensePose
  - https://github.com/facebookresearch/DensePose
- Detectron
  - https://github.com/facebookresearch/Detectron
- TensorFlow object detection API
  - https://github.com/tensorflow/models/tree/master/research/object_detection

- Speed/accuracy trade-offs for modern convolutional object detectors
  - https://arxiv.org/abs/1611.10012
- Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3)
  - https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359
- Review of deep learning algorithms for object detection
  - https://medium.com/comet-app/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852
- Deep learning for generic object detection: a survey
  - https://arxiv.org/abs/1809.02165

# Object Detection

- OverFeat: integrated recognition, localization and detection using convolutional networks
  - https://arxiv.org/abs/1312.6229
- You only look once: unified, real-time object detection
  - https://arxiv.org/abs/1506.02640
- YOLO9000: better, faster, stronger
  - https://arxiv.org/abs/1612.08242
- YOLOv3: an incremental improvement
  - https://arxiv.org/abs/1804.02767
- SSD: single shot multibox detector
  - https://arxiv.org/abs/1512.02325
- Focal loss for dense object detection
  - https://arxiv.org/abs/1708.02002

# Object Detection

- Rich feature hierarchies for accurate object detection and semantic segmentation
  - https://arxiv.org/abs/1311.2524
- Spatial pyramid pooling in deep convolutional networks for visual recognition
  - https://arxiv.org/abs/1406.4729
- Fast R-CNN
  - https://arxiv.org/abs/1504.08083
- Faster R-CNN: towards real-time object detection with region proposal networks
  - https://arxiv.org/abs/1506.01497
- R-FCN: object detection via region-based fully convolutional networks
  - https://arxiv.org/abs/1605.06409
- Feature pyramid networks for object detection
  - https://arxiv.org/abs/1612.03144
- Cascade R-CNN: delving into high quality object detection
  - https://arxiv.org/abs/1712.00726
- Object detection and classification using R-CNNs
  - http://www.telesens.co/2018/03/11/object-detection-and-classification-using-r-cnns/

# Object Detection

- Soft-NMS - Improving Object Detection With One Line of Code
  - https://arxiv.org/abs/1704.04503
- Learning non-maximum suppression
  - https://arxiv.org/abs/1705.02950
- Object detection in 20 years: a survey
  - https://arxiv.org/abs/1905.05055

- The Pascal visual object classes (VOC) challenge
  - http://host.robots.ox.ac.uk/pascal/VOC/pubs/everingham10.pdf
- What makes for effective detection proposals?
  - https://arxiv.org/abs/1502.05082
- Metrics for object detection
  - https://github.com/rafaelpadilla/Object-Detection-Metrics

# Object Segmentation

- Instance-aware semantic segmentation via multi-task network cascades
  - https://arxiv.org/abs/1512.04412
- Instance-sensitive fully convolutional networks
  - https://arxiv.org/abs/1603.08678
- R-FCN: object detection via region-based fully convolutional networks
  - https://arxiv.org/abs/1605.06409
- Mask R-CNN
  - https://arxiv.org/abs/1703.06870
- TensorMask: a foundation for dense object segmentation
  - https://arxiv.org/abs/1903.12174

# Depth Estimation

- A taxonomy and evaluation of dense two-frame stereo correspondence algorithms
  - http://vision.middlebury.edu/stereo/taxonomy-IJCV.pdf
- On building an accurate stereo matching system on graphics hardware
  - http://www.nlpr.ia.ac.cn/2011papers/gjhy/gh75.pdf
- Depth estimation from stereo cameras
  - http://www.cs.tut.fi/~suominen/SGN-1656-stereo/stereo_instructions.pdf
- Depth from stereo
  - http://www.cs.toronto.edu/~fidler/slides/2015/CSC420/lecture12_hres.pdf

# Depth Estimation

- Computing the stereo matching cost with a convolutional neural network
  - https://arxiv.org/abs/1409.4326
- Stereo matching by training a convolutional neural network to compare image patches
  - https://arxiv.org/abs/1510.05970
- A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation
  - https://arxiv.org/abs/1512.02134
- Cascade residual learning: a two-stage convolutional neural network for stereo matching
  - https://arxiv.org/abs/1708.09204
- EdgeStereo: a context integrated residual pyramid network for stereo matching
  - https://arxiv.org/abs/1803.05196
- Pyramid stereo matching network
  - https://arxiv.org/abs/1803.08669
- On the importance of stereo for accurate depth estimation: an efficient semi-supervised deep neural network approach
  - https://arxiv.org/abs/1803.09719

# Depth Estimation

- Evaluation of CNN-based single-image depth estimation methods
  - https://arxiv.org/abs/1805.01328
- Practical deep stereo (PDS): toward applications-friendly deep stereo matching
  - https://arxiv.org/abs/1806.01677
- Learning depth with convolutional spatial propagation network
  - https://arxiv.org/abs/1810.02695
- Pseudo-LiDAR from visual depth estimation: bridging the gap in 3D object detection for autonomous driving
  - https://arxiv.org/abs/1812.07179

- Detect, replace, refine: deep structured prediction for pixel wise labeling
  - https://arxiv.org/abs/1612.04770

# Motion Estimation

- DeepFlow: large displacement optical flow with deep matching
  - https://hal.inria.fr/hal-00873592
- EpicFlow: edge-preserving interpolation of correspondences for optical flow
  - https://arxiv.org/abs/1501.02565
- FlowNet: learning optical flow with convolutional networks
  - https://arxiv.org/abs/1504.06852
- PatchBatch: a batch augmented loss for optical flow
  - https://arxiv.org/abs/1512.01815
- Exploiting semantic information and deep matching for optical flow
  - https://arxiv.org/abs/1604.01827
- CNN-based patch matching for optical flow with thresholded hinge embedding loss
  - https://arxiv.org/abs/1607.08064
- Optical flow estimation using a spatial pyramid network
  - https://arxiv.org/abs/1611.00850

# Motion Estimation

- FlowNet 2.0: evolution of optical flow estimation with deep networks
  - https://arxiv.org/abs/1612.01925
- Scene flow estimation: a survey
  - https://arxiv.org/abs/1612.02590
- FusionSeg: learning to combine motion and appearance for fully automatic segmentation of generic objects in videos
  - https://arxiv.org/abs/1701.05384
- Accurate optical flow via direct cost volume processing
  - https://arxiv.org/abs/1704.07325
- Optical flow in mostly rigid scenes
  - https://arxiv.org/abs/1705.01352
- PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume
  - https://arxiv.org/abs/1709.02371
- LiteFlowNet: a lightweight convolutional neural network for optical flow estimation
  - https://arxiv.org/abs/1805.07036

# Motion Estimation

- ProFlow: learning to predict optical flow
  - https://arxiv.org/abs/1806.00800
- Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation
  - https://arxiv.org/abs/1808.01838
- A fusion approach for multi-frame optical flow estimation
  - https://arxiv.org/abs/1810.10066
- A lightweight optical flow CNN - revisiting data fidelity and regularization
  - https://arxiv.org/abs/1903.07414
- Deep discrete flow
  - https://pdfs.semanticscholar.org/8d19/2bb3feae3e445b3b30948edee907a1d2324a.pdf
  - http://www.cvlibs.net/publications/Guney2016ACCV_supplementary.pdf
- Bounding boxes, segmentations and object coordinates: how important is recognition for 3d scene flow estimation in autonomous driving scenarios?
  - http://www.cvlibs.net/publications/Behl2017ICCV.pdf
- Motion and optical flow
  - https://web.eecs.umich.edu/~jjcorso/t/598F14/files/lecture_1015_motion.pdf

# Point Clouds

- VoxNet: a 3d convolutional neural network for real-time object recognition
  - https://www.ri.cmu.edu/pub_files/2015/9/voxnet_maturana_scherer_iros15.pdf
- Generative and discriminative voxel modeling with convolutional neural networks
  - https://arxiv.org/abs/1608.04236
- PointNet: deep learning on point sets for 3d classification and segmentation
  - https://arxiv.org/abs/1612.00593
- VoxelNet: end-to-end learning for point cloud based 3d object detection
  - https://arxiv.org/abs/1711.06396
- SEGCloud: semantic segmentation of 3d point clouds
  - https://arxiv.org/abs/1710.07563
- PointRCNN: 3d object proposal generation and detection from point cloud
  - https://arxiv.org/abs/1812.04244

# Point Clouds

- Deep Hough voting for 3d object detection in point clouds
  - https://arxiv.org/abs/1904.09664
- A convolutional decoder for point clouds using adaptive instance normalization
  - https://arxiv.org/abs/1906.11478
- Point-Voxel CNN for Efficient 3D Deep Learning
  - https://arxiv.org/abs/1907.03739
- DensePoint: Learning Densely Contextual Representation for Efficient Point Cloud Processing
  - https://arxiv.org/abs/1909.03669

# Scene Understanding

- Neural scene representation and rendering
  - https://deepmind.com/blog/neural-scene-representation-and-rendering/