COMP9517: Computer Vision

Deep Learning

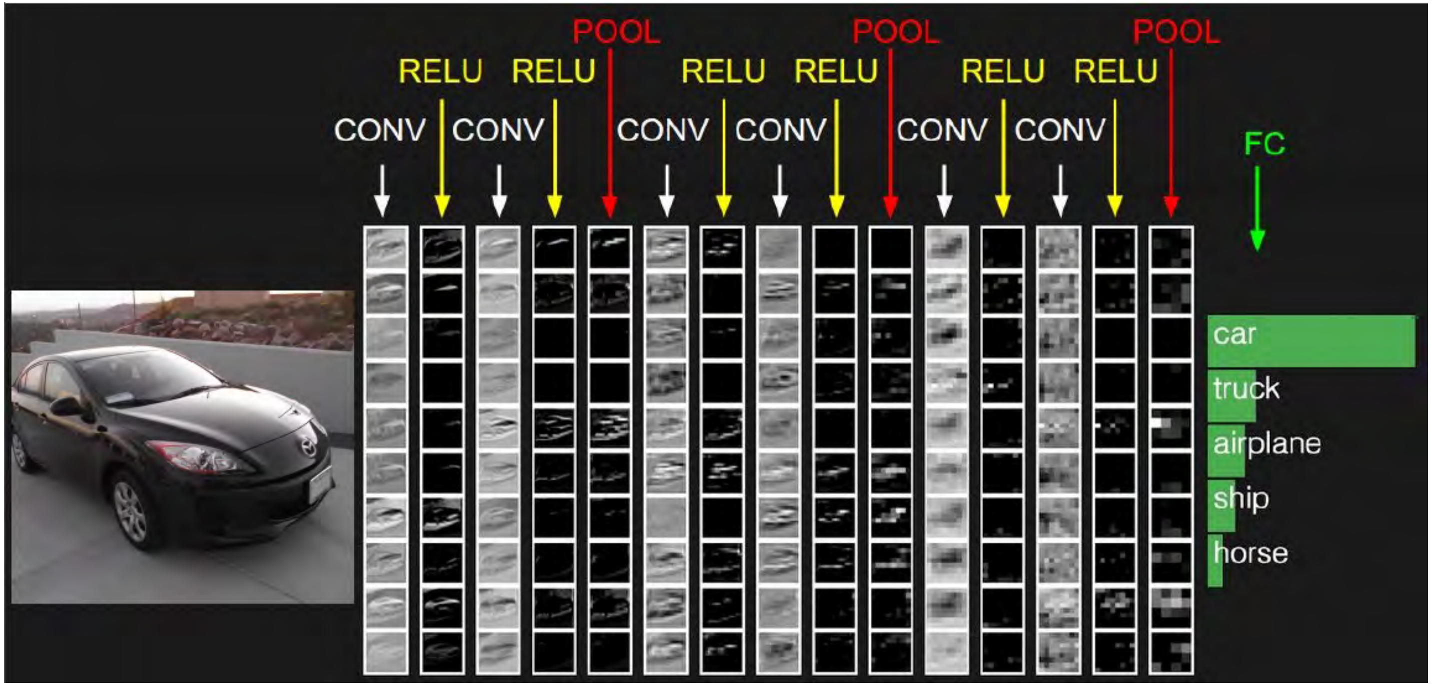
Part 2

Dong Gong

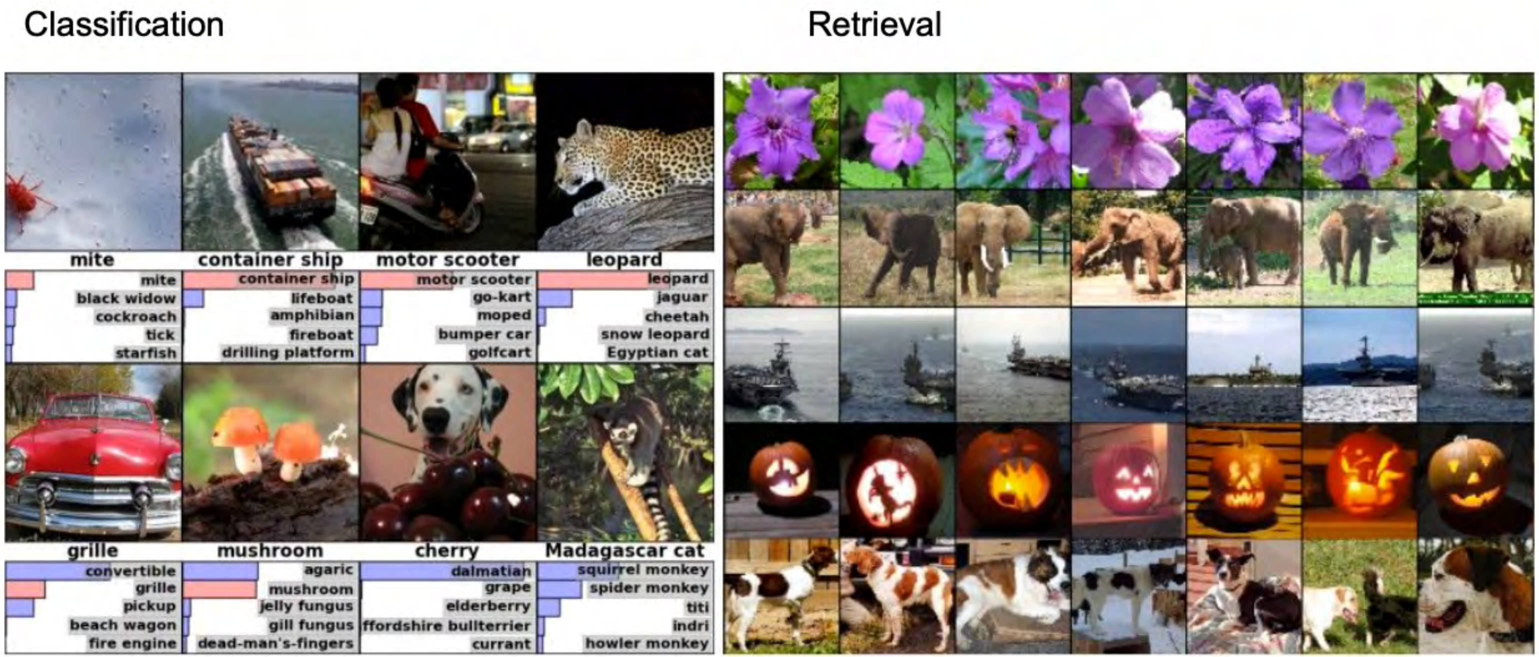
https://donggong1.github.io/
School of CSE, UNSW

Some slides are from Fei-Fei Li et al. (CS231n).

Recap: Convolutional Neural Network

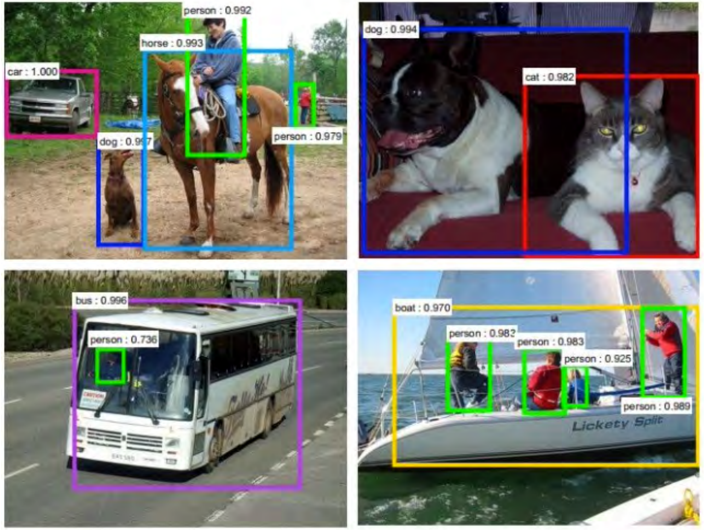


Recap: Applications of Deep Learning in Computer Vision



Recap: Applications of Deep Learning in Computer Vision

Detection



Figures copyright Shaoqing Ren, Kaiming He, Ross Girschick, Jian Sun, 2015. Reproduced with permission.

[Faster R-CNN: Ren, He, Girshick, Sun 2015]

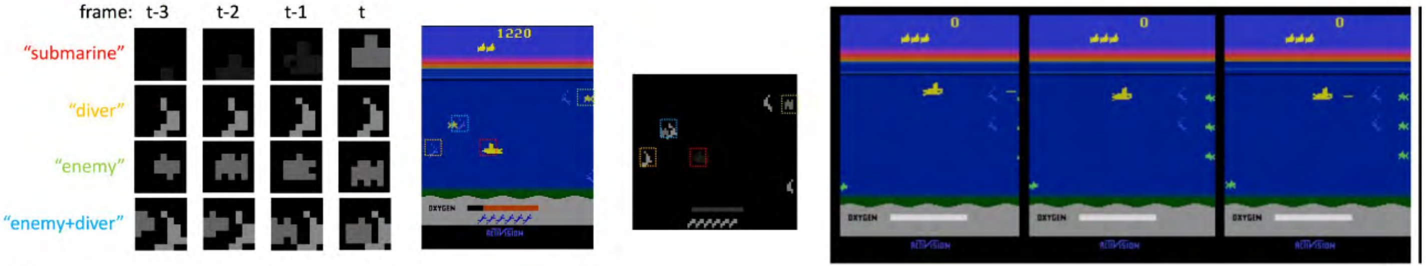
Segmentation

Figures copyright Clement Farabet, 2012.
Reproduced with permission.

[Farabet et al., 2012]

Recap: Applications of Deep Learning in Computer Vision

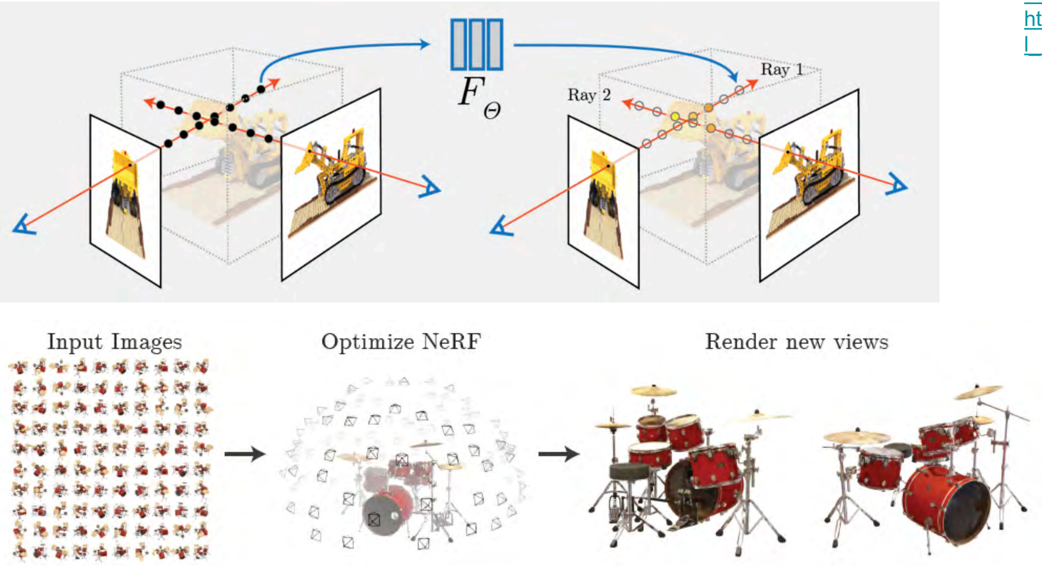
[Toshev, Szegedy 2014]



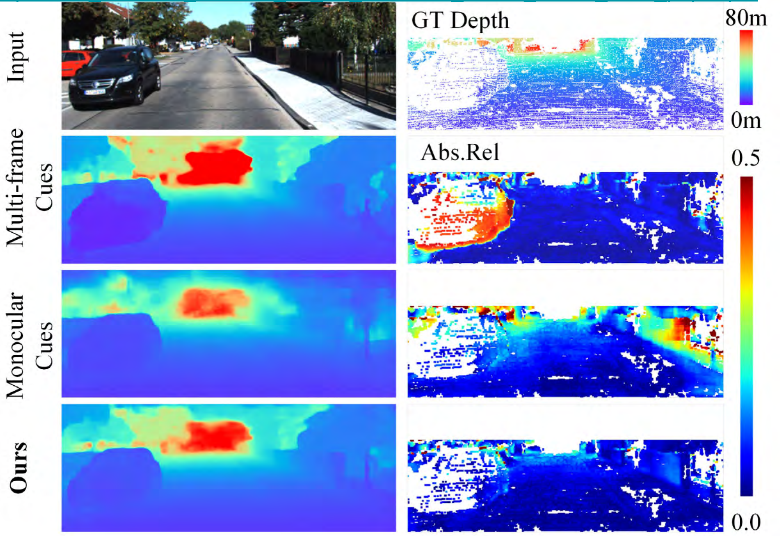
[Guo et al. 2014]

Recap: Applications of Deep Le

3D vision understanding
https://arxiv.org/pdf/2001.01349.pdf
http://openaccess.thecvf.com/content_CVPR_2019/papers/Li_RGBD_Based_Dimensional_Decomposition_Residual_Network_for_3D_Semantic_Scene_CVPR_2019_paper.pdf



Neural Radiance Fields (NeRF) for 3D vision
https://www.matthewtancik.com/nerf



Deep learning for depth estimation
https://ruili3.github.io/dymultidepth/index.html

Recap: Applications of Deep Learning in Computer Vision

self-driving cars

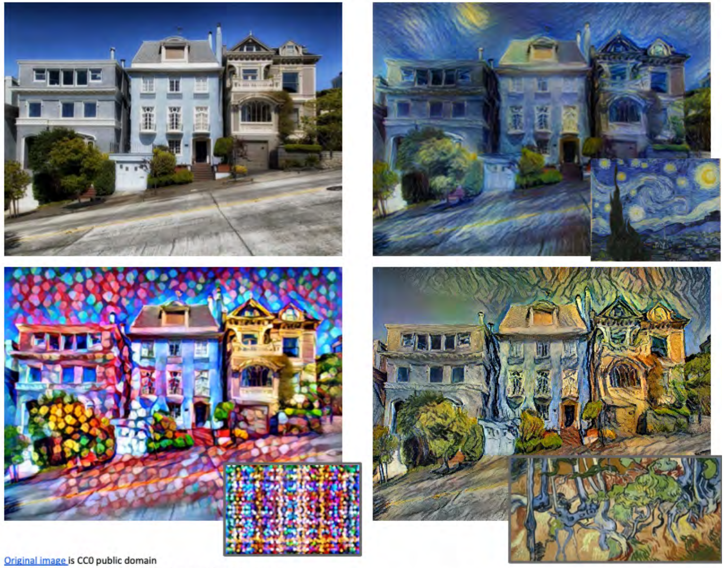
Photo by Lane McIntosh. Copyright CS231n 2017.

This image by Christin Khan is in the public domain and originally came from the U.S. NOAA.

Whale recognition, Kaggle Challenge

Photo and figure by Lane McIntosh; not actual example from Mnih and Hinton, 2010 paper.

Mnih and Hinton, 2010



Original image is CC0 public domain
Starry Night and Tree Roots by Van Gogh are in the public domain
Bokeh image is in the public domain
Stylized images copyright Justin Johnson, 2017;

Gatys et al, "Image Style Transfer using Convolutional Neural Networks", CVPR 2016
Gatys et al, "Controlling Perceptual Factors in Neural Style Transfer", CVPR 2017

https://github.com/donggong1/learn-optimizer-rgdn
https://donggong1.github.io/blur2mflow.html

Recap: Applications of Deep Learning in Computer Vision

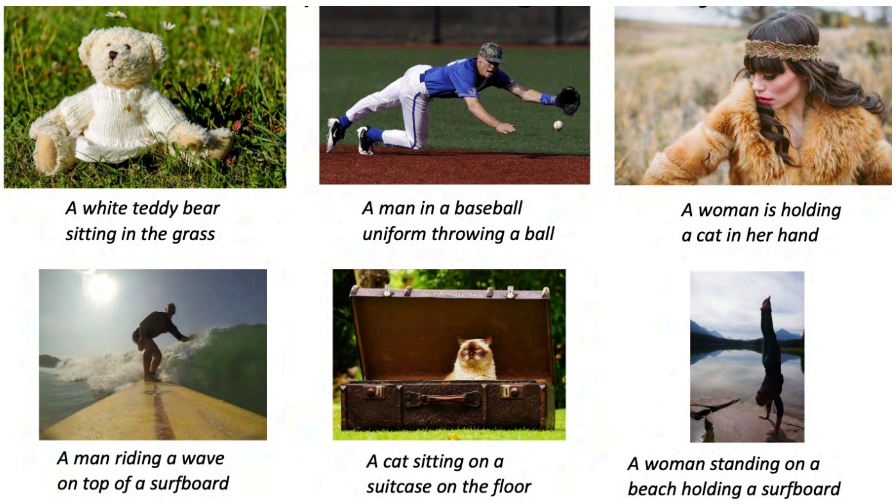
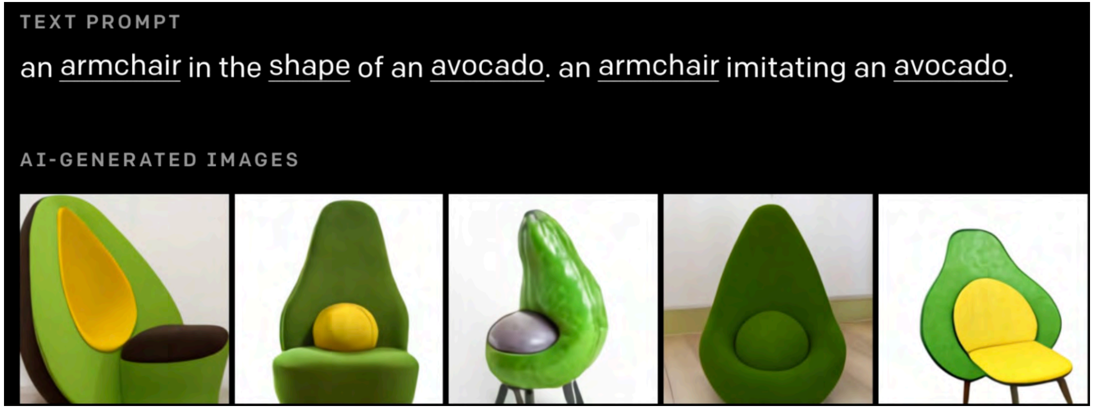


Image Captioning. Vinyals et al, 2015 Karpathy and Fei-Fei, 2015

Vision question answering (VQA)

Recap: Applications of Deep Learning in Computer Vision

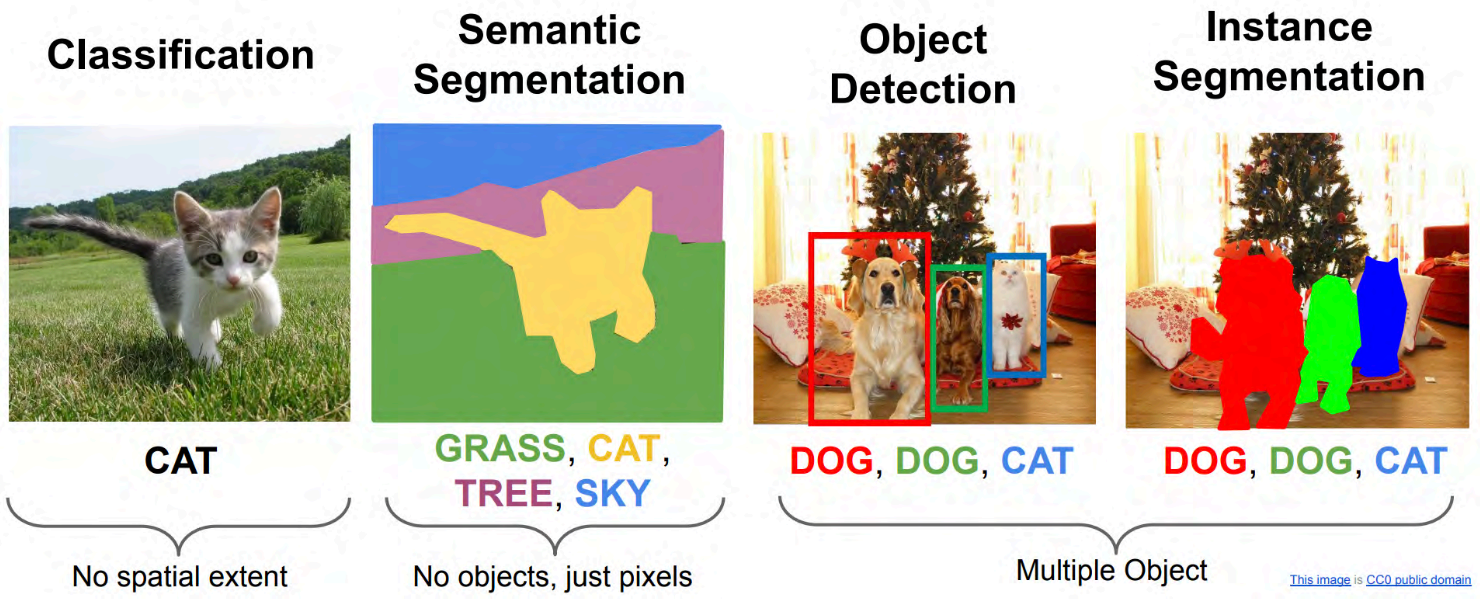
“A raccoon astronaut with the cosmos reflecting on the glass of his helmet dreaming of the stars”



Ramesh et al, “DALL·E: Creating Images from Text”, 2021. https://openai.com/blog/dall-e/

Image generation from text

Vision Tasks Beyond Classification



How should we design deep neural networks for different tasks?

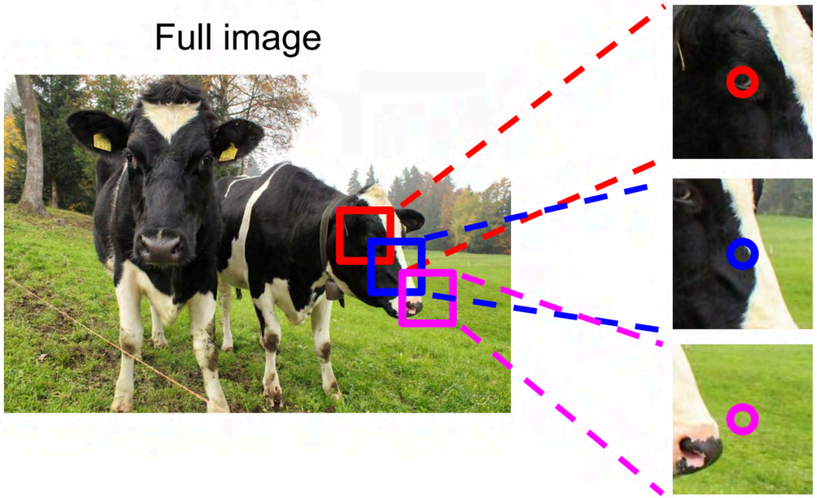
Semantic Segmentation

- Input: images (RGB images)
- Output: Class labels for all pixels
- Paired training data: each pixel is label with a semantic class category.
- Pixel-wise classification/semantic labelling

Image from COCO dataset - Microsoft COCO: Common Objects in Context, Lin et al, 2014

Semantic Segmentation

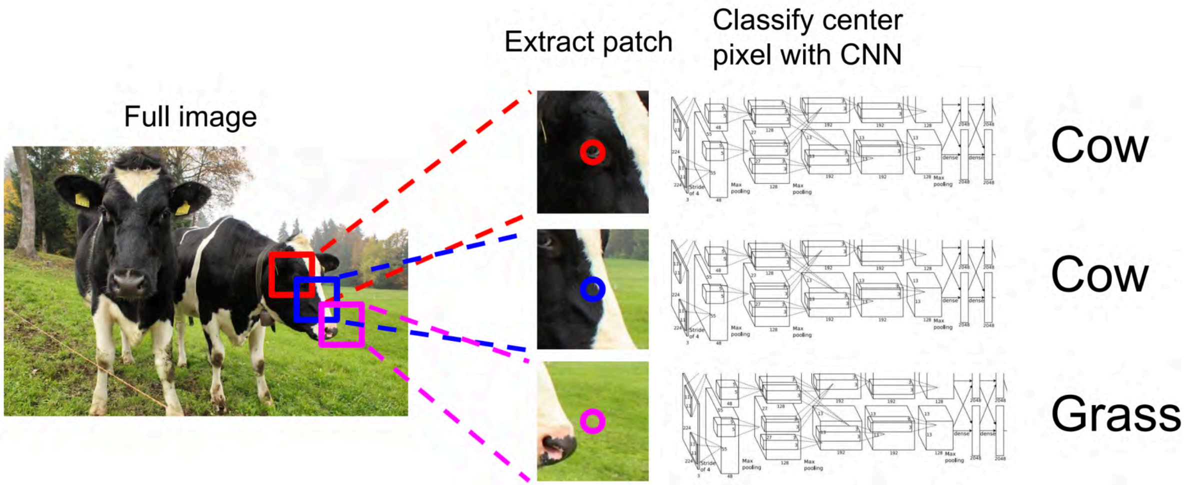
No semantic information with only a single pixel value



Extracting semantic information from the local context
(the surrounding area of a pixel)

Semantic Segmentation

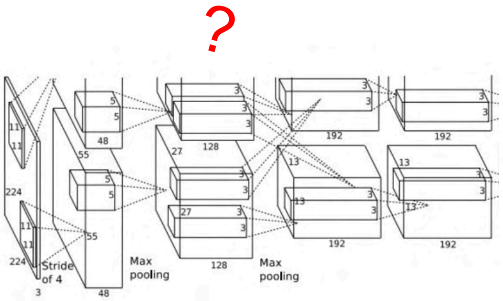
- Using image classification models
- Semantic segmentation relying on classifying image patches.
- It is inefficient and cannot capture shared representations.



Farabet et al, “Learning Hierarchical Features for Scene Labeling,” TPAMI 2013
Pinheiro and Collobert, “Recurrent Convolutional Neural Networks for Scene Labeling”, ICML 2014

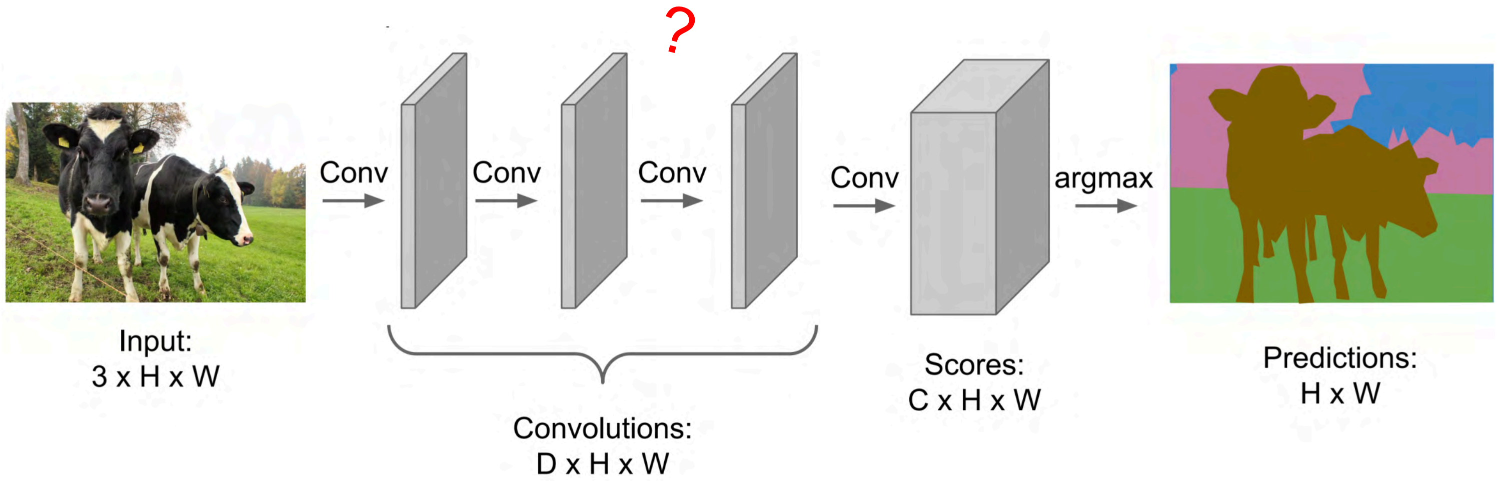
Semantic Segmentation

- Can we get segmentation map from the image in an end-to-end manner?
- Classification architectures often reduce feature spatial sizes to go deeper
- Semantic segmentation requires the output size to be the same as input size.



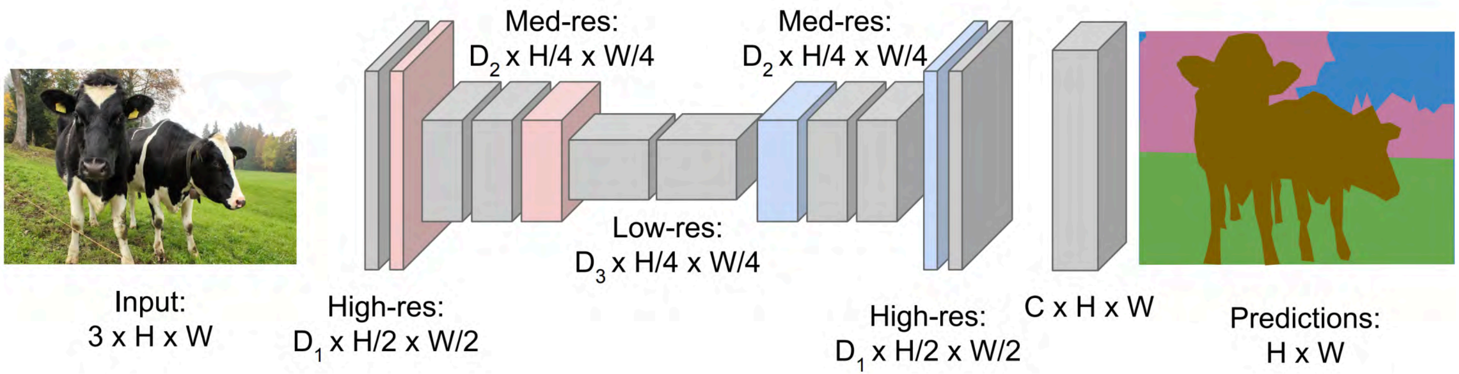
Semantic Segmentation

- Design a network with only convolutional layers without downsampling operators to make predictions for pixels all at once
- Is this a good design?



Semantic Segmentation

- Design network as a bunch of convolutional layers, with downsampling (encoder) and upsampling (decoder) inside the network
- Fully convolutional networks (FCN)
- downsampling: pooling, strided convolution
- upsampling: ?



Long, Shelhamer, and Darrell, “Fully Convolutional Networks for Semantic Segmentation”, CVPR 2015
Noh et al, “Learning Deconvolution Network for Semantic Segmentation”, ICCV 2015

Semantic Segmentation

- **Unpooling (v.s. pooling)**
- No parameters to learn (as pooling operations)

Nearest Neighbor

Input: 2 x 2

1	2
3	4

Output: 4 x 4

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

"Bed of Nails"

Input: 2 x 2

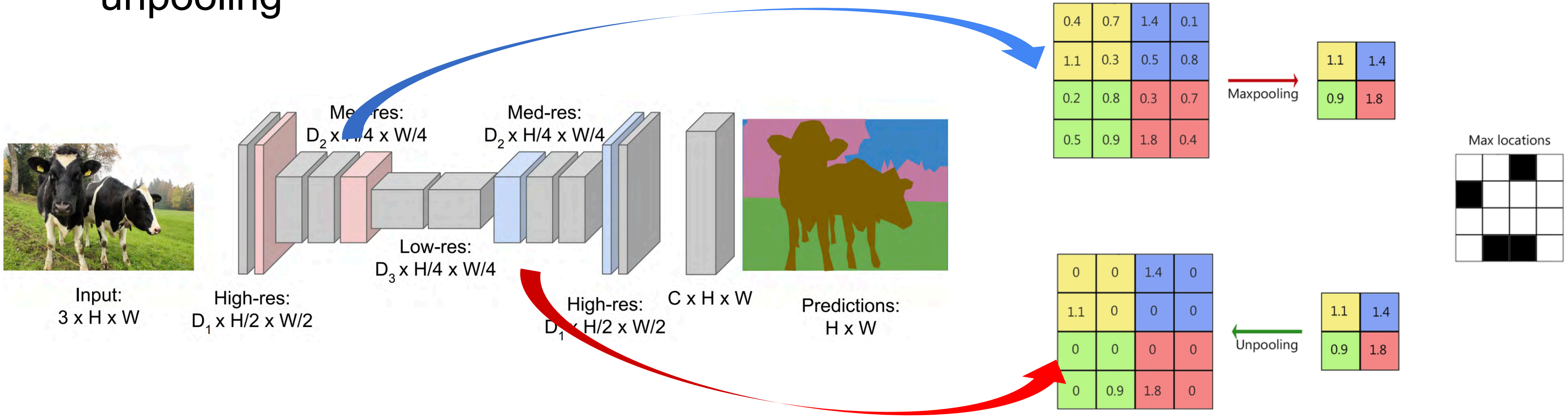
1	2
3	4

Output: 4 x 4

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

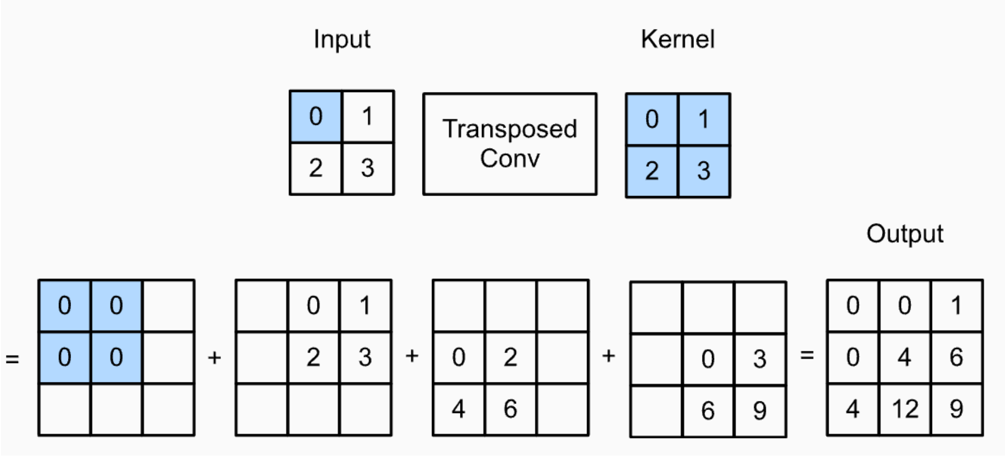
Semantic Segmentation

- **Unpooling (v.s. pooling)**
- No parameters to learn (as pooling operations)
- Recording the max locations in maxpooling and then using them for max unpooling



Semantic Segmentation

- **Learnable upsampling – transposed convolution (v.s. strided conv.)**
- Learn filters/kernels of a transposed convolution layer for upsampling

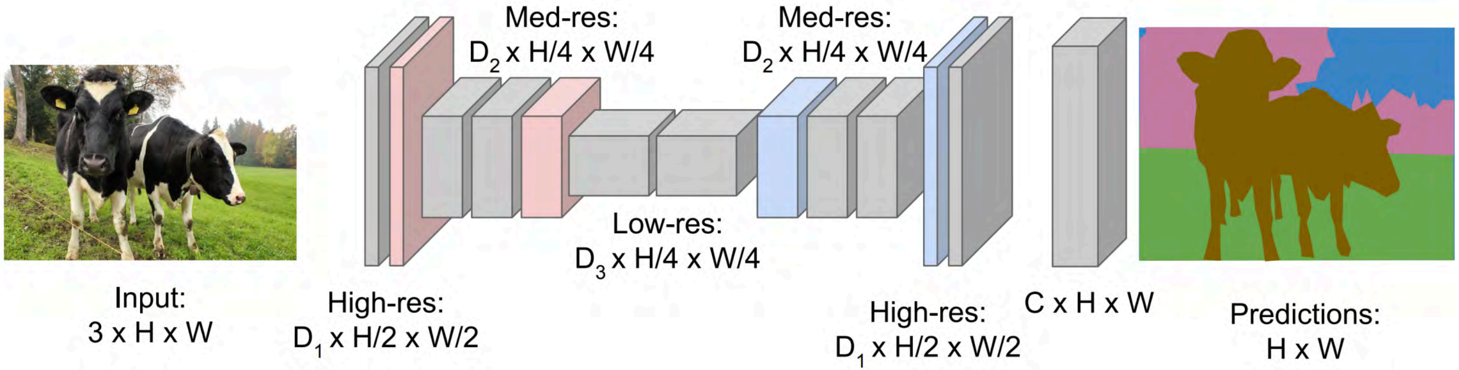


2x2 → 3x3

https://d2l.ai/chapter_computer-vision/transposed-conv.html

Semantic Segmentation

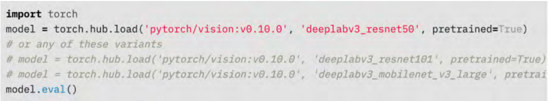
- Design network as a bunch of convolutional layers, with downsampling (encoder) and upsampling (decoder) inside the network
- Fully convolutional networks (FCN)
- downsampling: pooling, strided convolution
- upsampling: uppooling, transposed convolution



Long, Shelhamer, and Darrell, "Fully Convolutional Networks for Semantic Segmentation", CVPR 2015
Noh et al, "Learning Deconvolution Network for Semantic Segmentation", ICCV 2015

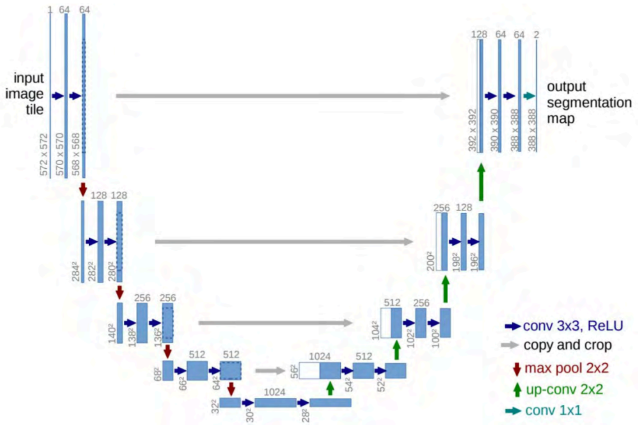
Semantic Segmentation

- Many designs of the method to achieve good performance for different application scenarios and requirements
- U-Net
- DeepLabV3
- ...

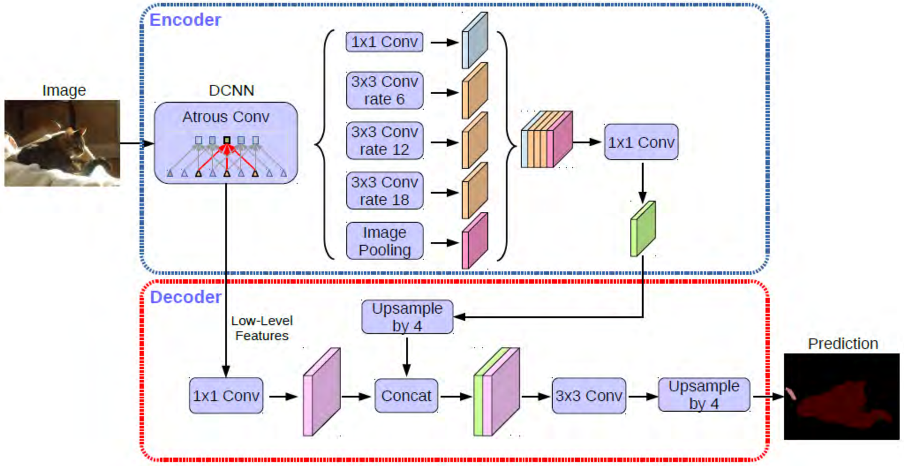


```python
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'deeplabv3_resnet50', pretrained=True)
# or any of these variants
# model = torch.hub.load('pytorch/vision:v0.10.0', 'deeplabv3_resnet101', pretrained=True)
# model = torch.hub.load('pytorch/vision:v0.10.0', 'deeplabv3_mobilenet_v3_large', pretrai
model.eval()
```

https://pytorch.org/hub/pytorch_vision_deeplabv3_resnet101/



[U-Net: Convolutional Networks for Biomedical Image Segmentation, Ronneberger et al, 2015]



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

Some Other ‘Dense Prediction’ Tasks

- Predicting pixel-wise output (map) from the input image
- Image restoration (restorating degenerated images, such as blurry, noising images)

High dynamic range imaging

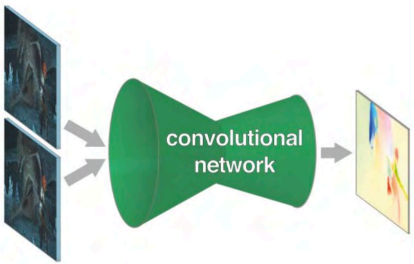
Image deblurring

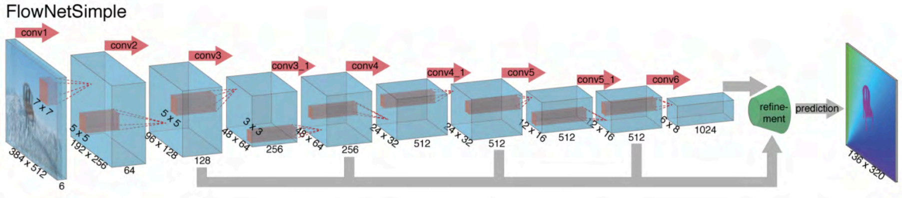
Reflection removal

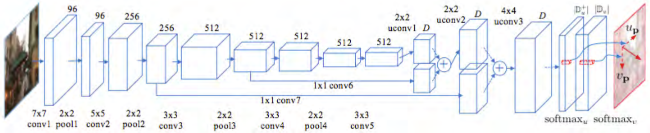
From Motion Blur to Motion Flow: a Deep Learning Solution for Removing Heterogeneous Motion Blur. ***Dong Gong****, Jie Yang, Lingqiao Liu, Yanning Zhang, Ian Reid, Chunhua Shen, Anton van den Hengel, Qinfeng Shi, CVPR, 2017.*
Attention-guided Network for Ghost-free High Dynamic Range Imaging. Qingsen Yan*, **Dong Gong***, Qinfeng Shi, Anton van den Hengel, Chunhua Shen, Ian Reid, Yanning Zhang (* Equal contribution), *CVPR, 2019.*
Seeing Deeply and Bidirectionally: A Deep Learning Approach for Single Image Reflection Removal. Jie Yang*, **Dong Gong***, Lingqiao Liu, Qinfeng Shi (* Equal contribution). *ECCV, 2018.*

Some Other ‘Dense Prediction’ Tasks

- Predicting pixel-wise output (map) from the input image
- Optical flow (motion flow) prediction
- Depth prediction



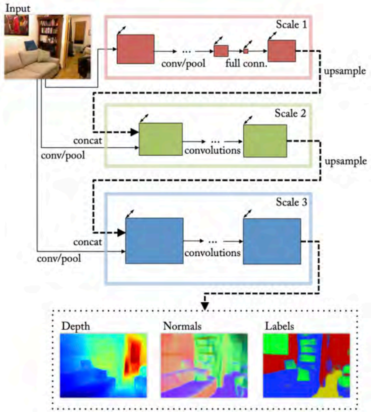


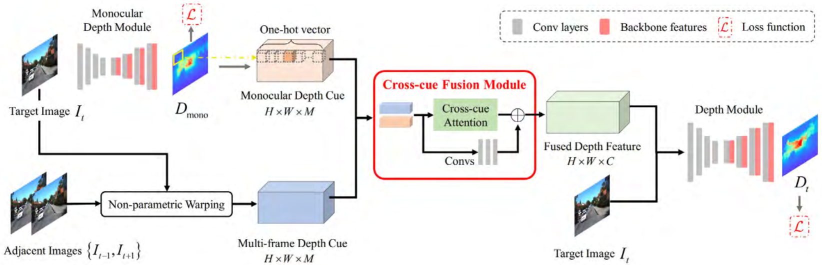


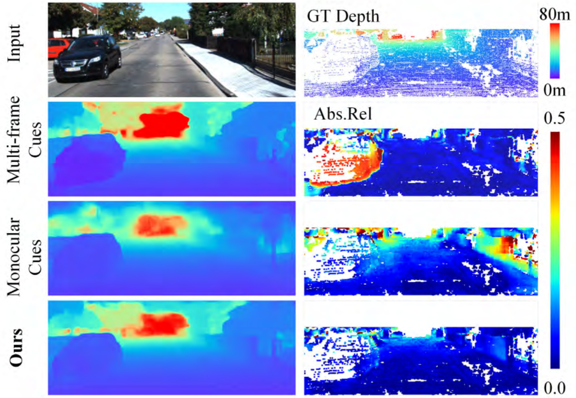
FlowNet: Learning optical flow with convolutional network, Wang et al, 2020
From Motion Blur to Motion Flow: a Deep Learning Solution for Removing Heterogeneous Motion Blur. ***Dong Gong****, Jie Yang, Lingqiao Liu, Yanning Zhang, Ian Reid, Chunhua Shen, Anton van den Hengel, Qinfeng Shi, CVPR, 2017.*

Some Other ‘Dense Prediction’ Tasks

- Predicting pixel-wise output (map) from the input image
- Depth prediction – predicting the depth map from images

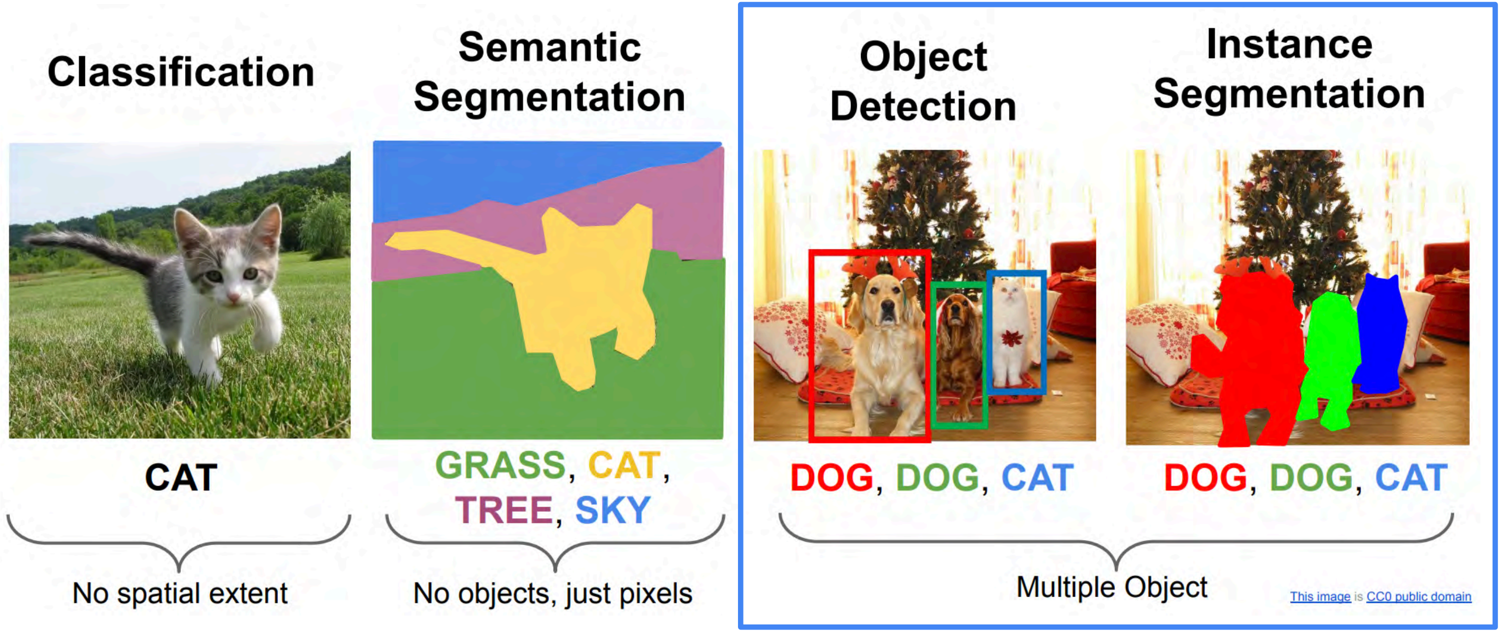






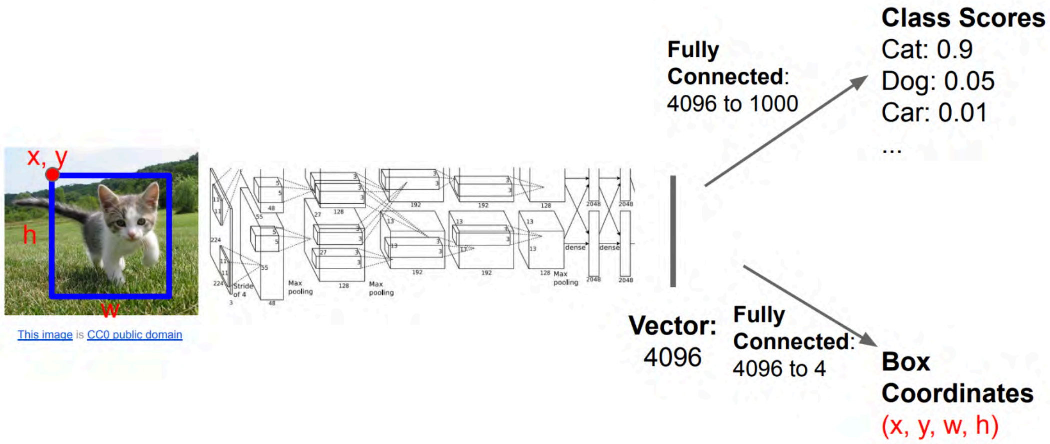
Eigen, David, and Rob Fergus. "Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture." ICCV. 2015.
Li, Rui, Dong Gong, Wei Yin, Hao Chen, Yu Zhu, Kaixuan Wang, Xiaozhi Chen, Jinqiu Sun, and Yanning Zhang. "Learning to Fuse Monocular and Multi-view Cues for Multi-frame Depth Estimation in Dynamic Scenes." CVPR 2023.

Vision Tasks Beyond Classification



Object Detection

- Object Classification + Localization
- Classification: semantic labeling (softmax+cross entropy loss)
- Localization: predicting the bounding box of each interested objects (regression problem)
- Multi-task objective

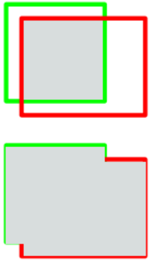


- How should we design the DNNs for object detection?

Object Detection – Evaluation metrics

- Classification
  - Accuracy: percentage of correct predictions

- Object detection & segmentation
  - Intersection-over-union (IoU)

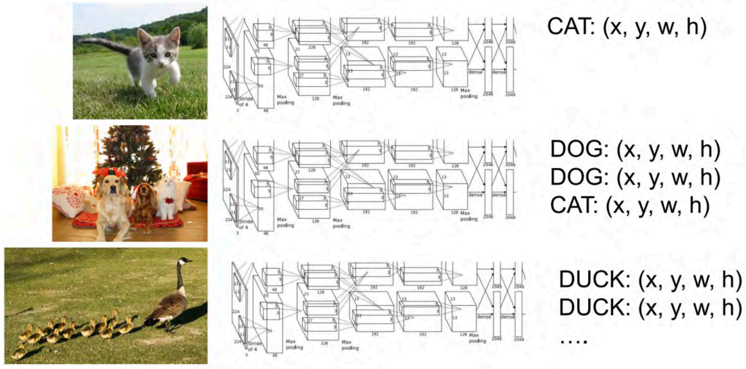


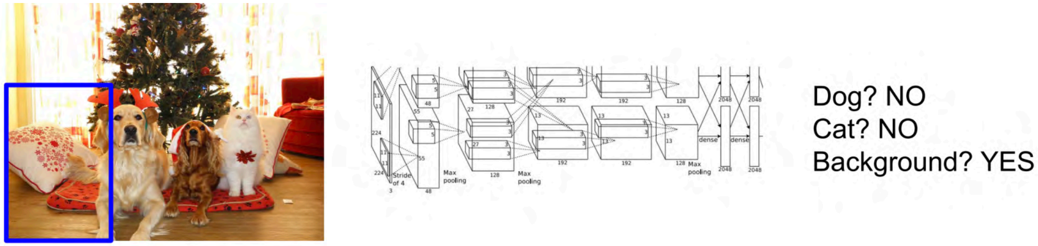
$$\mathcal{J}(\mathbf{P},\mathbf{T}) = \frac{\mathbf{P} \bigcap \mathbf{T}}{\mathbf{P} \bigcup \mathbf{T}}$$

  - IoU non-differentiable: used only for evaluation

Object Detection

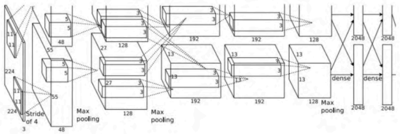
- Multiple and unknown number of objects (unknown/arbitrary total number of outputs for an image)
- A naïve solution: sliding window with varying scales and locations
- Problem: too many options for locations, scales, and aspect ratios, leading to highly expensive computations.



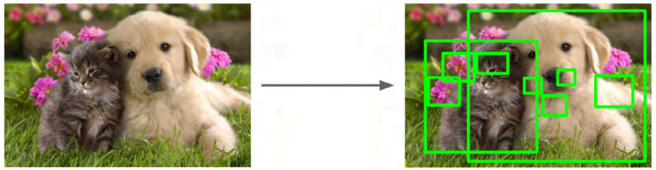


Object Detection

- Multiple and unknown number of objects (unknown/arbitrary total number of outputs for an image)
- A naïve solution: sliding window with varying scales and locations
- Problem: too many options for locations, scales, and aspect ratios, leading to highly expensive computations.
- Solution: region proposals – generating bounding box proposals (potentially to be objects) based on other methods/priors -- can be fast



Dog? NO
Cat? NO
Background? YES

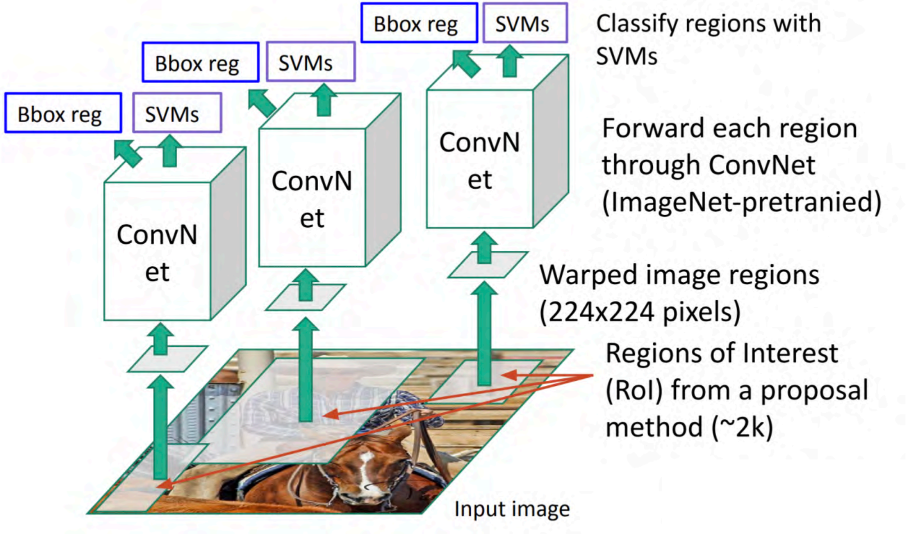


Alexe et al, "Measuring the objectness of image windows", TPAMI 2012
Uijlings et al, "Selective Search for Object Recognition", IJCV 2013
Cheng et al, "BING: Binarized normed gradients for objectness estimation at 300fps", CVPR 2014
Zitnick and Dollar, "Edge boxes: Locating object proposals from edges", ECCV 2014

Object Detection – R-CNN

- Published in CVPR 2014
- Not end-to-end training
- Extracting features in the generated proposals with a pre-trained image classification network (on ImageNet)
- Classify the regions and refining the bounding box location (based on the proposal box)
- Slow. Independent forward process fo each RoI (region of interest)

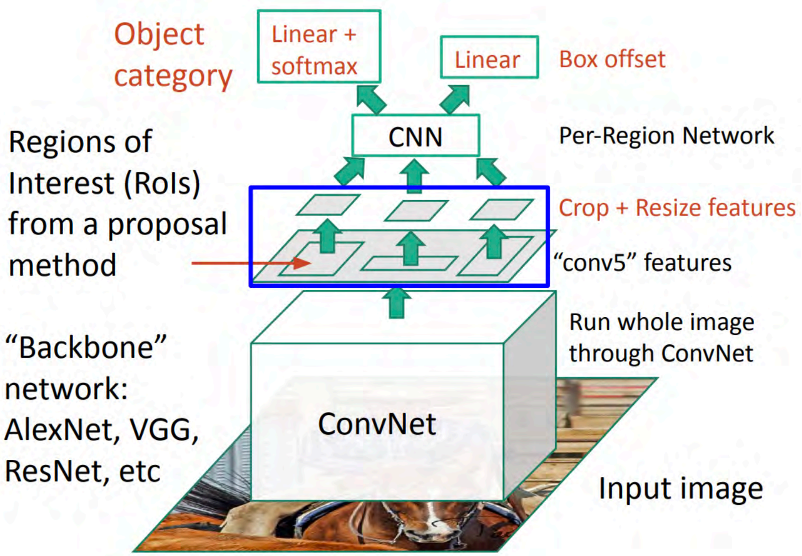
Predict “corrections” to the RoI: 4 numbers: (dx, dy, dw, dh)



Girshick et al, “Rich feature hierarchies for accurate object detection and semantic segmentation”, CVPR 2014.
https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0

Object Detection – Fast R-CNN

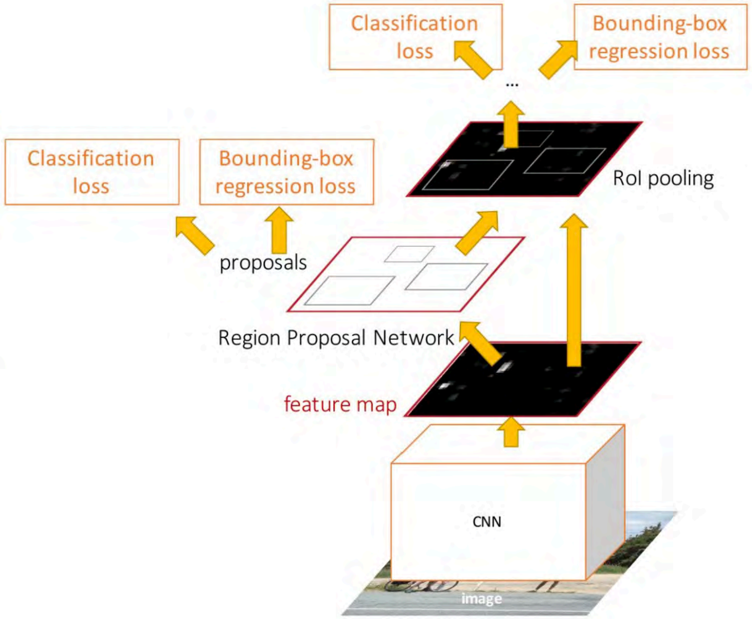
- Published in ICCV 2015
- Relying on pre-generated propsoal as R-CNN
- Forward the image through CNN before cropping with proposal bbox.
- Cropping on con feature map – RoI pooling! – applying the proposal bbox on image coordinates to feature maps
- RoI align is a “sub-pixel” version of RoI pooling – from “Mask R-CNN”
- End-to-end



Girshick, “Fast R-CNN”, ICCV 2015.
He et al, “Mask R-CNN”, ICCV 2017 – RoI Align

Object Detection – Faster R-CNN

- Insert a Region Proposal Network (RPN) into Fast R-CNN to predict proposals from features
- Region proposal network: predicting whether there should be an anchor bbox at a location – the classification loss on RPN
- Generating proposal in a unified framework and network
- Two-stage method: generating proposal (RPN) and detection
- Non-maximun suppression (NMS) – filtering the redundant proposal boxes (relying on proposal confidence, overlapping/IoU)

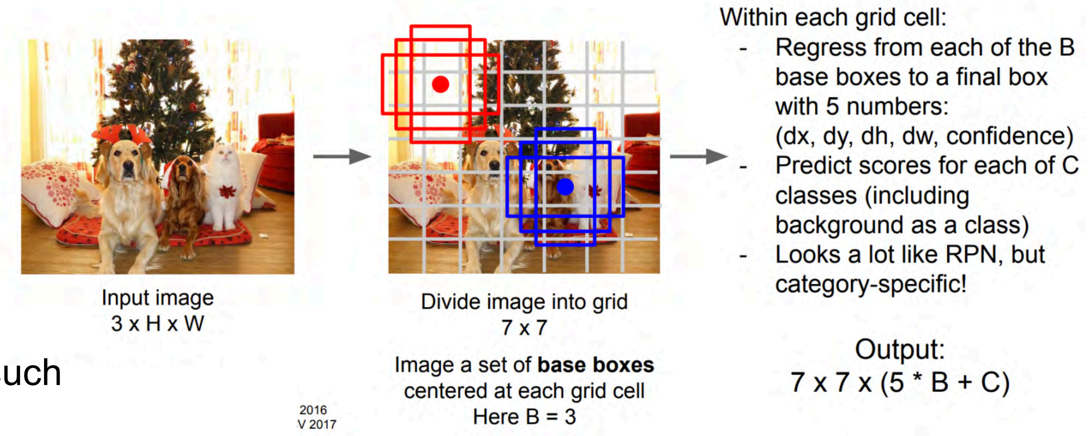


Ren et al, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”, NIPS 2015
https://dl.dropboxusercontent.com/s/vlyrkgd8nz8gy5l/fast-rcnn.pdf?dl=0
http://cs231n.stanford.edu/slides/2023/lecture_11.pdf

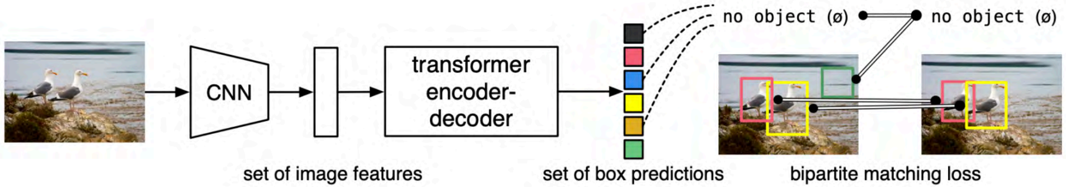
Object Detection – one-stage methods

- YOLO (), SSD, RetinaNet
- Much faster

- Others:
- Some anchor box free methods -- such as FOCS.
- Transformer-based detection -- DETR



http://cs231n.stanford.edu/slides/2023/lecture_11.pdf

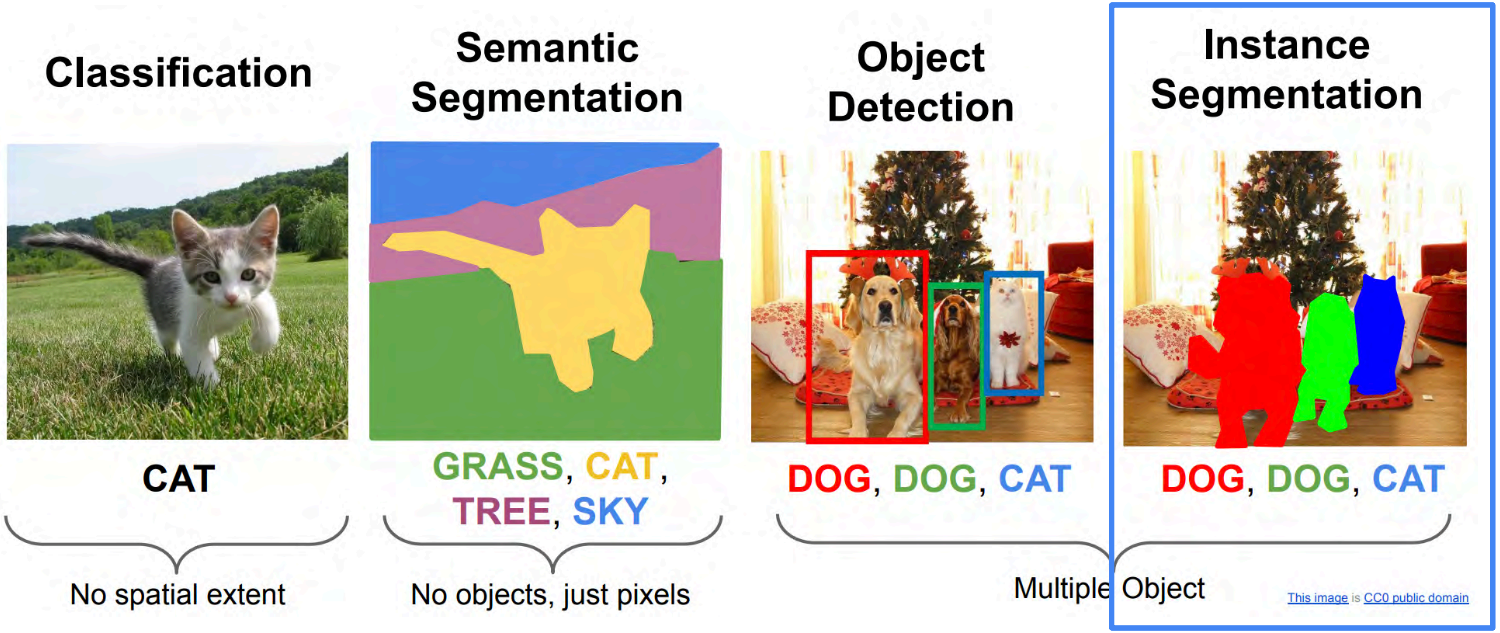


Redmon et al, “You Only Look Once: Unified, Real-Time Object Detection”, CVPR 2016
Liu et al, “SSD: Single-Shot MultiBox Detector”, ECCV 2016
Lin et al, “Focal Loss for Dense Object Detection”, ICCV 2017
Tian, Zhi, et al. "Fcos: Fully convolutional one-stage object detection." *ICCV*. 2019.

Carion, Nicolas, et al. "End-to-end object detection with transformers." *ECCV*, 2020.

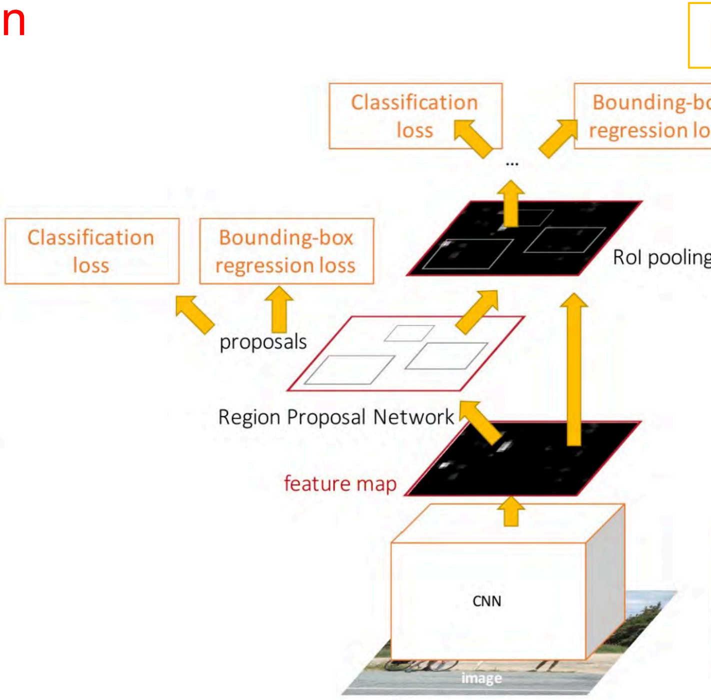
Instance Segmentation

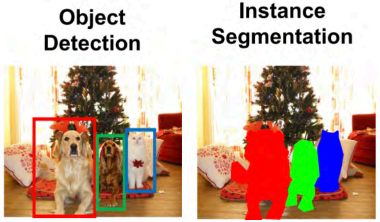
- Combination of detection + segmentation
- Needs to identify each object

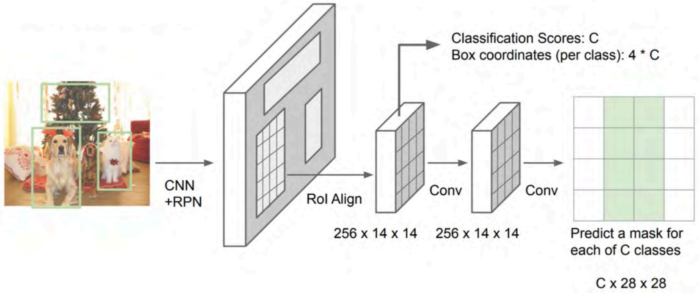


Instance Segmentation

- Faster R-CNN + segmentation mask prediction
- Mask F-CNN
- Multi-task learning
- End-to-end

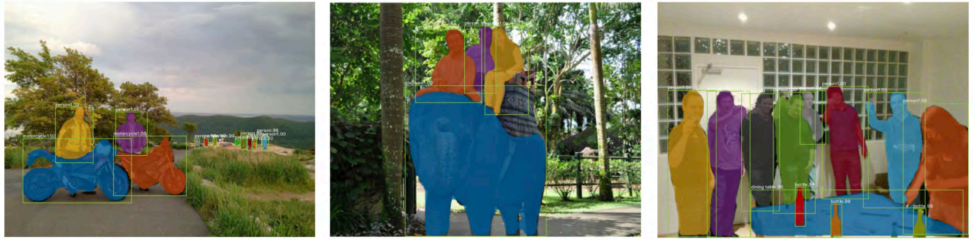






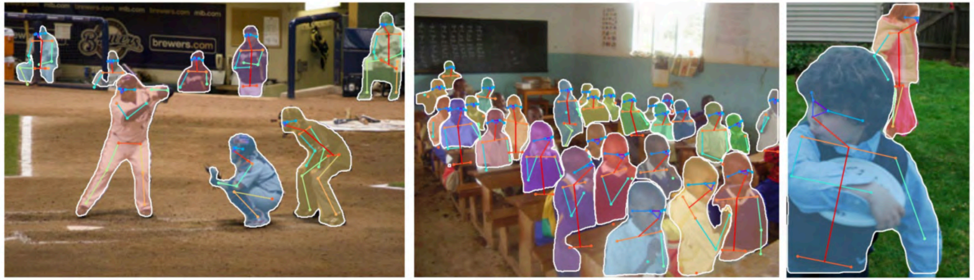
He et al, “Mask R-CNN”, ICCV 2017

Instance Segmentation



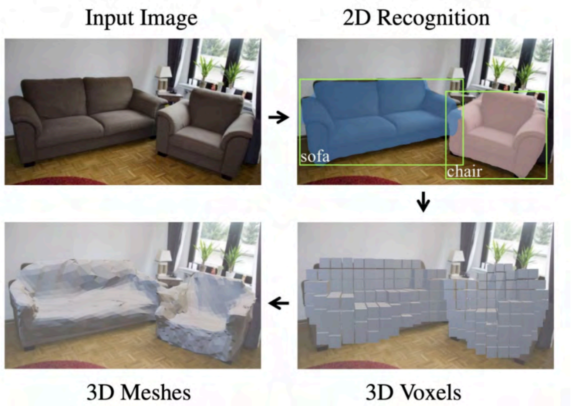
He et al, “Mask R-CNN”, ICCV 2017

Can also predict pose



He et al, “Mask R-CNN”, ICCV 2017

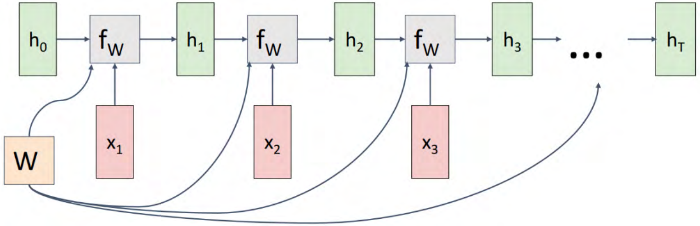
Can work in 3D



Gkioxari et al., Mesh RCNN, ICCV 2019

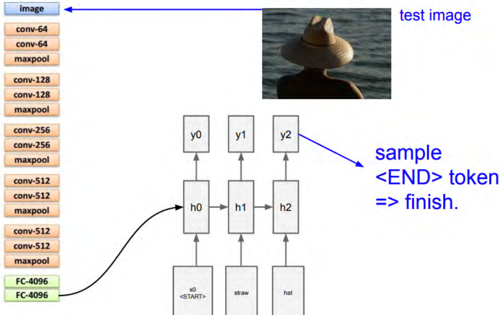
[Learning and Memorizing Representative Prototypes for 3D Point Cloud Semantic and Instance Segmentation](). Tong He*, **Dong Gong***, Zhi Tian, Chunhua Shen. *ECCV, 2020*. (* Equal contr.)

Recurrent Neural Networks (RNNs)



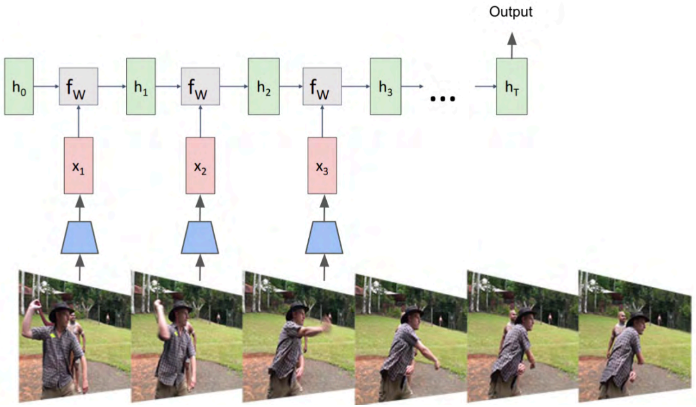
- Sequential modeling
- RNN, GRU, LSTM, …

- Action recognition or video classification – can also be handled by 3D CNNs.
- Image captioning



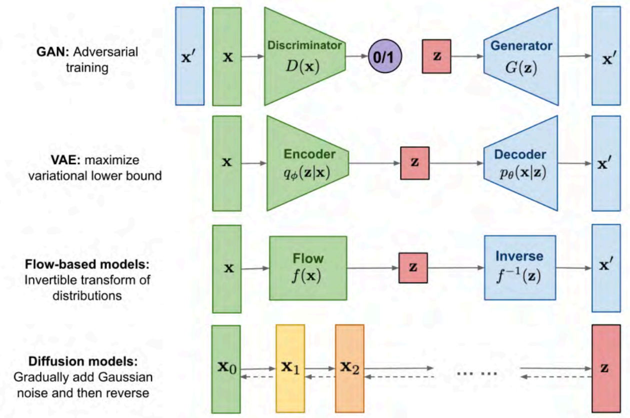
A dog is running in the grass with a frisbee

Two giraffes standing in a grassy field



Generative Models

- Generating images -- random sampling or generation from text description (conditional modeling)
- Modeling the distribution of the data (images)
- Variational Autoencoder (VAE)
- Generative adversarial network (GAN)
- Flow-based models
- Diffusion model



https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Images generated by GAN model ("BigGAN")

Image generated by diffusion models (Stable Diffusion)

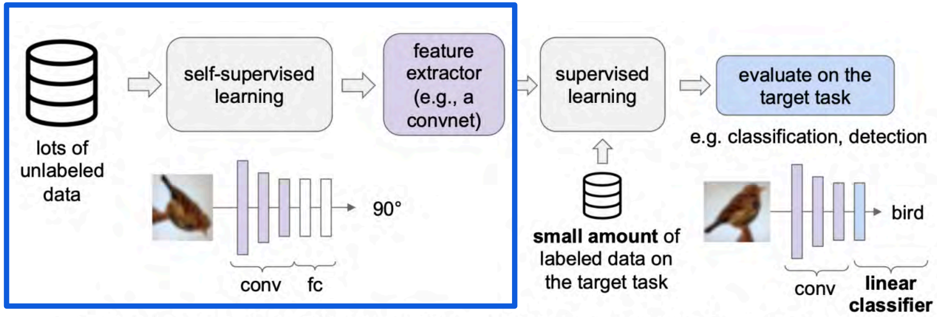
A brain riding a rocketship heading towards the moon.

"A raccoon astronaut with the cosmos reflecting on the glass of his helmet dreaming of the stars"

Image generated by diffusion models based on text description (Imagen by Goodle and DALL-E by OpenAI)

Self-Supervised Learning

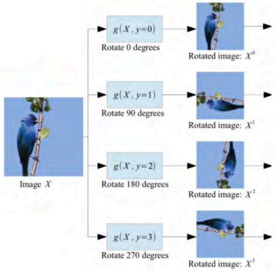
- Learning strong representations without supervision
- Using large-scale unlabeled data
- Can be used as backbone model for downstream tasks
- Need carefully designed pretext supervision for training – image reconstruction (from cropped image/noise/patches); predicting rotations; contrastive learning (relying on data augmentation); …



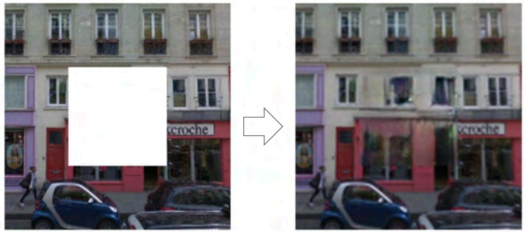
1. Learn good feature extractors from self-supervised pretext tasks, e.g., predicting image rotations

2. Attach a shallow network on the feature extractor; train the shallow network on the target task with small amount of labeled data

http://cs231n.stanford.edu/slides/2023/lecture_13.pdf



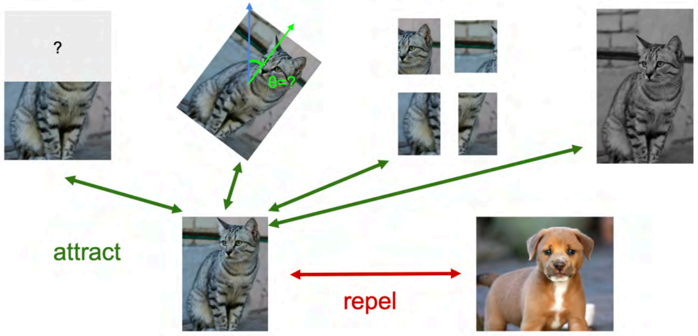
Unsupervised Representation Learning by Predicting Image Rotations, ICLR 2018



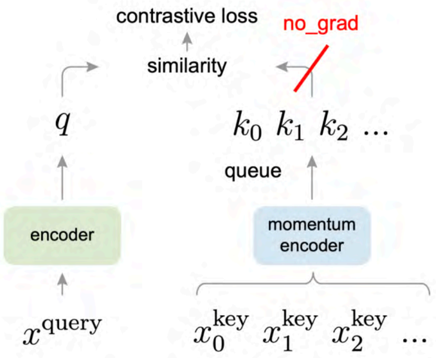
Pathak, Deepak, et al. "Context encoders: Feature learning by inpainting." *CVPR*. 2016.

Self-Supervised Learning

- Learning strong representations without supervision
- Using large-scale unlabeled data
- Can be used as backbone model for downstream tasks
- Need carefully designed pretext supervision for training – image reconstruction (from cropped image/noise/patches); predicting rotations; contrastive learning (relying on data augmentation); …



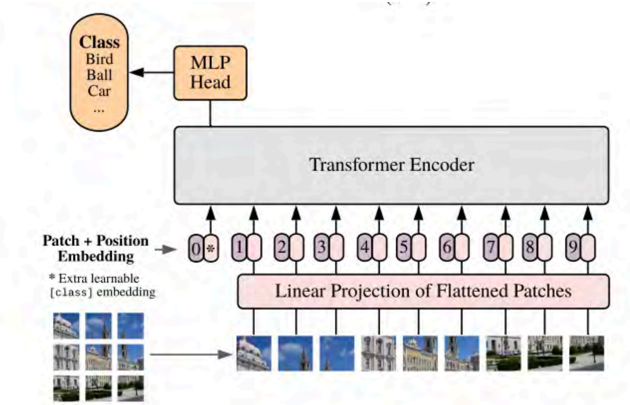
The idea of contrastive learning



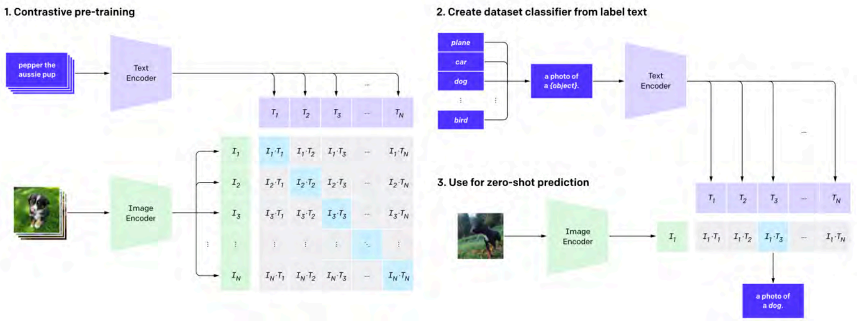
Momentum Contrastive Learning (MoCo), He et al, CVPR 2020

Vision Transformer Models

- Transformer-based models are getting to be powerful backbone for vision tasks.
- ViT, SwinTransformer, …
- Self-supervised learning based pretraining and Language–Image Pre-training help to train large-scale Transformers with strong representation ability.
- Convolution-based networks are still useful! Carefully designed large CNNs can also outperform Transformers based networks.



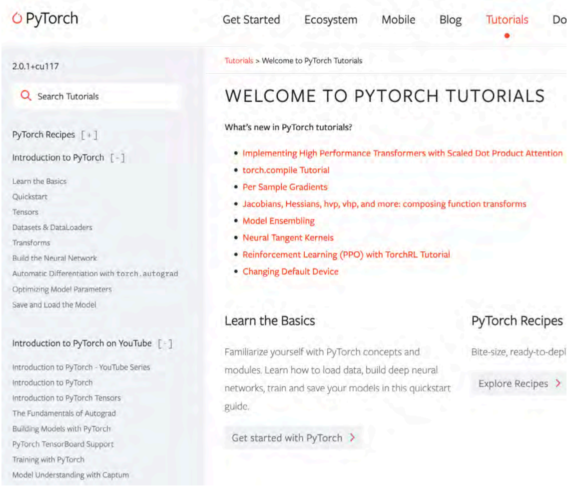
Dosovitskiy, Alexey, et al. "An image is worth 16x16 words: Transformers for image recognition at scale." *arXiv preprint arXiv:2010.11929* (2020).

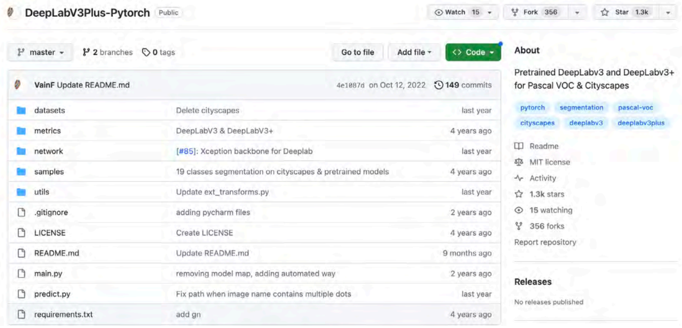


https://openai.com/research/clip

A little bit about practice

- Implementation based on PyTorch, Tensorflow, or other packages.
  - Learning programming using the official tutorials (e.g, PyTorch tutorials)
- Google colab (https://colab.google/) – restricted free computational resources
- Read the code of the classical models to know the details
  - You can get almost all resources via Google
- Given a task, do some research at first to get an overview of the area and different methods
  - Find the code of existing works as your baseline models
  - Test on your data and task





Example: Github repo of DeepLabV3 pytorch implementation

A little bit about practice

- Implementatic ... ckages.
- Google colab ... utational

d

,

: l

c

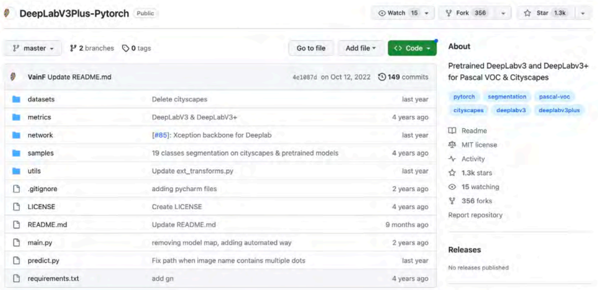

```python
class VGG(nn.Module):
    def __init__(
        self, features: nn.Module, num_classes: int = 1000, init_weights: bool = True, (
    ) -> None:
        super().__init__()
        _log_api_usage_once(self)
        self.features = features
        self.avgpool = nn.AdaptiveAvgPool2d((7, 7))
        self.classifier = nn.Sequential(
            nn.Linear(512 * 7 * 7, 4096),
            nn.ReLU(True),
            nn.Dropout(p=dropout),
            nn.Linear(4096, 4096),
            nn.ReLU(True),
            nn.Dropout(p=dropout),
            nn.Linear(4096, num_classes),
        )
        if init_weights:
            for m in self.modules():
                if isinstance(m, nn.Conv2d):
                    nn.init.kaiming_normal_(m.weight, mode="fan_out", nonlinearity="re
                    if m.bias is not None:
                        nn.init.constant_(m.bias, 0)
                elif isinstance(m, nn.BatchNorm2d):
                    nn.init.constant_(m.weight, 1)
                    nn.init.constant_(m.bias, 0)
                elif isinstance(m, nn.Linear):
                    nn.init.normal_(m.weight, 0, 0.01)
                    nn.init.constant_(m.bias, 0)

    def forward(self, x: torch.Tensor) -> torch.Tensor:
        x = self.features(x)
        x = self.avgpool(x)
        x = torch.flatten(x, 1)
        x = self.classifier(x)
        return x
```

```python
def make_layers(cfg: List[Union[str, int]], batch_norm: bool = False) -> nn.Sequenti
    layers: List[nn.Module] = []
    in_channels = 3
    for v in cfg:
        if v == "M":
            layers += [nn.MaxPool2d(kernel_size=2, stride=2)]
        else:
            v = cast(int, v)
            conv2d = nn.Conv2d(in_channels, v, kernel_size=3, padding=1)
            if batch_norm:
                layers += [conv2d, nn.BatchNorm2d(v), nn.ReLU(inplace=True)]
            else:
                layers += [conv2d, nn.ReLU(inplace=True)]
            in_channels = v
    return nn.Sequential(*layers)
```

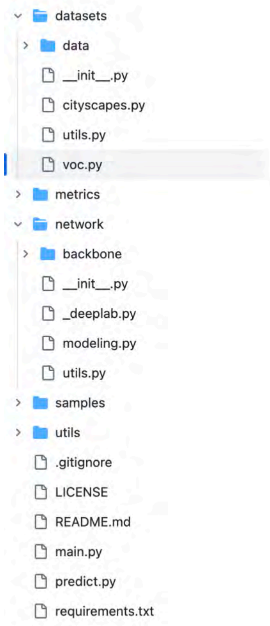
Example: VGG model implementation
https://github.com/pytorch/vision/blob/main/torchvision/models/vgg.py

A little bit about practice



- Implementation based on PyTorch, Tensorflow, or other packages.
- Google colab (https://colab.google/) – restricted free computational resources
- Read the code of the classical models to know the details
- Given a task, do some research at first to get an overview of the area and different methods
- Data + Network + Optimizer + Training/eval script + …
- There are many programming ways to implement a specific model
  - Getting experiences by reading more different code

- Deep learning book + tutorials with more practice/implementation related materials: https://d2l.ai/index.html



Example: Github repo of DeepLabV3 pytorch implementation

Summary

Part 1

- Why do we need non-linear deep neural networks (DNNs)
- CV applications of DNNs
- Convolutional Neural Networks (CNNs) – Conv./padding/stride/pooling/…
- Training DNNs – backpropagation/optimization/data augmentation/regularization/dropout/batchnorm/data preprocessing

Part 2

- Semantic segmentation and other (image restoration/depth est./optical flow)
- Object detection (R-CNN series and one-stage methods, e.g., YOLO)
- Instance segmentation (Mask R-CNN)
- Others – RNNs (action recog./image captioning), generative models, self-supervised learning, Transformer-based models