Neural Networks

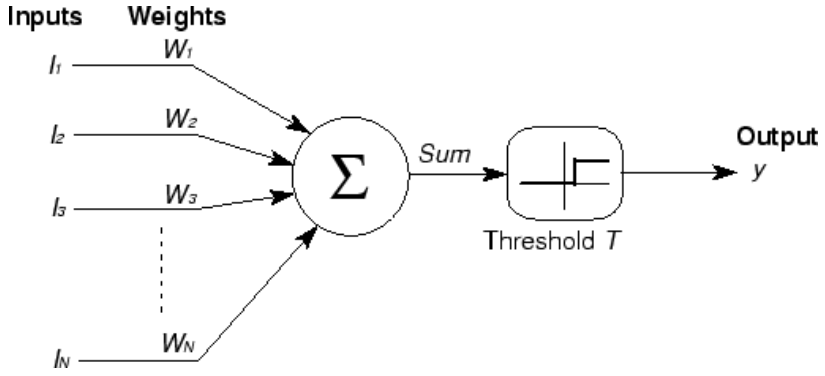
Applications to computer vision

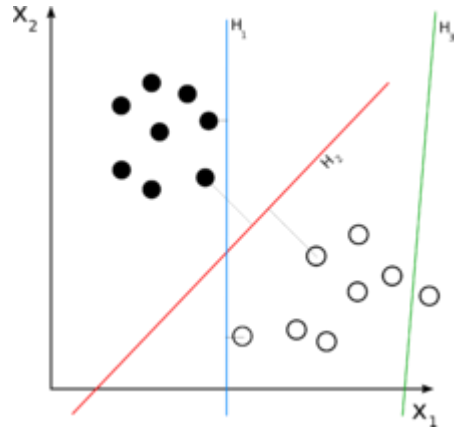
Overview

- History, motivations
- Introduction to Neural Networks
  - Supervised learning
  - Backpropagation
- Introduction to CNNs
  - Parts of a CNN
  - Types of CNN
- Image segmentation
  - Evaluation, IoU
  - Common architectures
- Object detection
  - Evaulation, mAP and AP
  - Architectures, single shot, two stage

Artificial Neural Networks

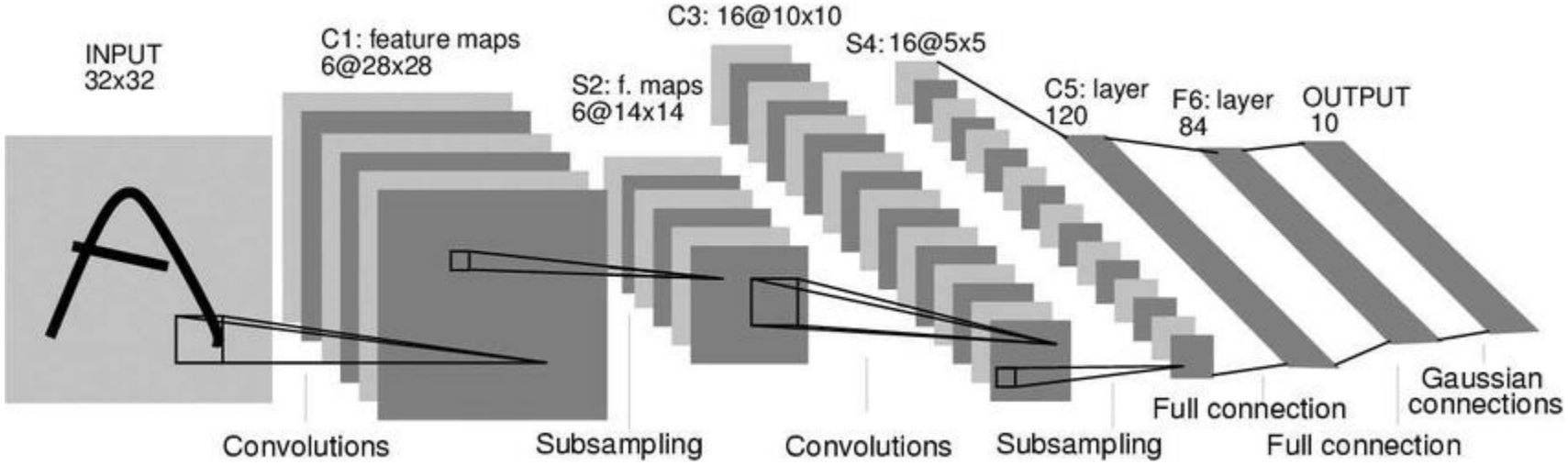
- Inspired by the brain
- The perceptron 1958 (Rosenblatt)
  - Linear classifier (also SVMs, logisitic regression)
  - Unable to learn the XOR function
  - Not differentiable
- Backpropagation 1975 (Werbos)
  - Enabled composition of networks built from multiple layers of differentiable functions
  - Long training/lacked processing resources
  - Vanishing gradient problem



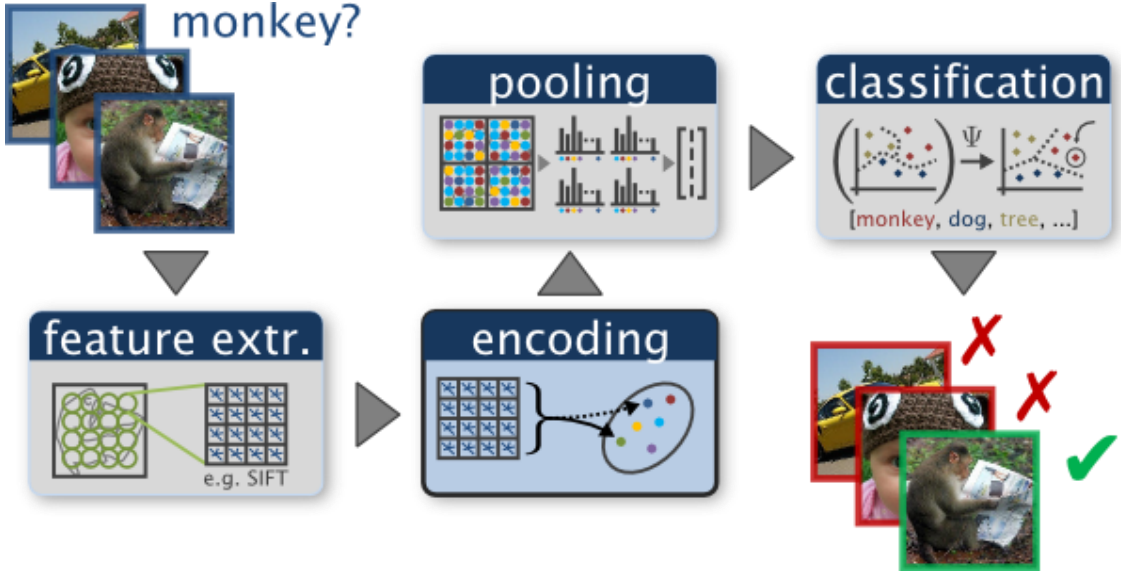


Convolutional Neural Networks (CNN)

- Image convolution is a differentiable function
  - Learnable parameters are the image filters
- 1988 - Invented for classifying handwritten digits (Yann LeCunn)



90s/2000s - largely ignored

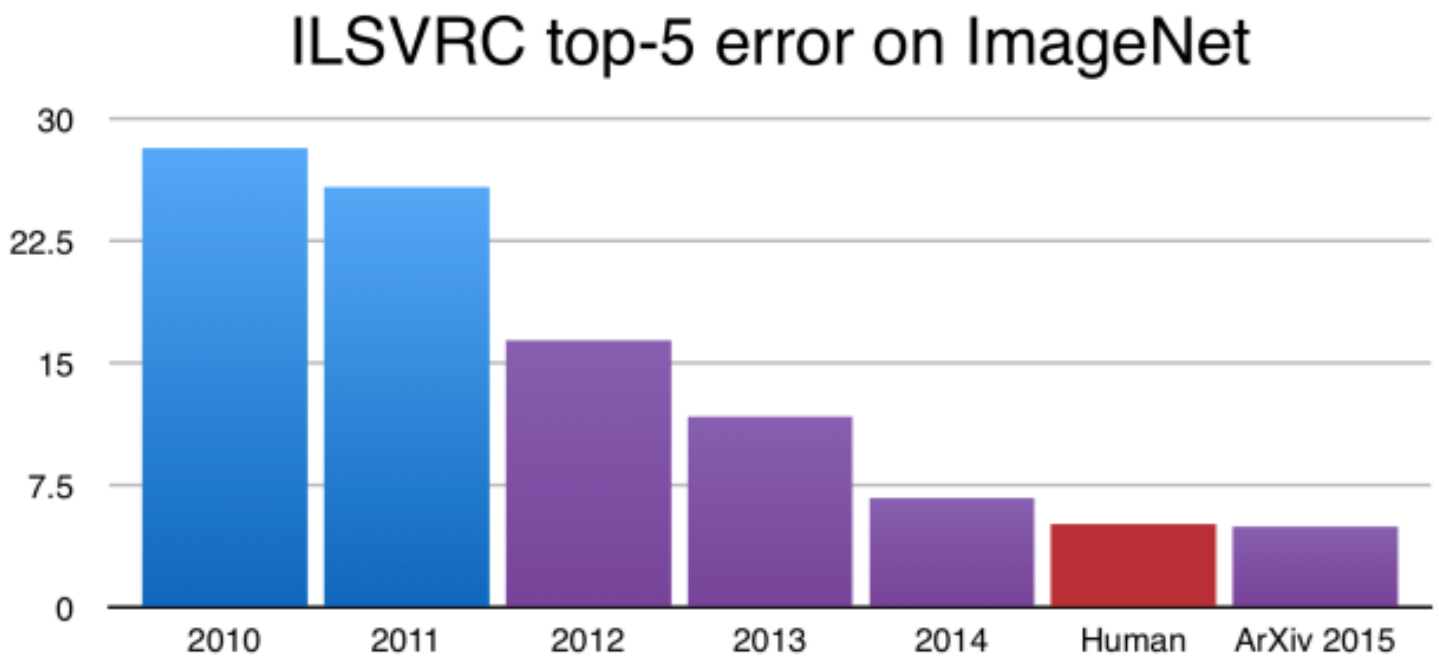


- Feature descriptor engineering - HoG, SIFT, SURF, ORB
- Unsupervised dimensionality reduction - Sparse coding, ICA, PCA
- Feature aggregation/pooling - BoVW, VLAD
- Classification - SVM, Nearest Neighbour

2009: imageNet 1k classification problem

- 1000 image classes, 1.4M images

ImageNet classification problem



Modern CNNs for visual recognition

- 2012 - Application of GPGPU to NNs for the ImageNet classification problem.
  - "ImageNet Classification with Deep Convolutional Neural Networks" Krizhevsky et al. 2011
- 2012-2015 - Refinement and introduction to basic CV applications
  - Object detection - COCO dataset
  - Image Segmentation
- 2021 - Currently CNNs solutions are (part of) state of the art of almost every computer vision application
  - 3D reconstruction: stereo, multi-view stereo, optical flow
  - 2d and 3d pose estimation
  - image generation (deep fake, texture synthesis etc.), in-painting, super resolution, style transfer
  - point clouds segmentation and object detection

Neural Networks in general

- “Differentiable programming” vs. “Deep Learning”
  - Reflects a more flexible usage
  - No longer a set of layers
- Supervised machine learning method (usually)
  - Examples with data (x), and target labels (y)
- Objective function: $J(\theta; x, y) = L(f(\theta; x), y)$
  - Model with parameters θ, and a function: $\hat{y} = f(\theta; x)$ to make predictions
  - Loss function: $L(y, \hat{y})$, giving the error of the predictions
- Minimise objective function, with mini-batch SGD
  - Compute gradient of the loss with respect to parameters ∇_θ for a batch of examples
  - Update weights $\theta_{i+1} = \theta_i - \alpha\nabla_\theta J(...)$ (α is a learning rate, a constant)

Computing derivatives

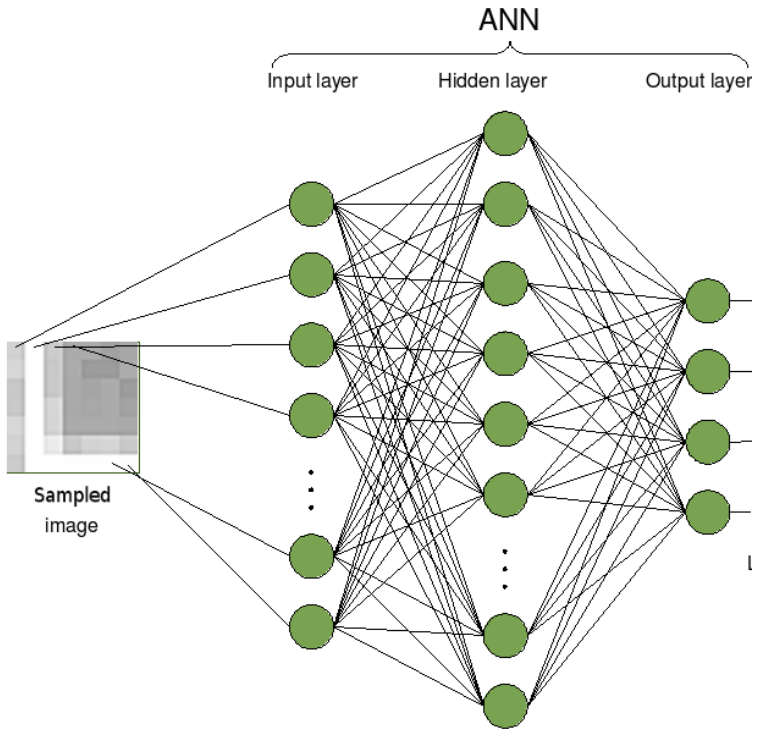
- In order to have the network “learn” we need to be able to compute the gradients with respect to the parameters ∇_θ
- By hand
- Symbolic differentiation
  - Also impractical for many parameters, terms may explode exponentially
- Finite differences
  - Compute $f(x + \varepsilon) - f(x)$
  - Simplest, but impractical for many parameters

Automatic differentiation

- Modern neural network libraries (e.g. pytorch) have automatic differentiation
  - No need to explicitly calculate derivatives, all variables track history automatically
- Forward mode
  - Dual numbers, a little similar to imaginary numbers
  - Two parts, real part and dual (epsilon) part ($z = a + b\varepsilon$), where $\varepsilon^2 = 0$
  - Multiple variables have multiple epsilons ($z = a + b\varepsilon_x + c\varepsilon_y$)
  - Dual-dual numbers for 2nd derivatives, etc.
- Reverse mode
  - Compute a graph during forward computation and store intermediate values
  - Work backwards from loss function error, utilizing the chain rule
  - Back propagation is a special case of reverse mode
  - Efficient for large numbers of parameters

What's wrong with (fully connected) NNs

- Images are very high dimensional
  - To a fully connected neural network an image shifted by a single pixel looks very different
- High dimensional inputs require a lot of data to cover the distribution
  - Large images are impractical at all
- The "curse of dimensionality"
  - At high dimensions all data becomes sparse

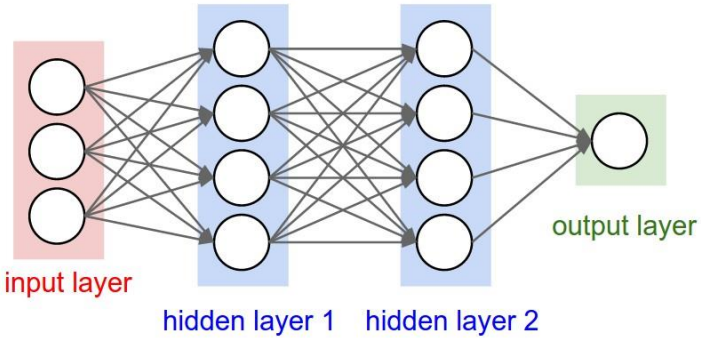


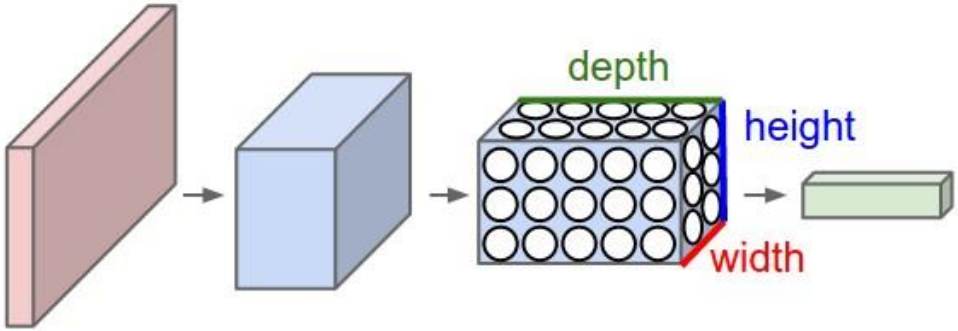
Solutions?

- Pixels aren’t independent, they’re assembled in a 2D grid
  - Nearby pixels are correlated, and..
- Invariance and equivariance
  - Images can be transformed (translated, and often scaled/rotated) without changing their content (or their class label)
  - Images can be transformed and their segmentation is transformed the same way
- Therefore, we can use operations exploiting these properties
  - Typically convolutions (CNN) - translationally equivariant, multi-scale approaches etc.
  - No longer require (as much) data
  - Different domains have different invariances/symmetries etc.

What is a Convolutional Neural Network?

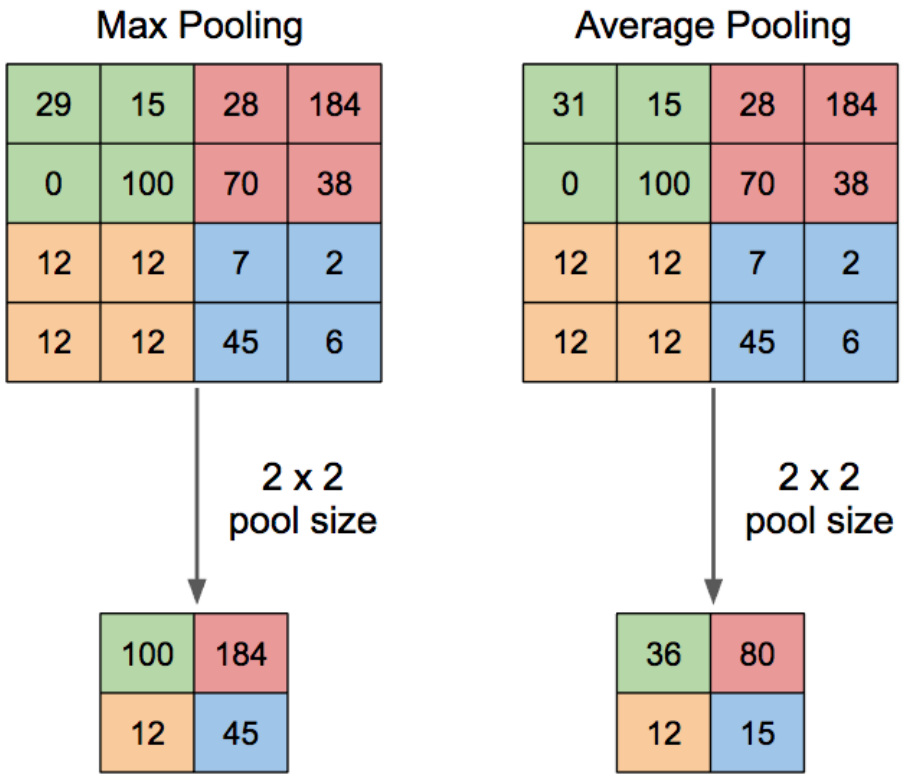
- A neural network which uses convolutions to operate on spatial data. E.g. Audio (1d), Images (2d), Video (3d)
- A set of kernels applied at every part of an image
  - The kernels are the parameters and are shared across an image (translational invariant)



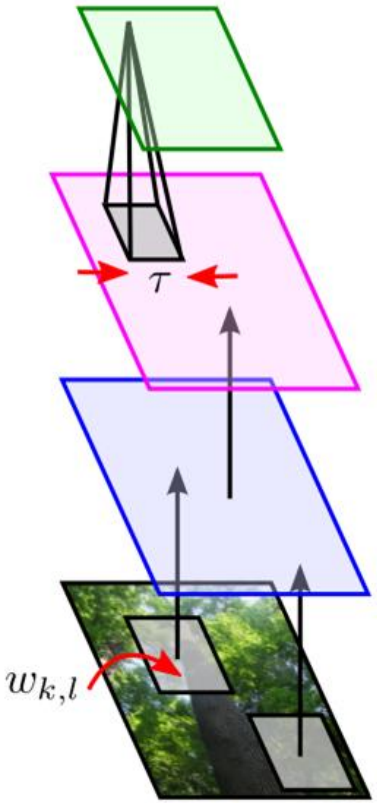


CS231n: Convolutional Neural Networks for Visual Recognition

Pooling/down sampling



Building blocks of a CNN



$$x_{i,j} = \max_{|k|<\tau,|l|<\tau} y_{i-k,j-l}$$

mean or subsample also used

pooling stage

$$y_{i,j} = f(a_{i,j})$$

e.g. $f(a) = [a]_+$

$f(a) = \text{sigmoid}(a)$

non-linear stage

$$a_{i,j} = \sum_{k,l} w_{k,l} z_{i-k,j-l}$$

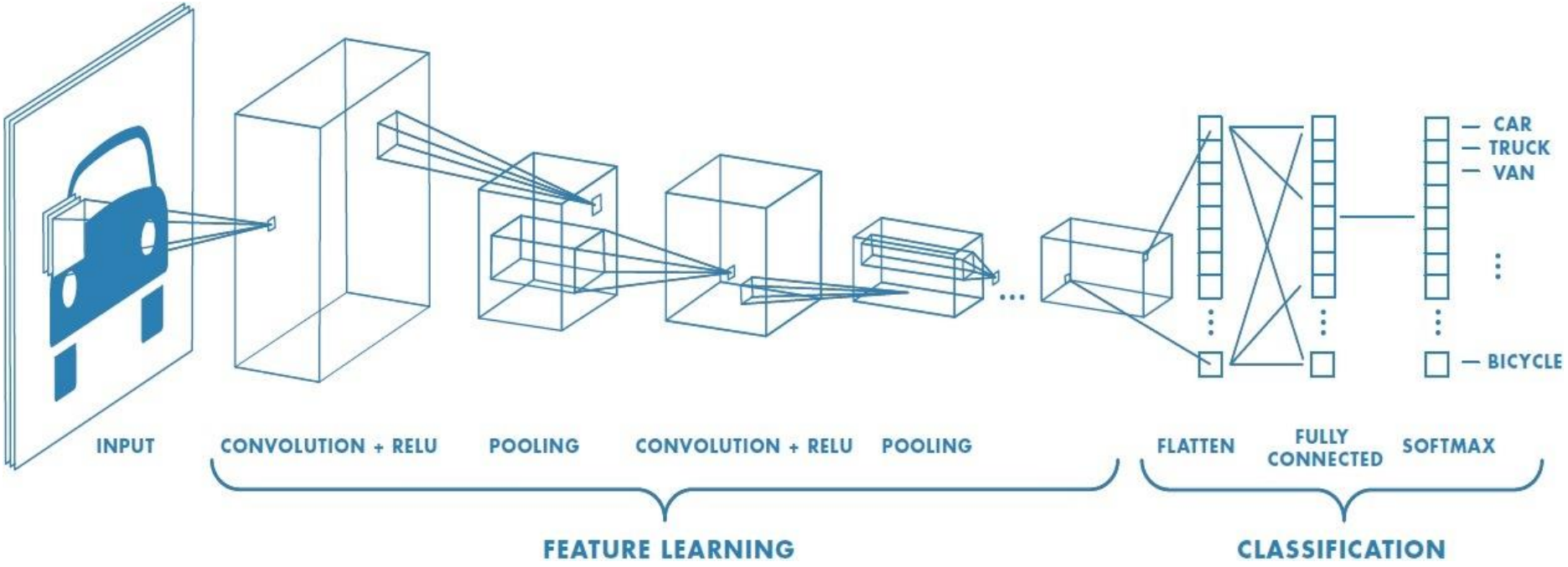
only parameters

convolutional stage

$z_{i,j}$

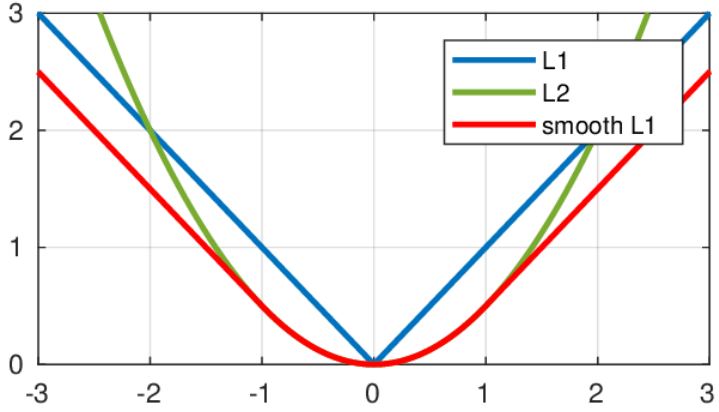
input image

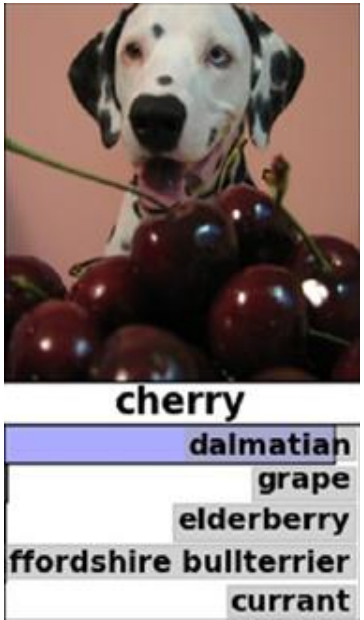
Example CNN


CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION

Output types

- Classification
  - Categorical variables
  - Softmax output with cross entropy loss
  - Sigmoid with binary cross entropy loss
- Regression
  - Continuous variables
  - Smooth L1, least square loss



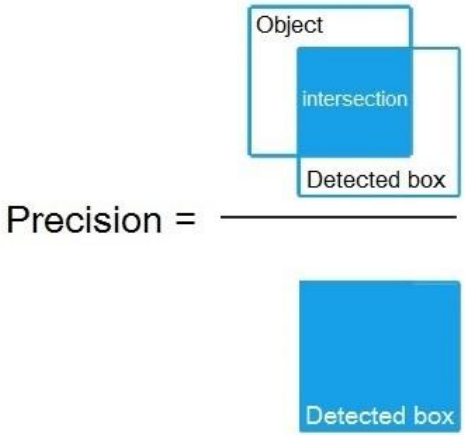


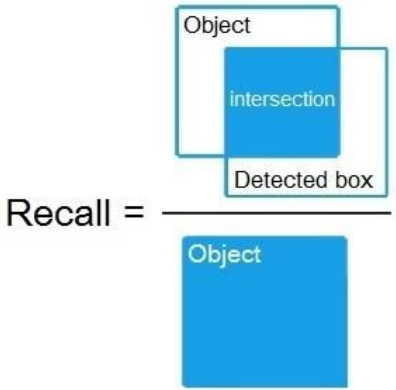
Dense Segmentation

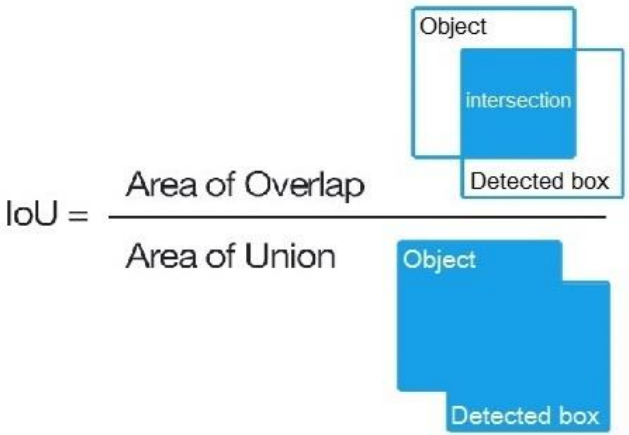
- Per pixel classification
- Irregular shaped objects
  - Can’t always fit a bounding box on things
- Scene understanding
  - E.g. find road surface for self driving car

Segmentation - measurement

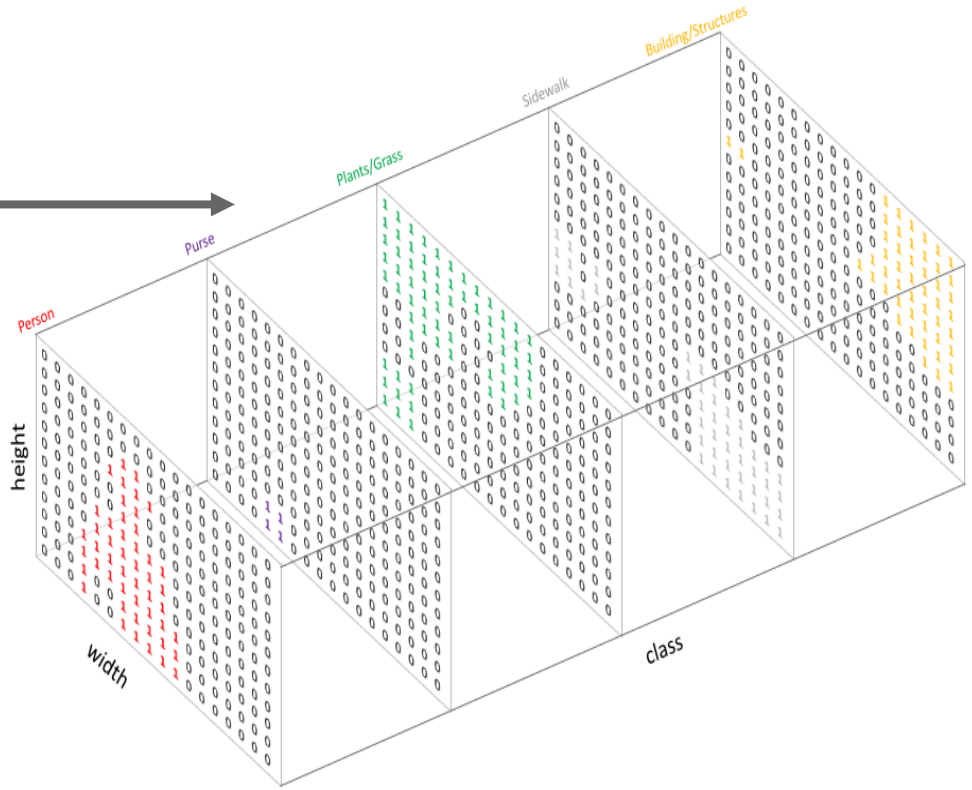
- Percent correct - problems with class imbalance
- Intersection Over Union (IOU) - single metric for class
  - Balance







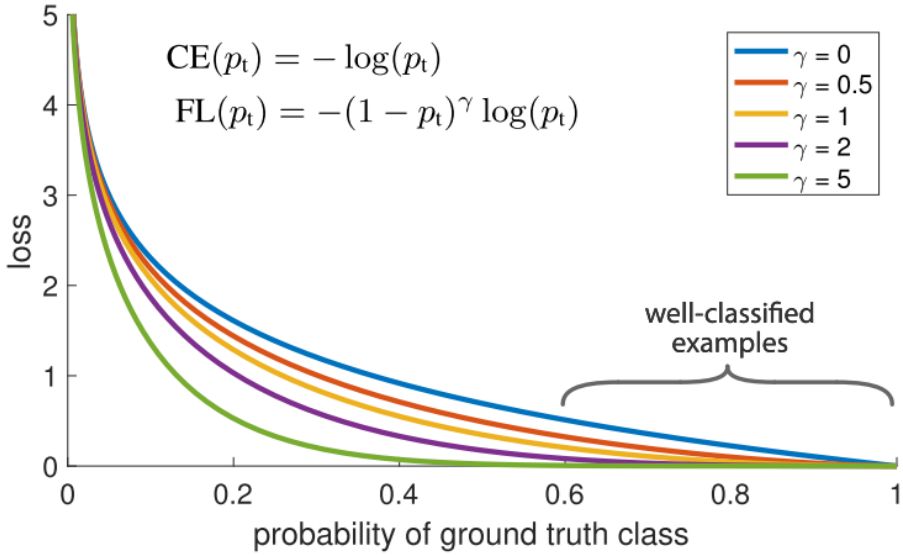
Target mask - one hot vectors



Loss functions

- Per pixel, softmax and cross entropy
  - Focal loss - to handle class imbalance
- Jaccard Index / soft IoU
  - |P| - sum of predictions (each 0 to 1)
  - |T| - size of positive targets

$$Jacc = 1 - \frac{|PT|}{|P^2| + |T^2| - |PT|}$$

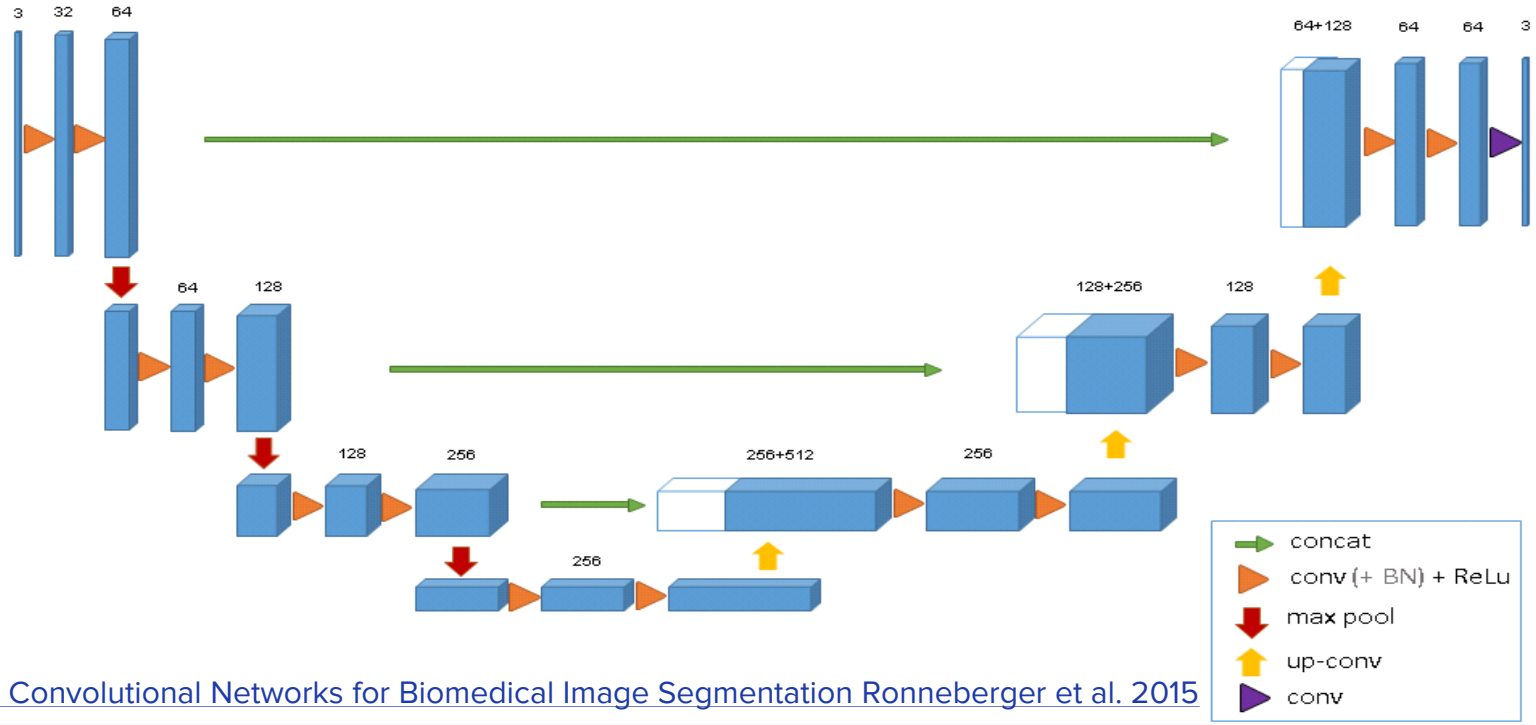


Focal Loss for Dense Object Detection Lin et al 2017

- Adversarial Loss (GAN)
  - Hard to design a loss function to impose constraints on the structure of output mask
  - Use a Neural Network to give the required gradient!

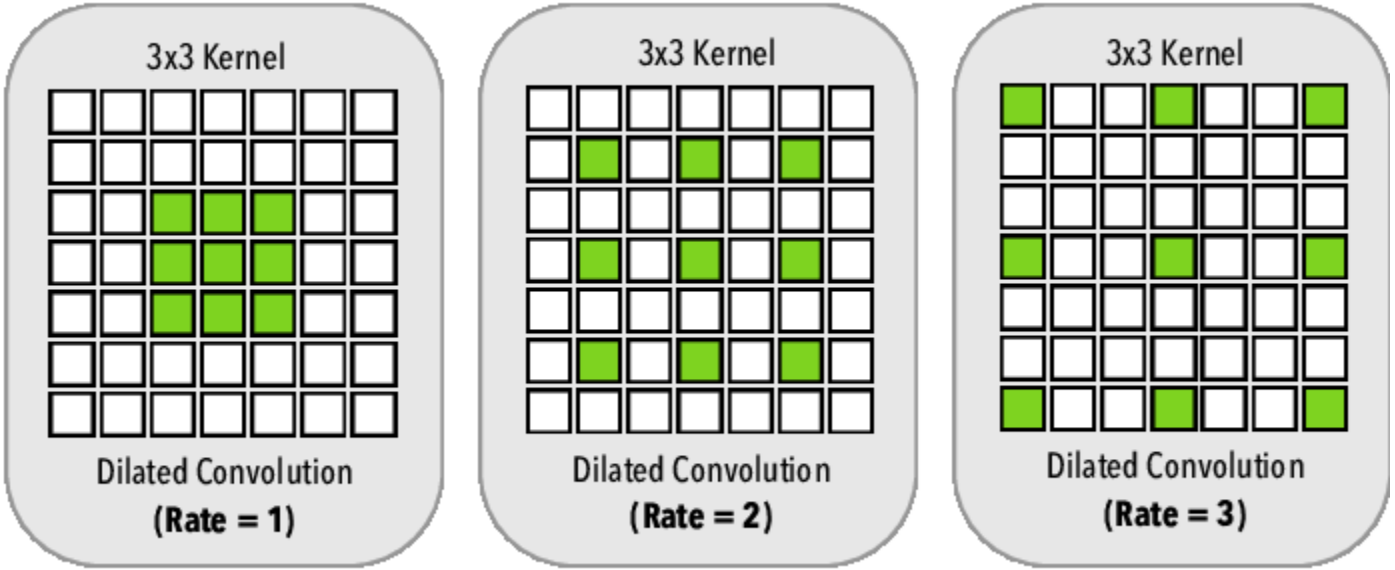
Segmentation - architecture

- Top down and bottom up



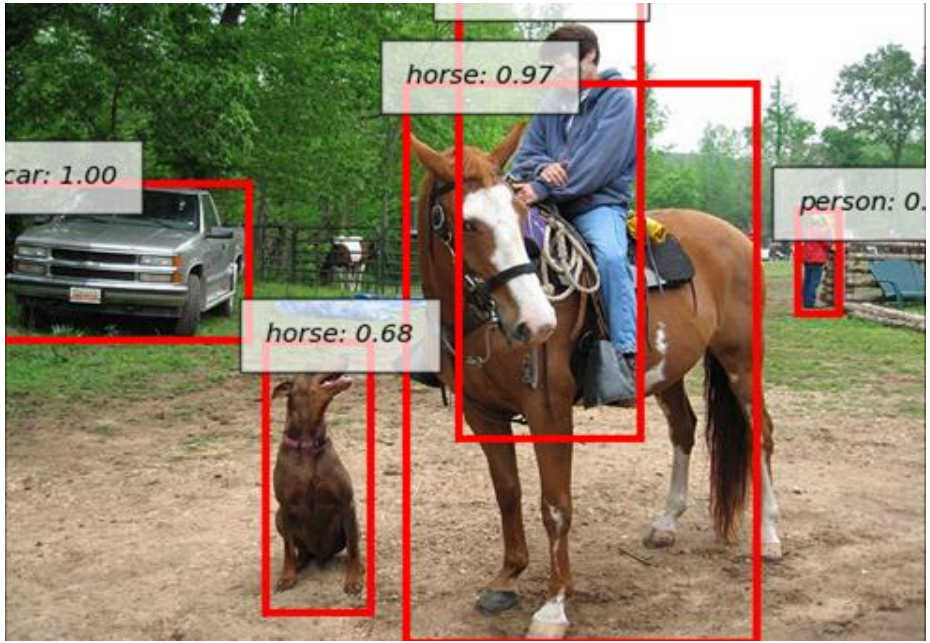
U-Net: Convolutional Networks for Biomedical Image Segmentation Ronneberger et al. 2015

Dilated/atrous Convolutions



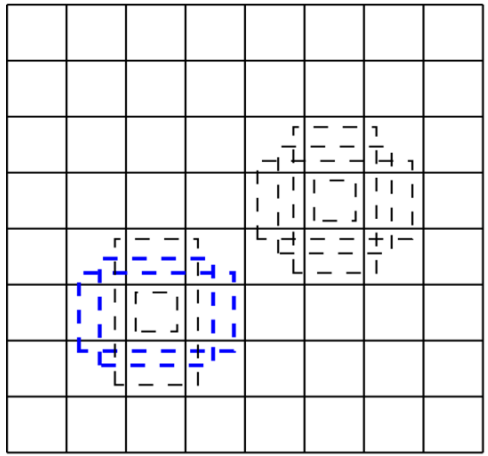
Object Detection

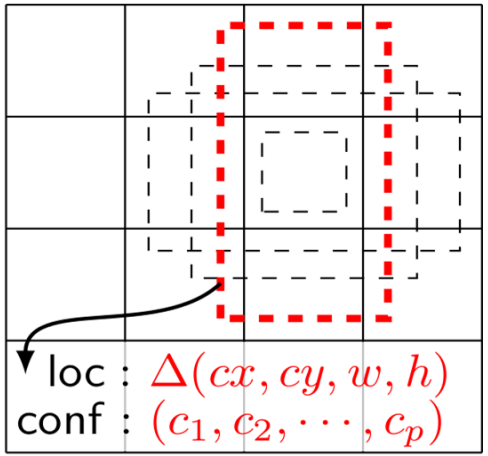
- Localisation plus classification
- Challenging because of variable number of outputs
- Datasets
  - Pascal VOC
  - Microsoft COCO

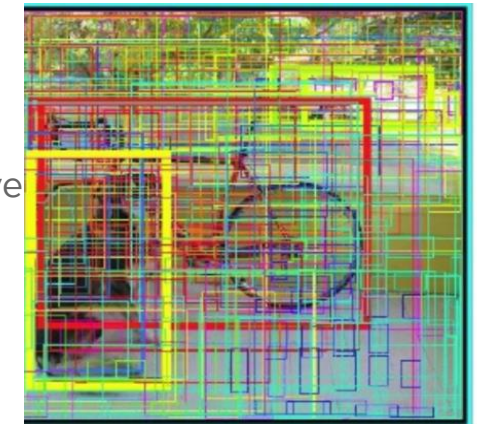


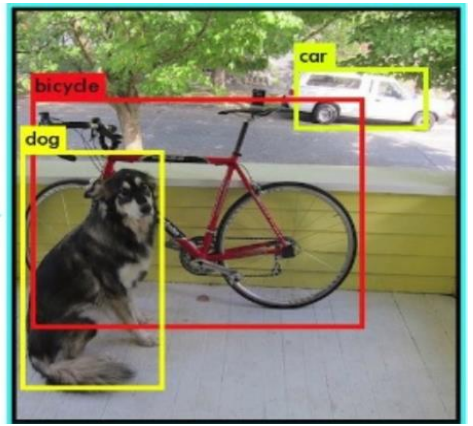
Object detection

- Anchor boxes
  - Class prediction (or lack) per anchor
  - Regression on box displacement and sizing
- Non maximum suppression
  - Problematic for closely overlapping objects
- Train by matching anchor boxes to ground truth boxes (iou > 0.5)
  - L1 loss for box regression for positive
  - Cross entropy on positive matches and negative box (iou < 0.4)



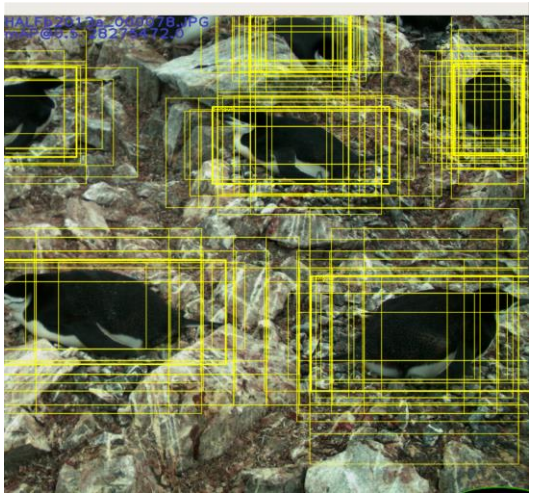




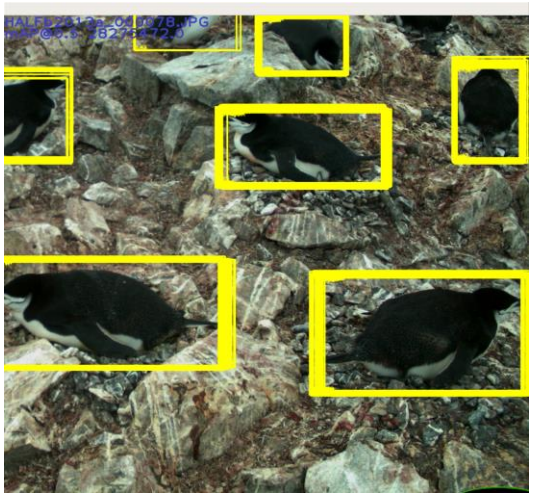


Object detection

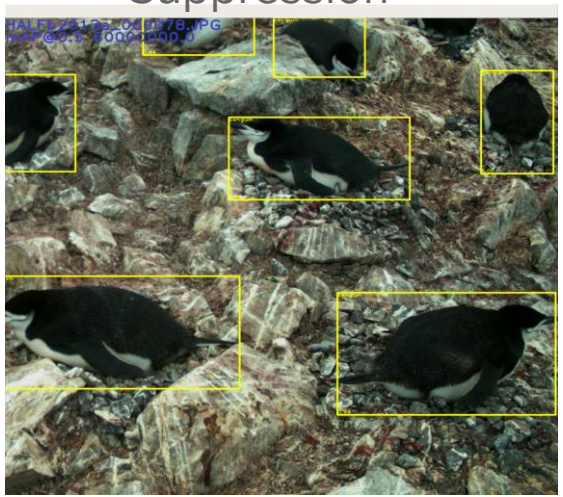
Classification



Regression

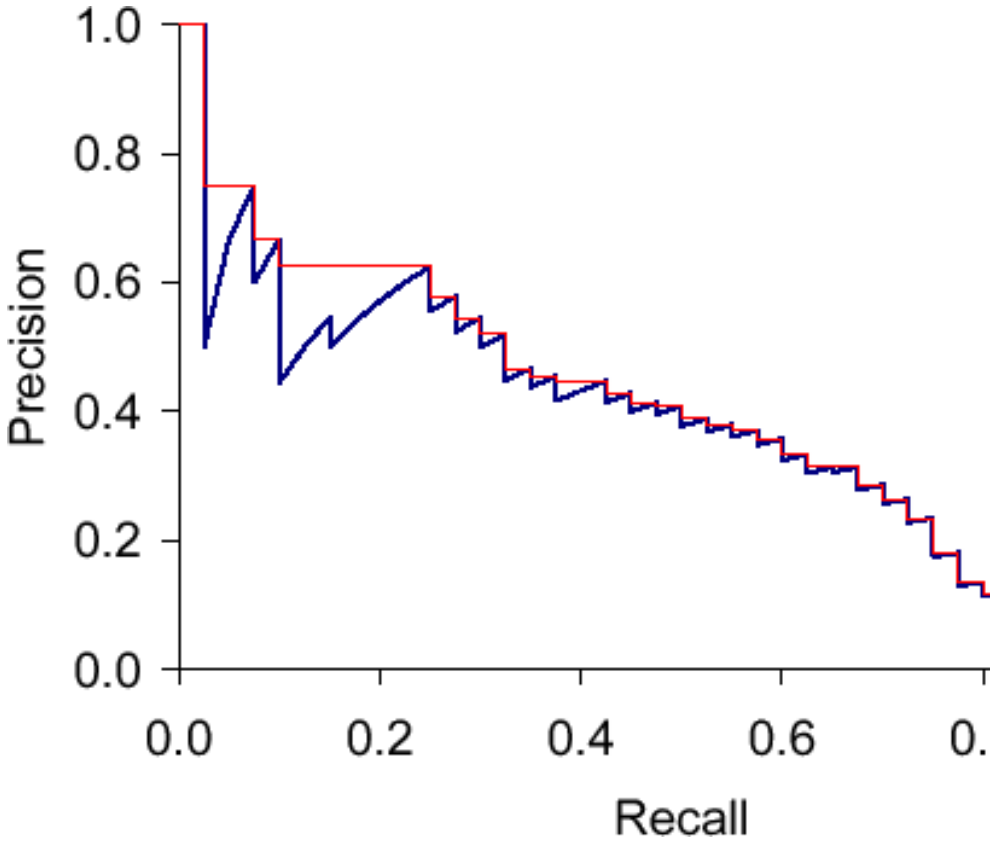


Non Maxima Suppression



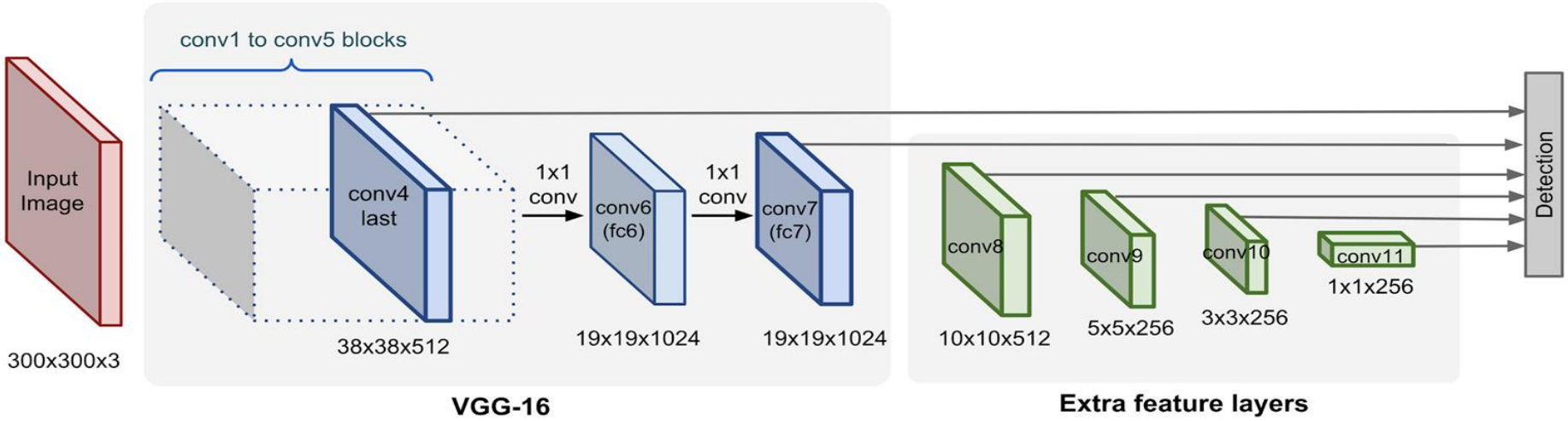
Object detection - measuring

- Greedy matching in order of confidence to ground truth boxes by threshold t
- Average Precision, area under Precision vs. Recall graph
  - Build by adding predictions from highest to lowest confidence
- mAP @ t - mean AP across all classes of dataset using matching threshold of t
- COCO AP - average across mAP for thresholds [0.5, 0.55.. 0.95]



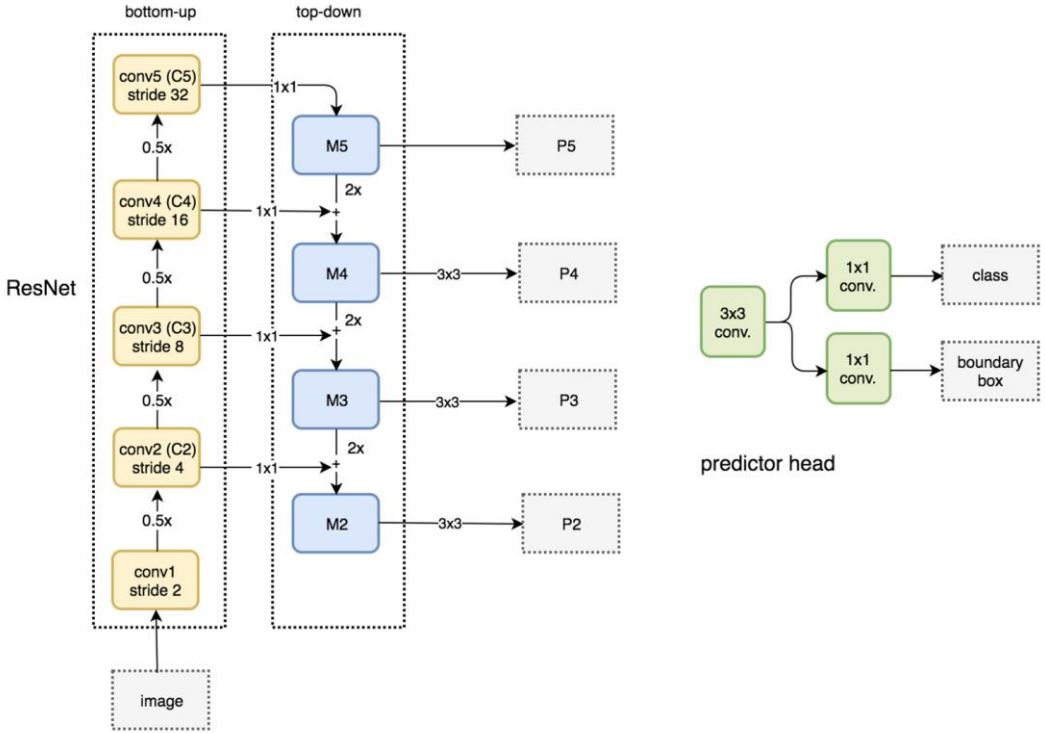
Object detectors, single shot

- Detect and localise objects in one pass
- Faster and simpler to train
- SSD and variants, RetinaNet, RefineDet, CentreNet



SSD: Single Shot MultiBox Detector - Liu 2015

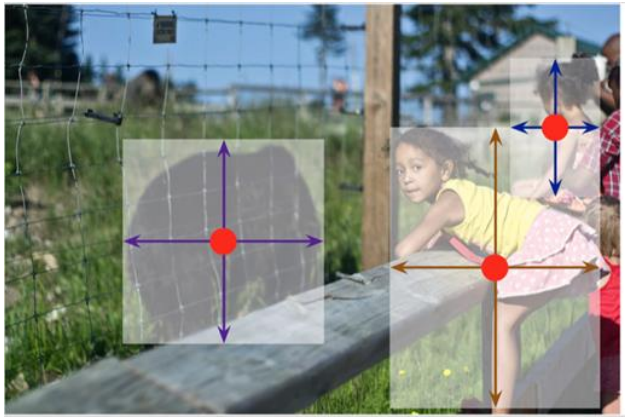
Object detection - single shot

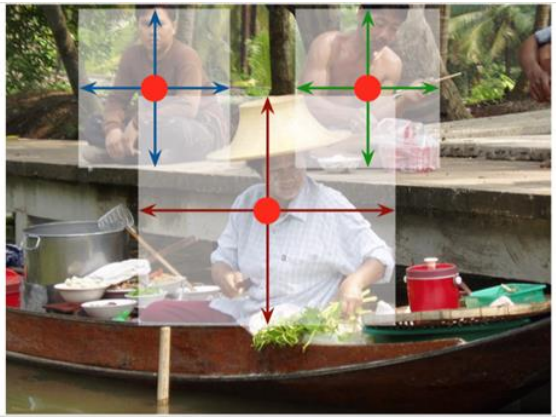


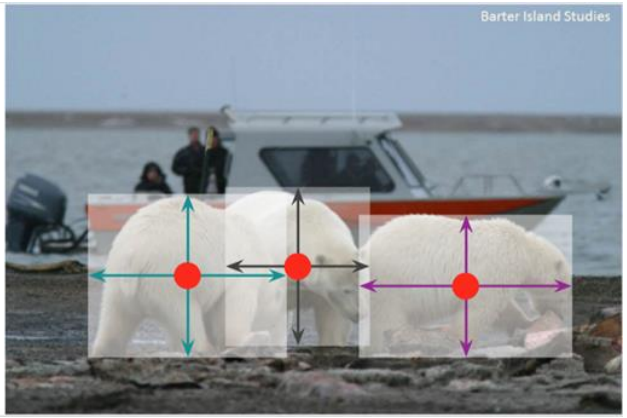
[Feature Pyramid Networks for Object Detection]

CenterNet

- Scrap anchor boxes
- Detect local maxima in feature maps
- Regression for x, y, width, height



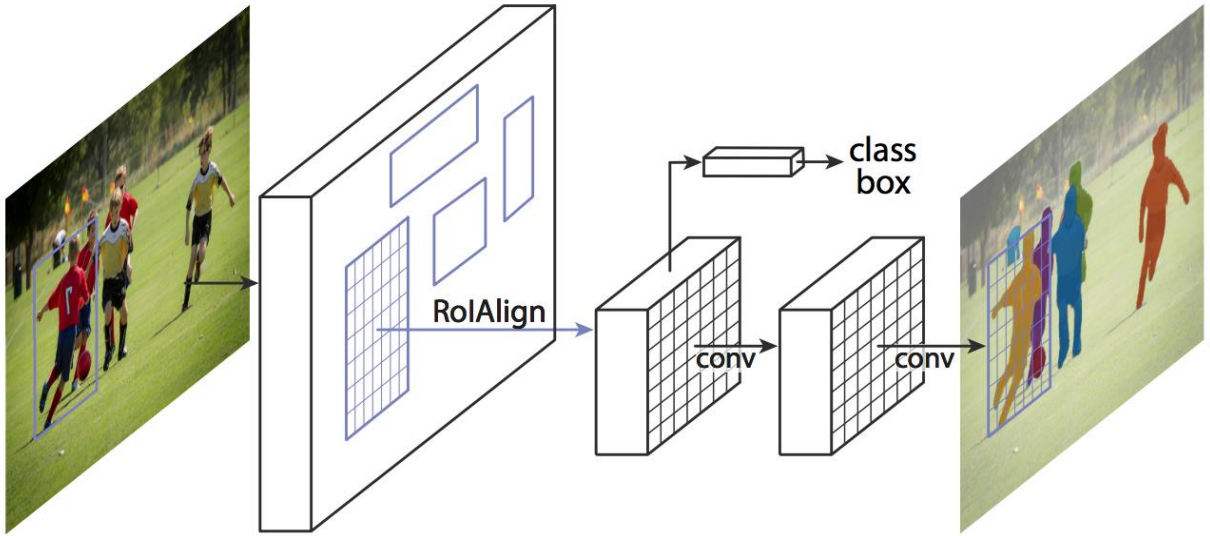




Objects as Points - Zhou et al. 2019

Two stage detector, Mask RCNN

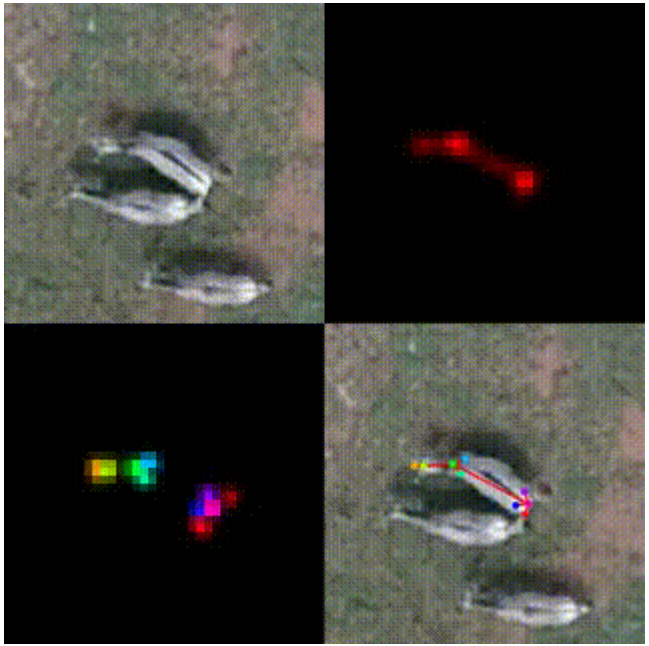
- Single stage detector first (calls this Region Proposal Network (RPN)
- For each detection, crop feature map (RoiAlign below) for refined class estimate
- Estimate mask segmentation as part of second stage

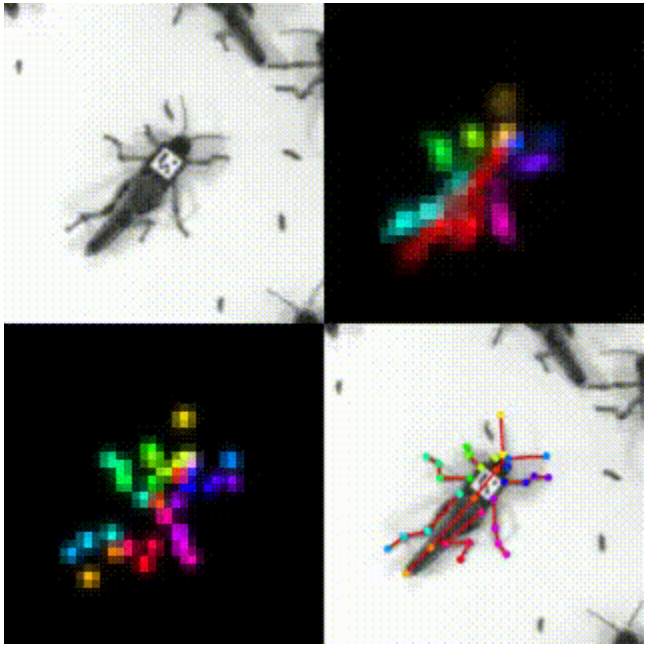


MaskRCNN - Kaiming et al 2017

Pose estimation

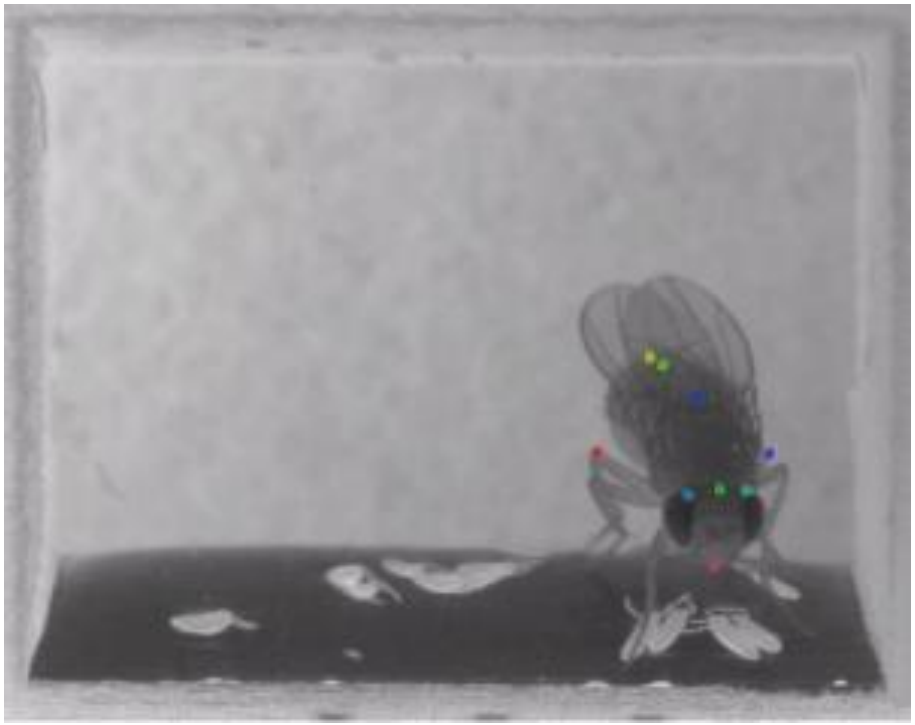
Not just human pose





Fast and robust animal pose estimation, Graving et al 2019

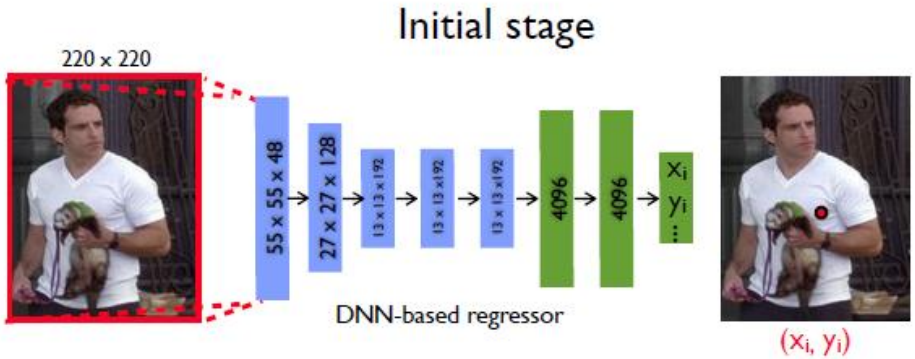
Not just human pose

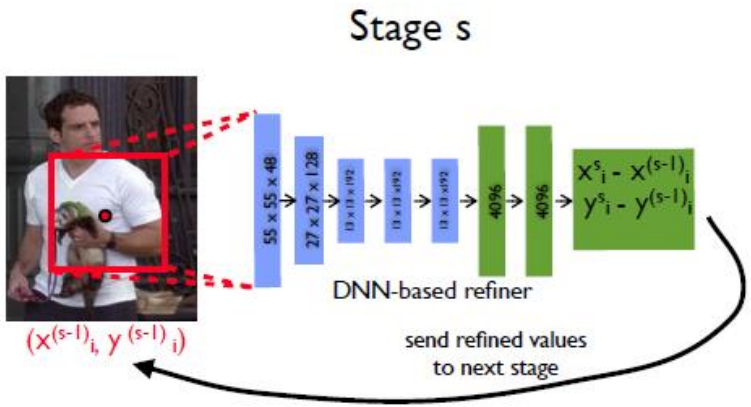


DeepLabCut - Markerless tracking of user-defined features with deep learning, Mathis et al. 2018

Approaches

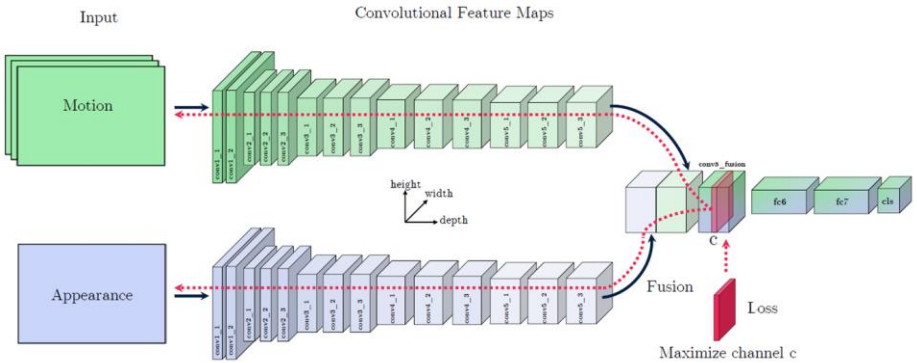
- Estimate points as regression of 2n vector
  - Fits with single shot object detector
  - Iterative refinement cascade
- Heatmaps for each keypoint
  - 2 stage per object
  - Can refine point regressions

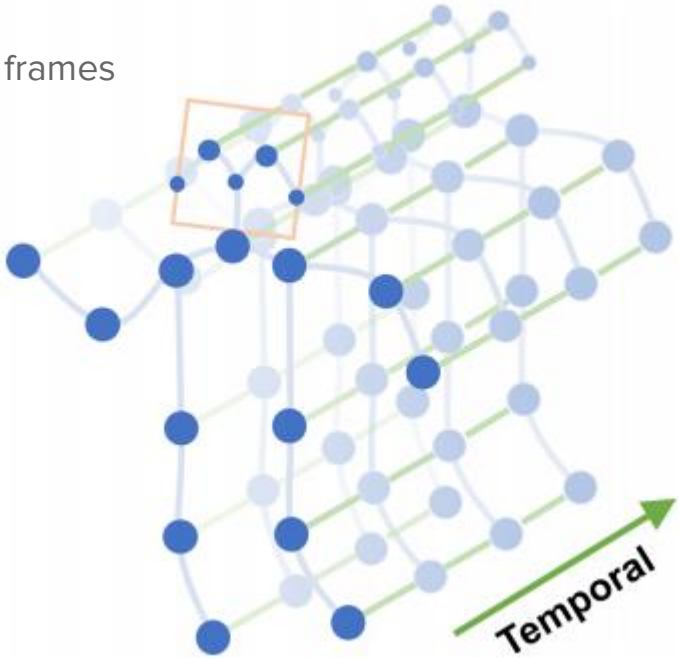




Action recognition

- Graph Convolutional Networks
  - Extract skeleton using pose recognition over frames
- 3D CNNs
- Two stream networks/Recurrent NNs





Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition, Yan et al. 2018

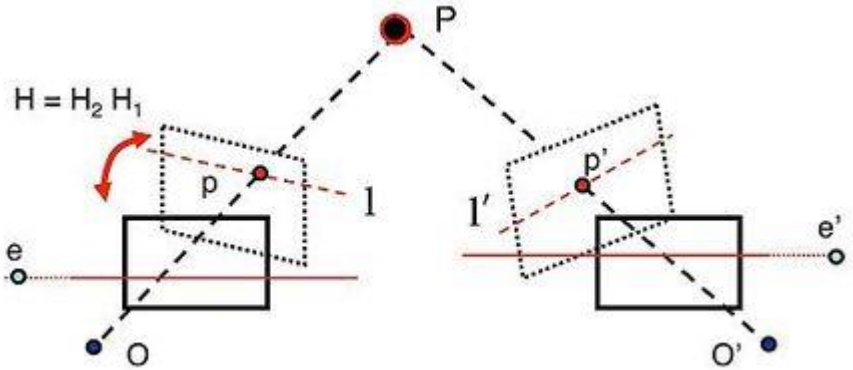
Image matching and correspondence

- Stereo matching
  - Rectified stereo
  - Multi--view stereo
- Image matching
  - Structure from motion
  - Sparse stereo
  - Tracking
- Optical flow
  - SLAM
- 3D model matching
  - 6DOF pose estimation
- 3D reconstruction
  - ‘implicit function’ representations

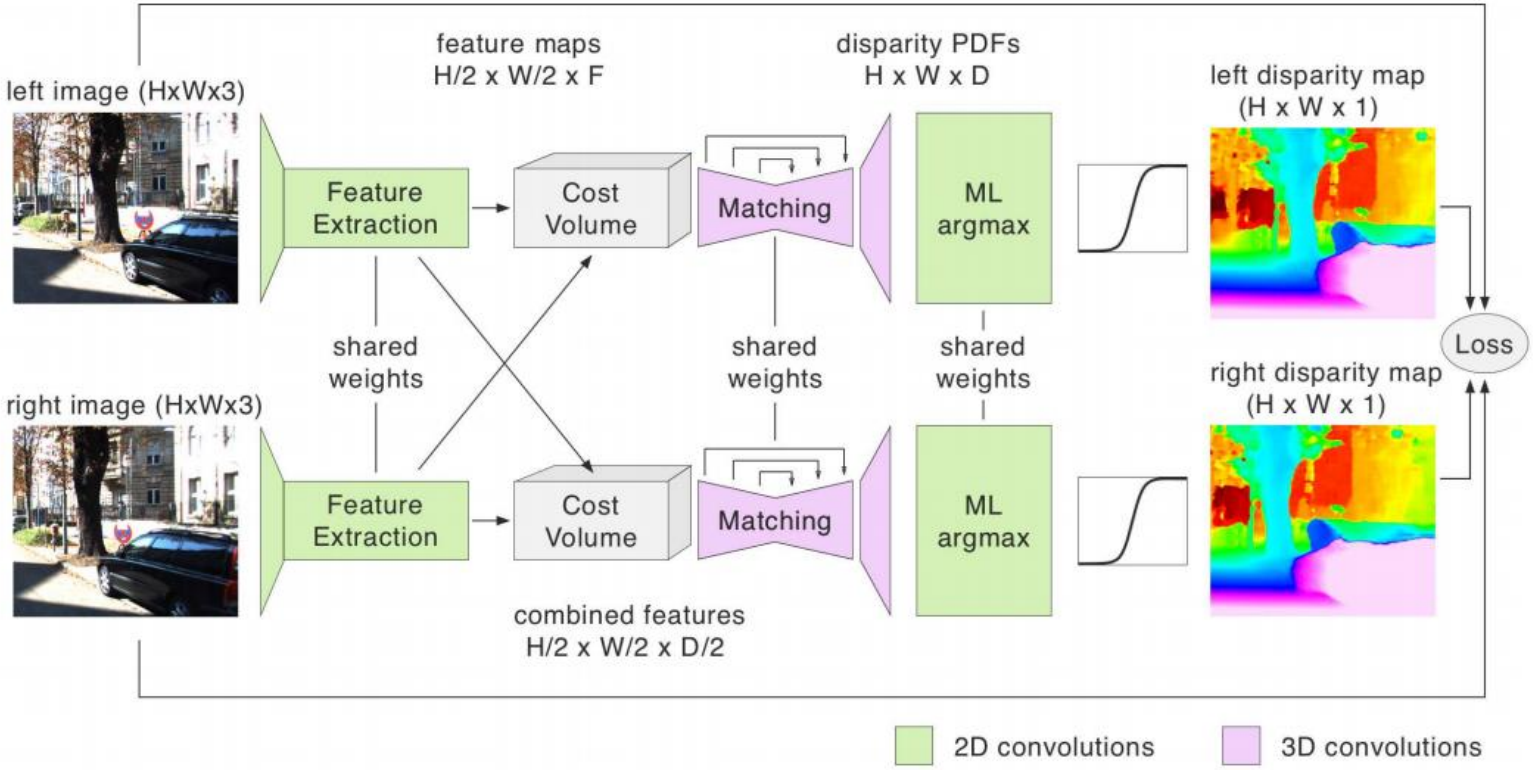
Stereo matching

- 1D search along epipolar lines (rectified)
- Multi-view extensions using differential warping
- Constrained to be piecewise smooth
  - Traditionally solved iteratively (SGBM) or globally (Graph cuts)
- Separate 2D networks encode features
- Matched using feature volumes (4d)

$$\hat{d} = \sum_{d=0}^{D_{max}} d \times \sigma(-c_d).$$

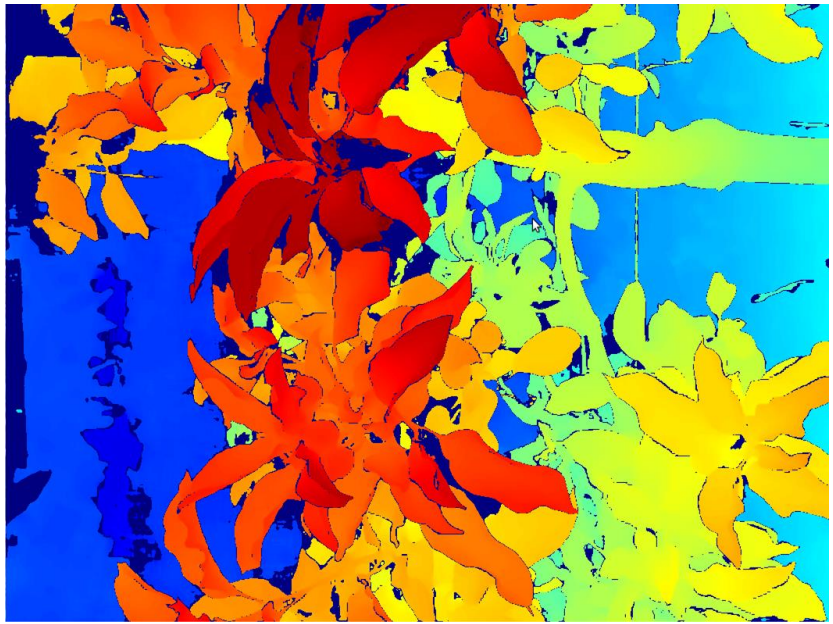


Stereo matching



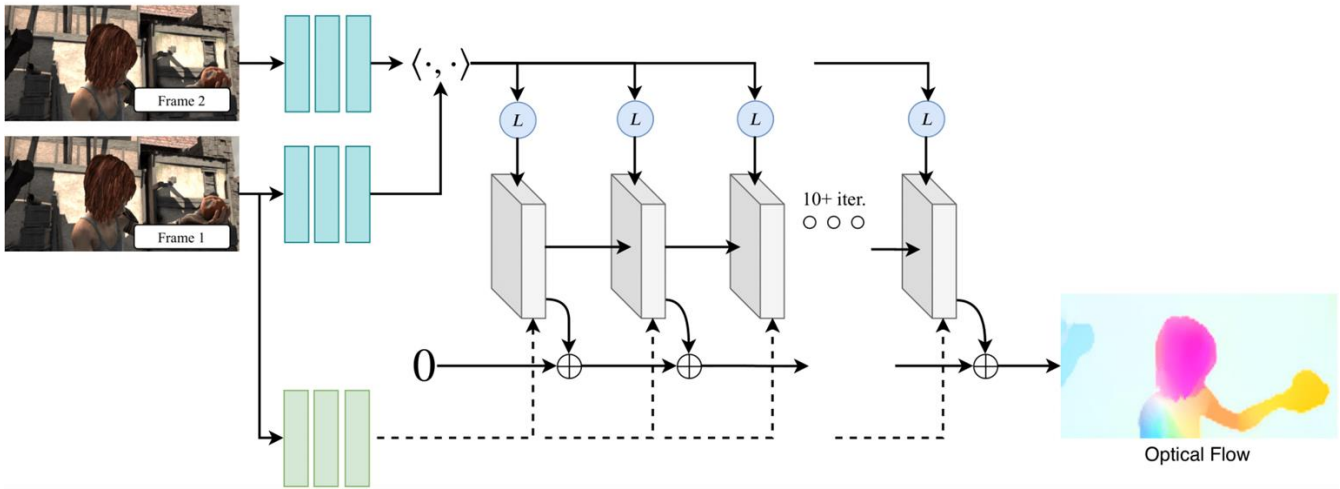
Stereo matching

- Hierarchical Deep Stereo Matching on High Resolution Images
  - https://github.com/gengshan-y/high-res-stereo



Optical flow

- Feature extraction per frame
- 2D correspondence search
- Mix of cost volume (5d) and sparse matching, iterative refinement and warping

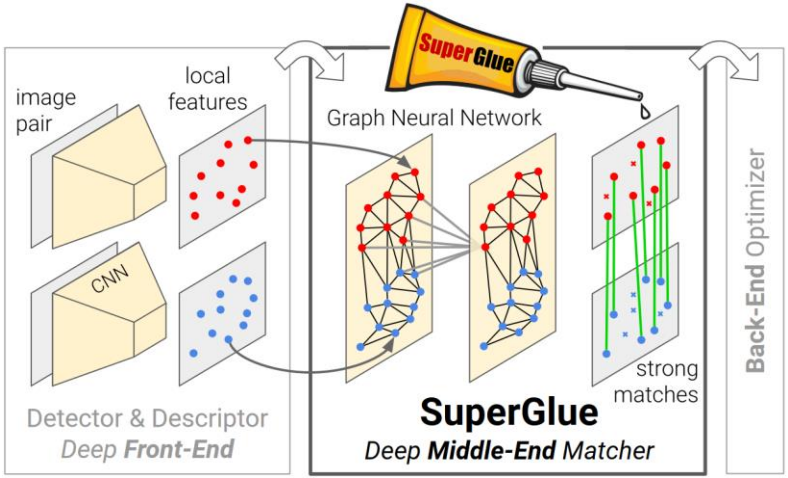


Optical flow

- Lite flownet - https://github.com/sniklaus/pytorch-liteflownet

Image feature matching

- Image matching
  - Find points of interest (detector)
  - Extract discriminative feature (descriptor)
  - Many options for matching - for example Graph NN or attention based
- Train using SLAM or SFM
  - Inliers are positives
  - Outliers negatives

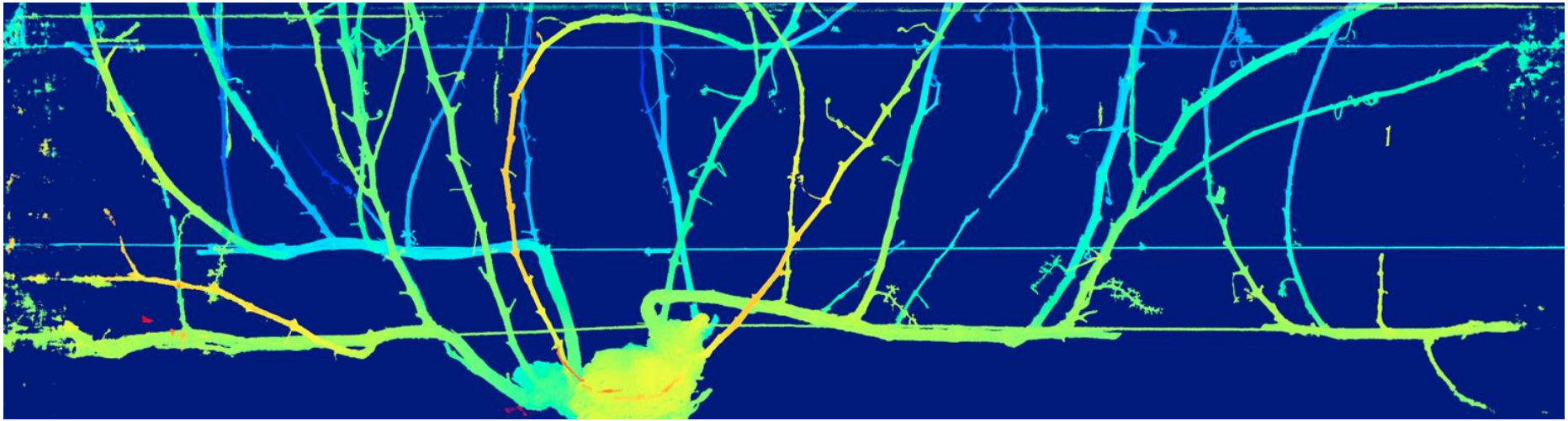


Volumetric reconstruction - NeRF

Flexible compression mechanism

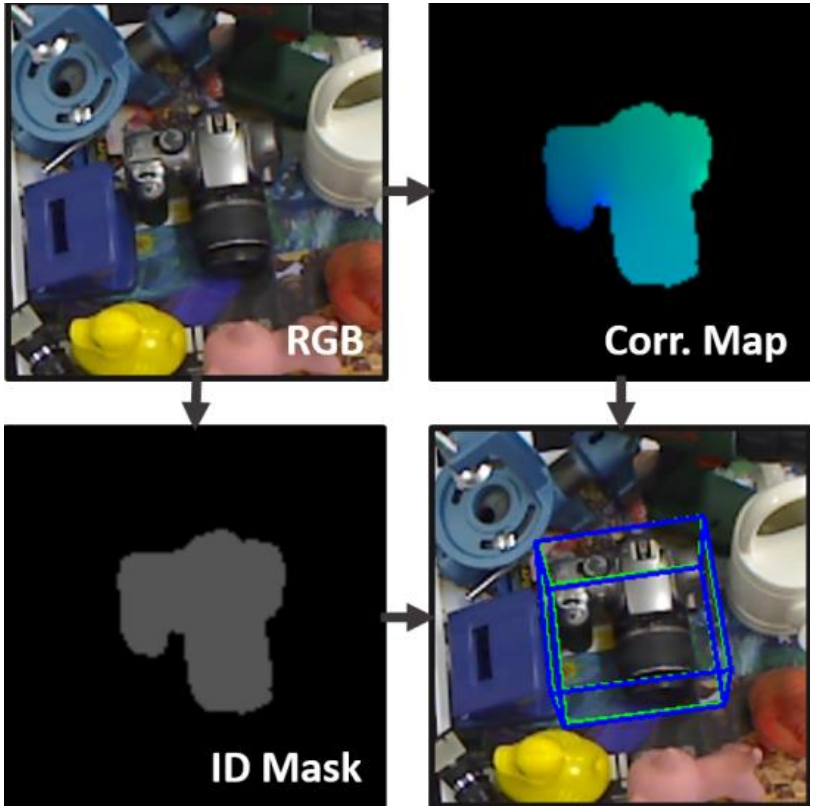
- 3D reconstruction with volumes uses a lot of memory - 3d for occupancy, and even worse 5d for representing radiance field for realistic view synthesis
- Idea - use a neural network (plain fully connected MLP) to encode the occupancy and learn the scene with volume raytracing
  - xyz ➔ occupancy
  - xyz, θφ ➔ color
- Problem - neural networks of this form have a natural bias towards smooth functions and cannot represent high frequency/discontinuity
- Solution - scale/encode the inputs as a fourier encoded sequence
  - sin(x), sin(2x), sin(4x)

Example



3D model matching (6DoF pose matching)

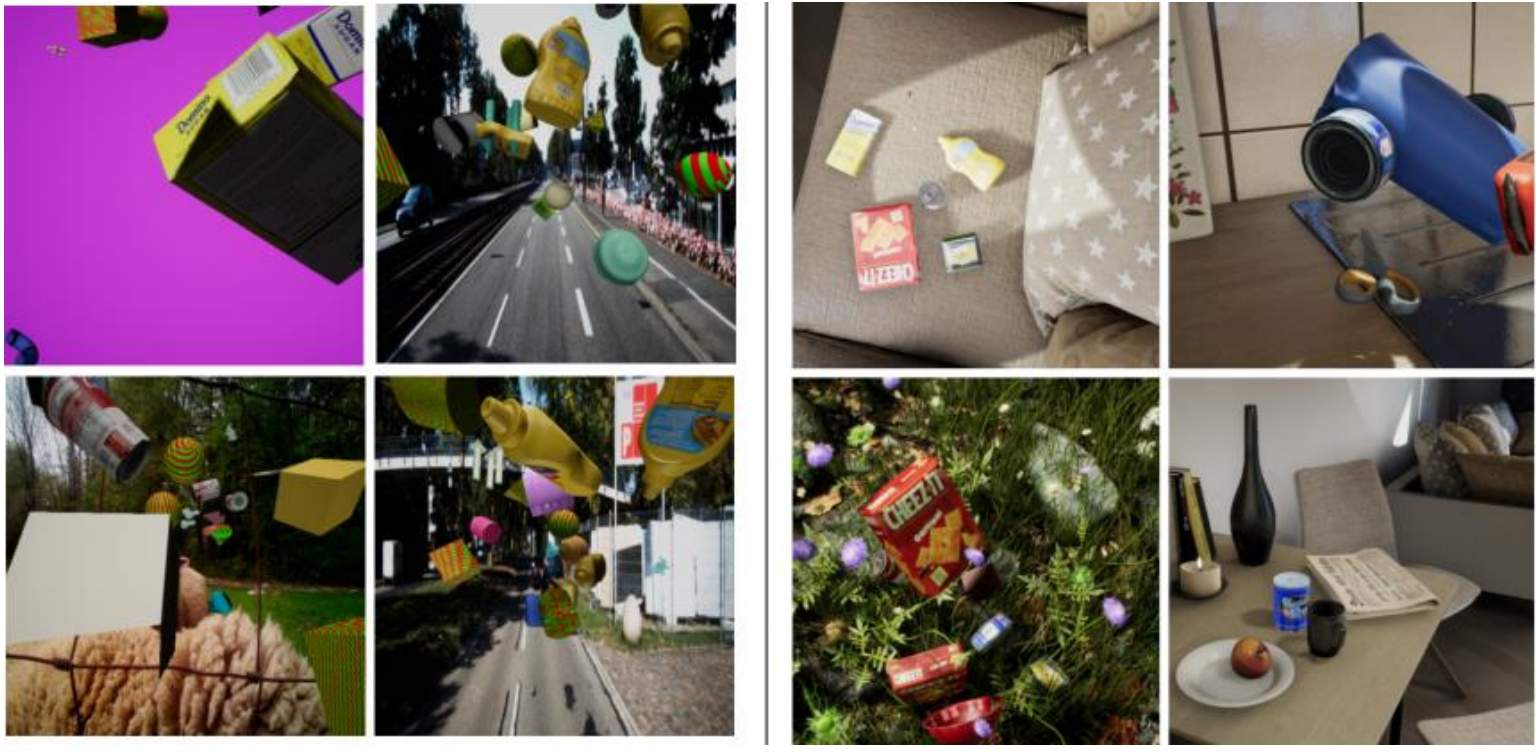
- 3D model available
- Estimating such a complex quantities for NN is difficult (leads to inaccurate result)
- Establish correspondences using CNN
  - Training data not easily available
  - ... but there is a 3d model
- Match to known model using geometry solver
- Refine using renderings and local search, or Iterative Closest Point if depth images are available



DPOD: 6D Pose Object Detector and Refiner

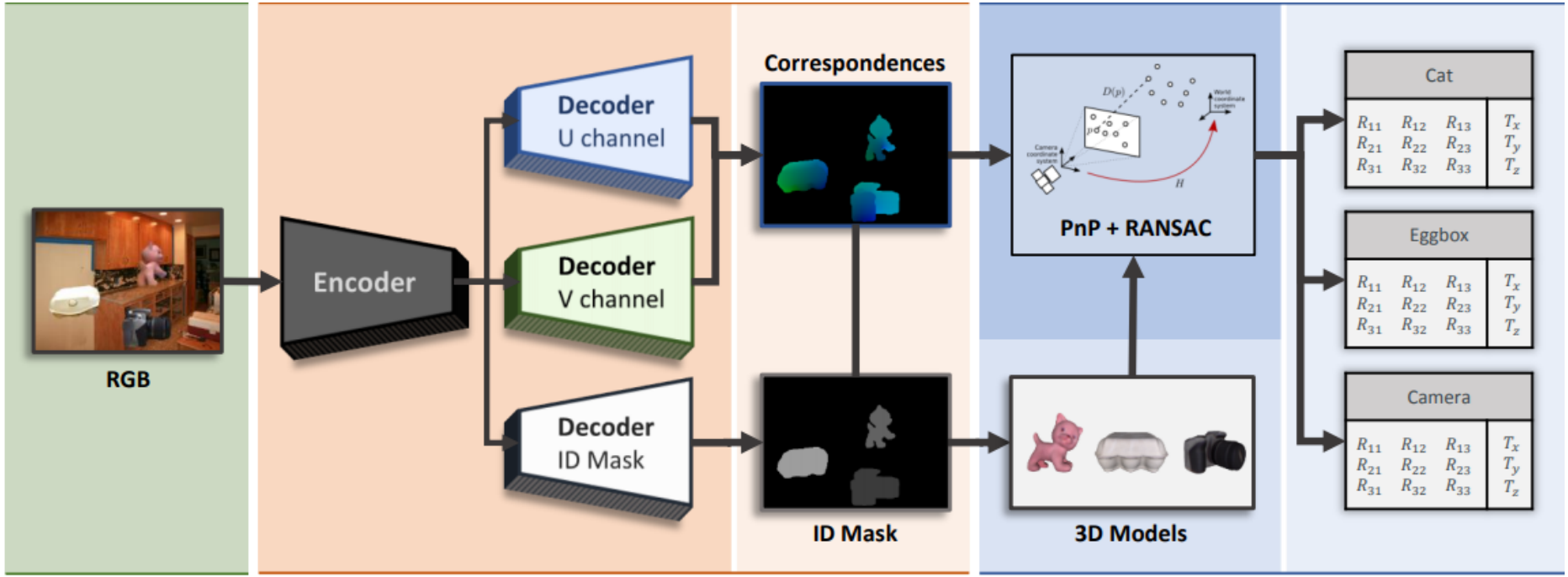
3D model matching

- Training on synthetic data (domain shift is often problematic!)



3D model matching - example

- Perspective and Point (PnP) to solve correspondences to model
  - Discard outliers using Randomised Sampling and Consensus (RANSAC)

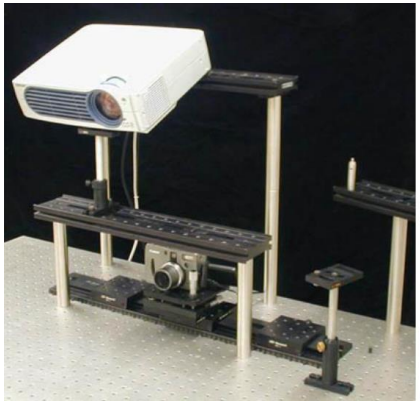


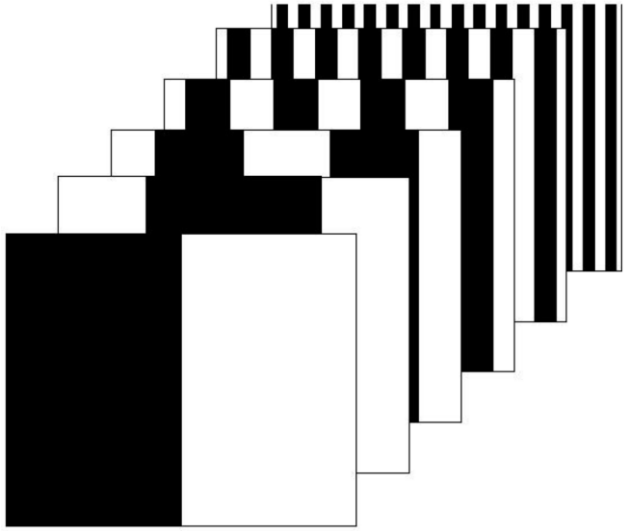
Training for image matching tasks

- Hand labelling is too much
  - Disparity maps need to be more accurate than segmentation masks
  - Need to record precise disparities per pixel, not just labels
- Synthetic data
  - Use computer graphics to render images with known properties
- Self supervised learning
  - Use known physical properties to provide supervision
- Pseudo ground-truths
  - Use device for high accuracy capture - for example LiDAR (Kitti) or Structured light (Middlebury)
  - Use more information - for example higher resolution or larger number of images
  - Use 3D reconstruction to generate higher quality models and train on these (modify images to make consistent)

Example Middlebury structured light

- Stereo correspondence between projector and cameras
- Flash sequence of gray coded binary (for robustness to errors)
- Decode binary code at position on image to position on projector and compute disparity



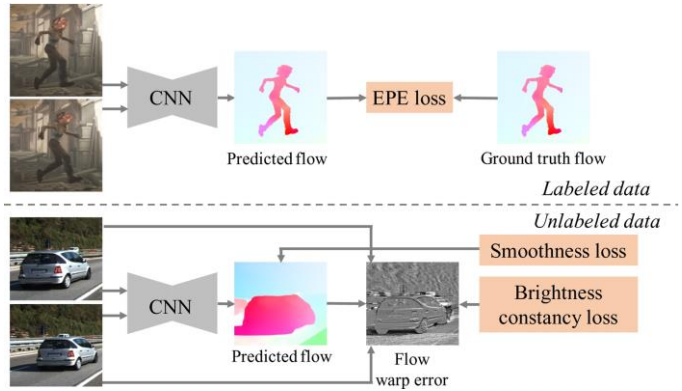


Training - synthetic data

- Inverse computer graphics
- Train on synthetic rendered models FlyingThings3D
- Refine on real scenes
  - Self supervised left-right consistency or Lidar captured KTTI

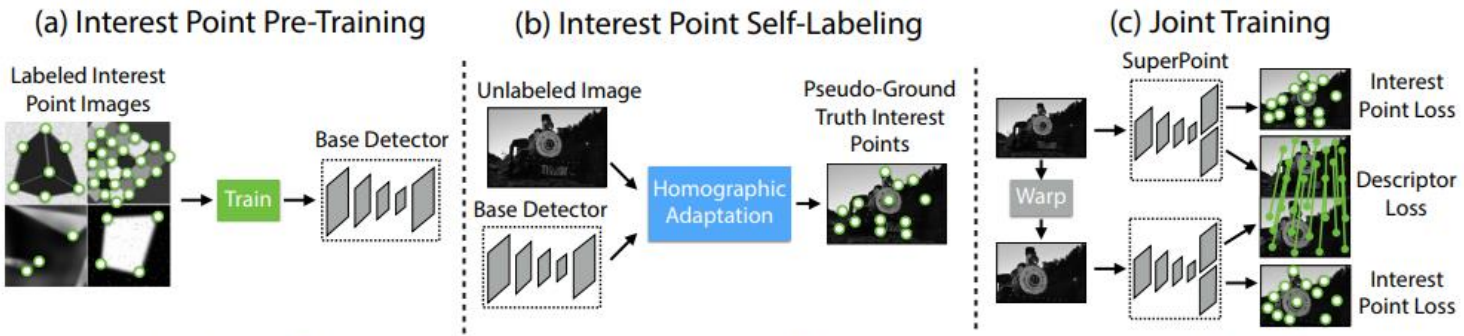
Training - self-supervised learning

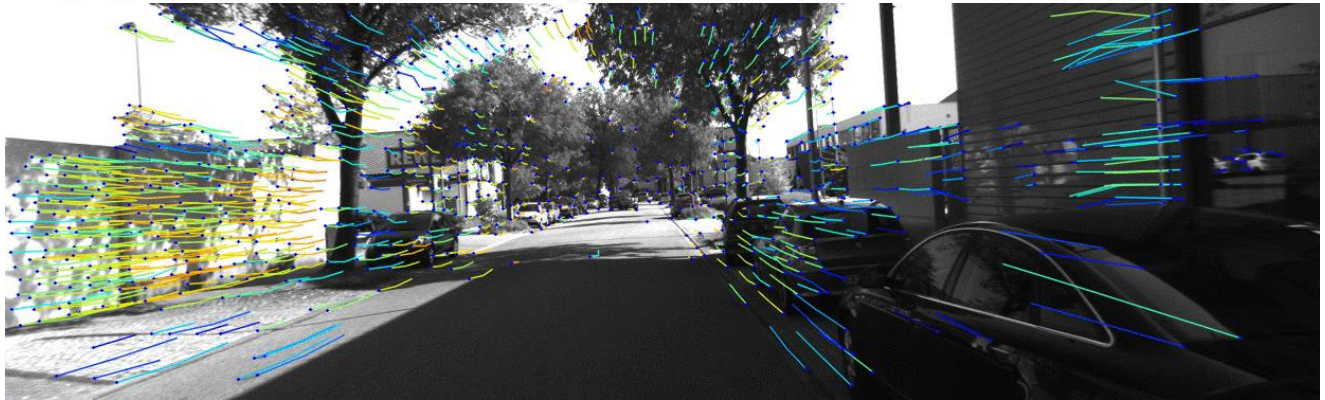
- Train on synthetic rendered scenes and refine
- Using consistency (left/right or inter-frame) using image warping
- Feature matching - reconstruct camera positions using Structure From Motion
  - Inliers become positive examples
  - Outliers become negative examples



Self supervised - key point detector

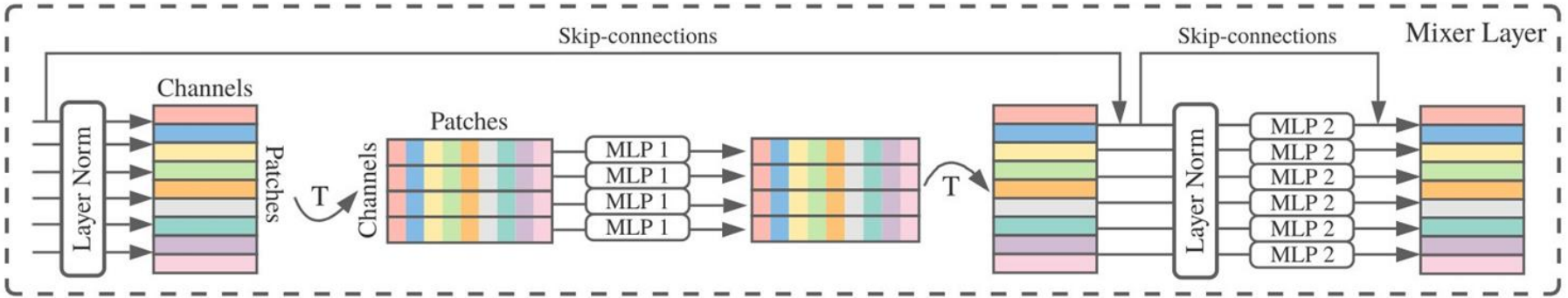
- Fills role of corner detector and descriptor - for example Harris/SIFT
  - Self-supervised by warp data augmentation





Recent directions in image NNs

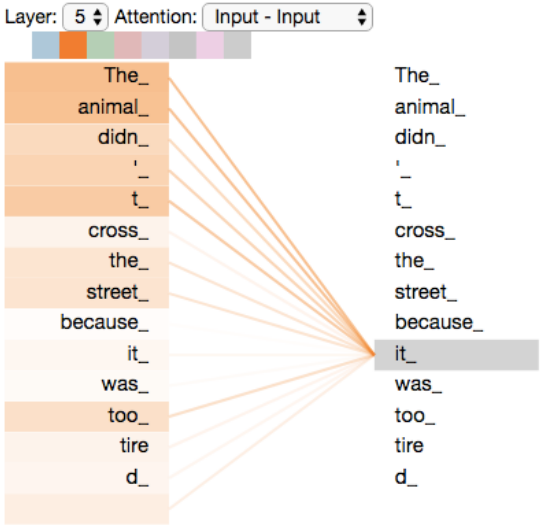
- Attention methods - transformers
  - Borrowed from natural language processing - GPT-3 etc.
  - Can effectively replace convolutions
  - ... however, not obviously ‘better’
- Very large scale unsupervised learning
  - Not very practical to train yourself outside google/facebook/open AI
  - However, can be used for fine tuning and/or used for inference (sometimes)



Attention mechanisms

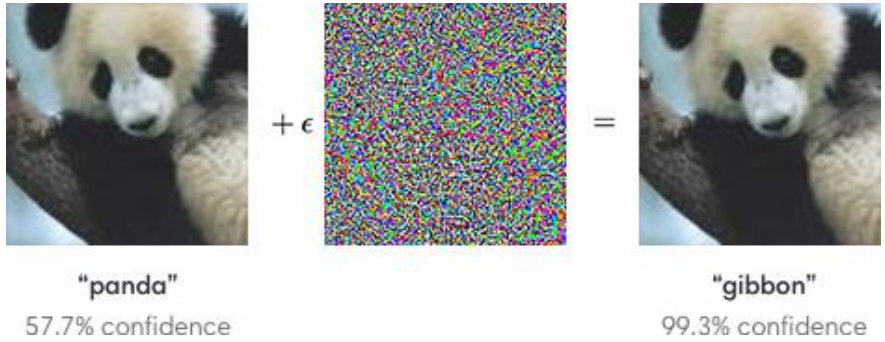
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- Forms association between two sets
  - Use positional encoding to encode sequences
  - Can be used to process sets more effectively
    - Object detectors without non maxima suppression
- O(n^2) in length of the set
  - For an image O(n^4) in NxN set of patches
- Form larger architectures with multiple layers
  - Transformer for processing natural language
  - Vision Transformers (ViT, DeIT) - images



Difficult problems

- Data hungry
  - Often need a lot of data to generalise well
  - Labelling data is laborious and takes a long time
- Domain shift causes problems
  - Out of Distribution examples
  - Not as general as you hope
- Adversarial attacks
- Training takes a long time (sometimes)
- Not easily interpreted
  - Some properties can be visualised, but in general it is a hard problem
- Hard work!
  - Large design space, a lot of of hyper-parameters, many things to go wrong
  - Complex to debug and understand
  - Many people get carried away with what is possible (yet)



Problem solving - overfitting

- Cause: insufficient data for generalisation
- Symptoms: Low training loss, but also low accuracy on validation/testing
- Potential solutions
  - More data - needs labelling and time consuming (but sometimes the only way)
  - Image augmentation - usually the first port of call, but no magic bullet
  - Synthetic data/generated images
  - Self supervised learning
  - Use a lightweight, faster neural network with less parameters

Problem solving - answering questions

- How many images are required? Which model is best?
- Solution: validation
- No easy shortcuts
  - Split images into validation and training
  - Train under different parameters (or different numbers of images)
  - Test against validation set (did results improve?)
- Issue... limited data for validation, testing
  - Use N-fold validation, use many small splits of the dataset to utilise all images for training (and validation)

Problem: limited labelling

- Data augmentation
  - Randomise data in ways which do not change the label
- Semi-supervised learning
  - Some labelled data, larger amount of unlabeled data
- Weakly supervised
  - Labels are weaker than required for the whole task
  - Example: Image level labels for object detection
- Self-supervised learning (aka. Unsupervised learning)
  - Use known properties of the data to provide supervision
- Human in the loop/interactive machine learning
  - Aims to make the best use of human annotator
  - Reduce cognitive load
  - Active learning - label the most informative/uncertain examples first
  - Validation based annotation

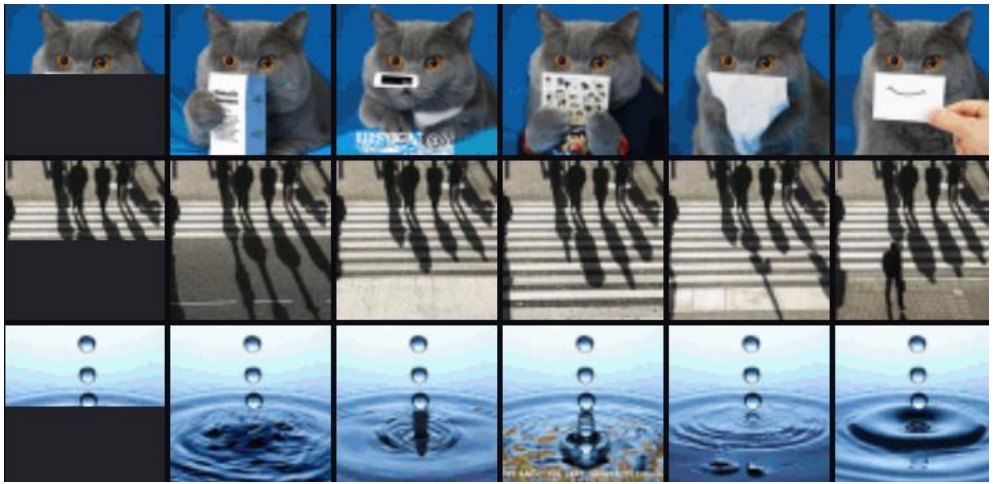
Data augmentation

- Apply transformations which are invariant/equivariant to the labels
- Randomly perturb data
  - Geometric - translation, scale, rotation
  - Photometric - brightness, contrast, hue
  - Mixing - mix images and labels
  - Add noise - gaussian, salt and pepper, synthetic rain, cutout
- ‘Infinite’ size training set
  - Practical limits - at some point need more real data

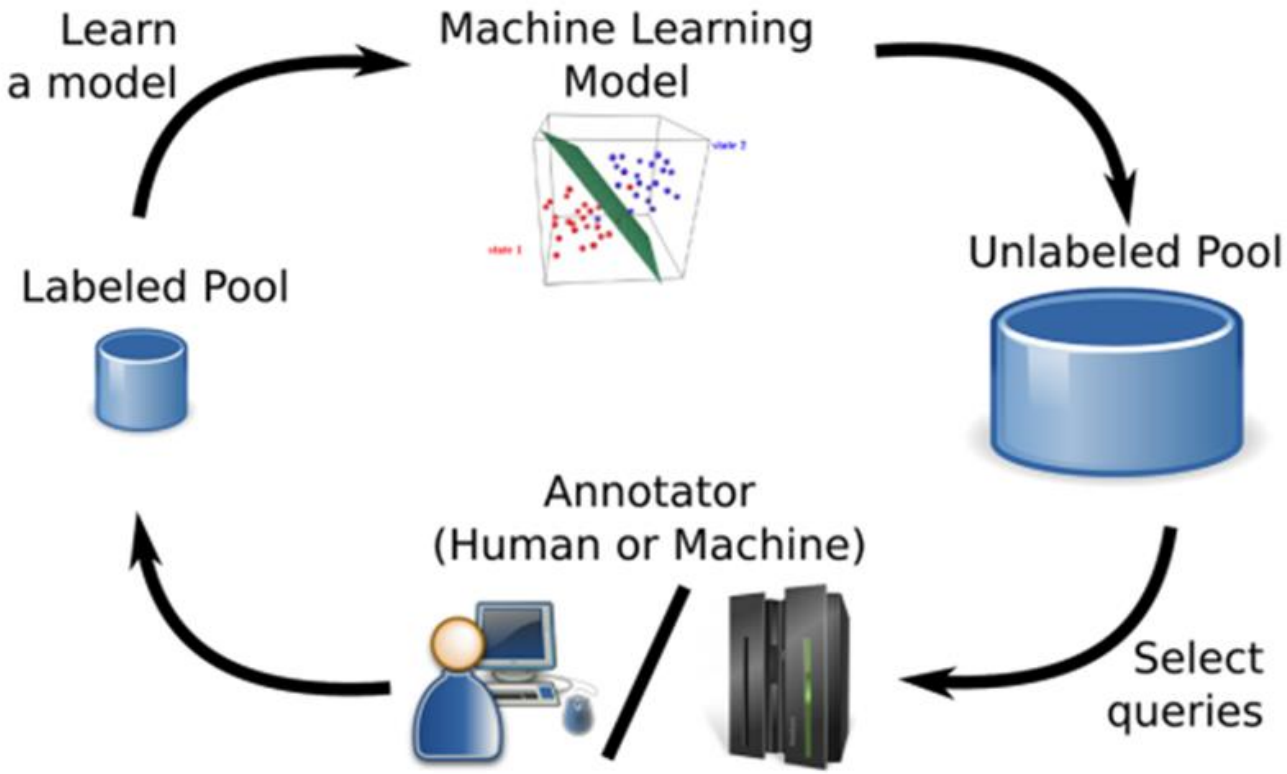



Un/self-supervised examples

- ‘Traditional’ unsupervised learning
  - Autoencoder, Clustering, Dimensionality reduction
- Auxillary task
  - Image completion - iGPT-2 - predict the next pixels
  - Image in-painting
- Knowledge distillation
  - Train large network on large dataset
  - Distill into smaller faster model
- Contrastive loss
  - Aim to make descriptor more unique
- Generative data/style transfer
  - GAN for data augmentation
- Reinforcement learning
  - Self play in games - e.g. alpha-zero

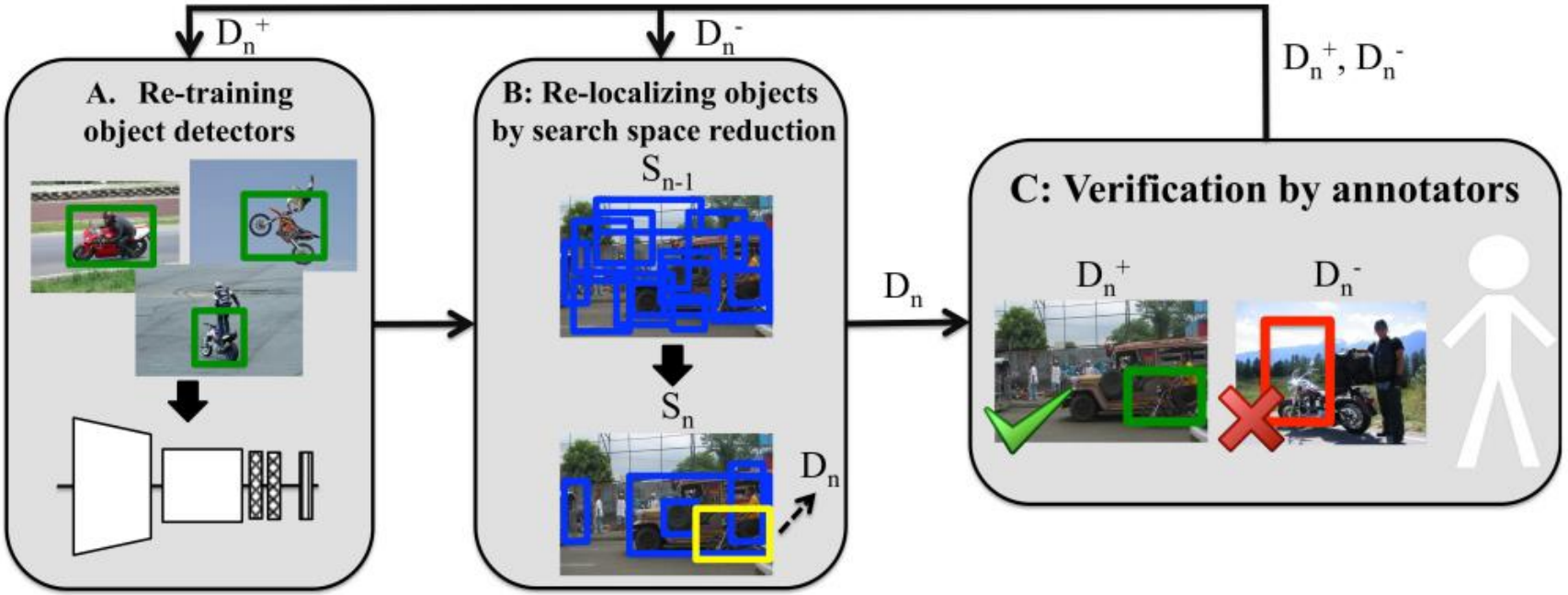


Active learning



Verification based annotation

- Question answering
  - Upgrade weak detector to strong detector with yes/no questions

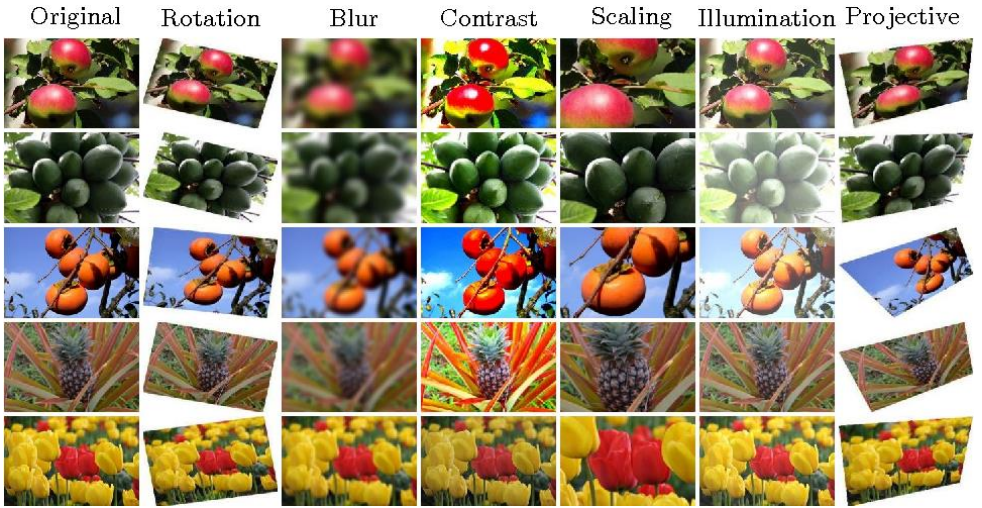


Resources - object detection/instance seg

- Detectron2
  - https://github.com/facebookresearch/detectron2
- Maskrcnn-benchmark
  - https://github.com/facebookresearch/maskrcnn-benchmark
- MMDet
  - https://github.com/open-mmlab/mmdetection
- Pytorch tutorial
  - https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

Resources - Image augmentation

- Libraries
  - Albumentations - https://github.com/albumentations-team/albumentations
  - BBAug - https://github.com/harpalsahota/bbaug
  - Torchvision (part of pytorch)
- Considerations
  - Many different types
  - Too much may be harmful
  - Not necessarily a silver bullet



Pose recognition

- Human pose
  - Alpha-pose
    - https://github.com/facebookresearch/detectron2
  - OpenPose
    - https://github.com/CMU-Perceptual-Computing-Lab/openpose
  - Detectron2 (see previous)
- Custom keypoints (any kind of pose)
  - DeepPoseKit
    - https://github.com/jgraving/deepposekit
  - Anipose
    - https://anipose.readthedocs.io/en/latest/

Resources - Newer directions

- Implicit function representations
  - Represent a scene in the neural network weights
  - NeRF - Neural radiance fields
    - https://github.com/bmild/nerf
- Differentiable algorithms
  - Kornia - https://torchgeometry.readthedocs.io/en/latest/
  - ‘Declaritive’ neural networks - https://github.com/anucvml/ddn
- Graph neural networks
  - https://pytorch-geometric.readthedocs.io/en/latest/
- Declarative neural networks
  - https://github.com/anucvml/ddn
- Volumetric and sparse 3D Point cloud neural networks
  - Minkowski Engine - https://github.com/StanfordVL/MinkowskiEngine
  - Torch points3d - https://github.com/nicolas-chaulet/torch-points3d

Questions?

Image manipulation

- Generative networks - GAN
- Use two competing (adversarial) networks.
  - Generator produces images from noise.
  - Discriminator distinguishes generated (fake) images from real ones
- Synthesise images and manipulate them in ways not possible before

Image manipulation

- Image completion
- Domain transfer - e.g. clean up underwater image
- Learn stereo correspondence from 3D movies
- Image super resolution