Deep Learning
Theory I

Girish Varma

Center for Security, Theory & Algorithmic Research (cstar.iiit.ac.in)

Machine Learning Lab (mll.iiit.ac.in)

IIIT Hyderabad

girishvarma.in

AI, ML and DL

Artificial Intelligence

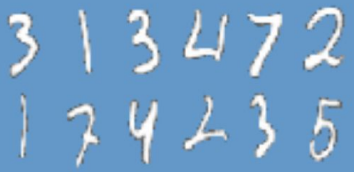
Any technique that enables computers to mimic human behavior

Machine Learning

Ability to learn without explicitly being programmed

Deep Learning

Learn underlying features in data using neural networks



A Machine Learning Problem

Given a image of a handwritten digit, find the digit.

No well defined function from input to output.

Programming vs Machine Learning

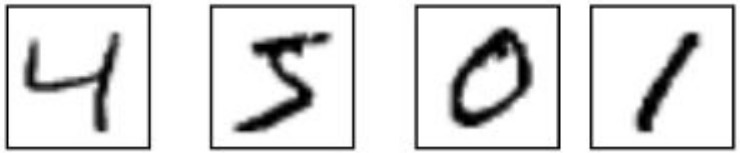
Programming:

Find the shortest path in an input graph **G**.

- Implement Dijkstra's algorithm for shortest path in a programming language.

Machine Learning:

Find the handwritten digit in an image.



- Collect (image, digit) pairs (**dataset**).
- **Train** a machine learning **model** to **fit** the dataset.
- Given a new image, apply the model to get the digit (**testing** or **inference**).

Dataset

- Consist of **(x,y)** pairs, **x** is the input and **y** is called the label.
- Examples
  - MNIST: x is a 28x28 b/w image of a handwritten digit, y is a digit in 0 to 9.
  - CIFAR10: x is a 32x32 color image, y is a label in {aeroplane, automobile, bird, car ..}. Y is given as a number in 0 to 9, and there is a mapping between the numbers and the correct label.
- Divided into train, test and validation.

Model

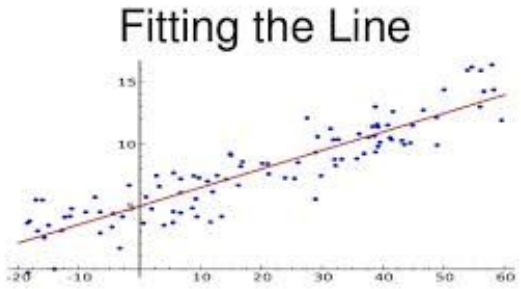
The function that maps the input to the output.

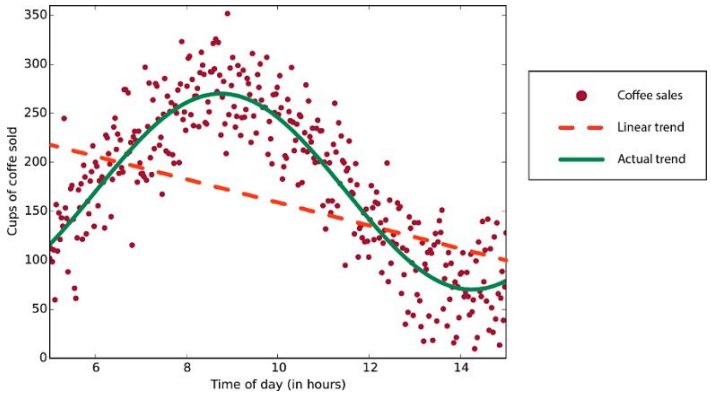
$$y = f_{\theta}(x)$$

A model has learnable parameters, θ.

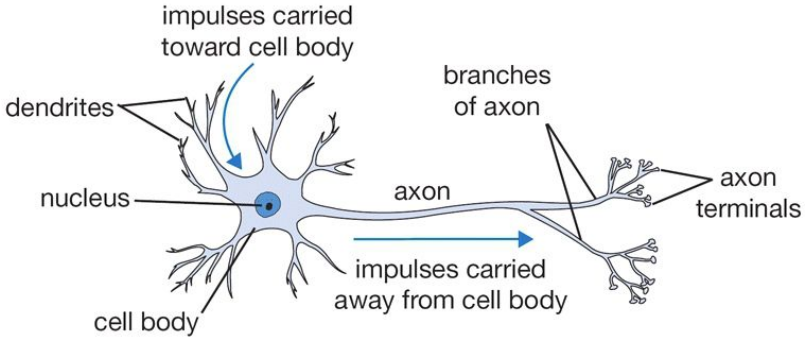
1. Fit a line to a set of points.
   - Slope and offset are learnable parameters.
2. Fit a degree 4 polynomial.
   - Coefficients are learnable parameters.
3. Fit a SVM (Support Vector Machine)

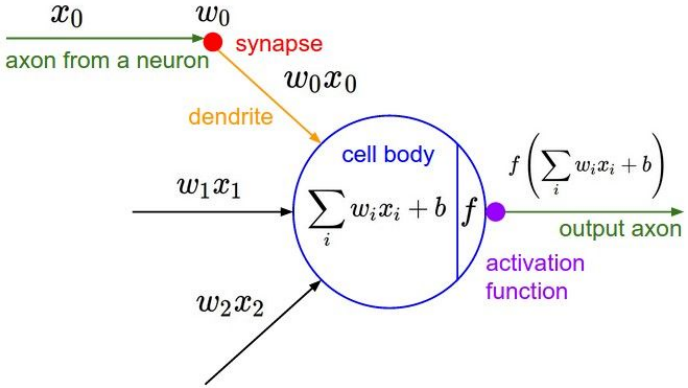
Exceptions : Nearest Neighbour Classifier





The Neural Network Model for Binary Classification

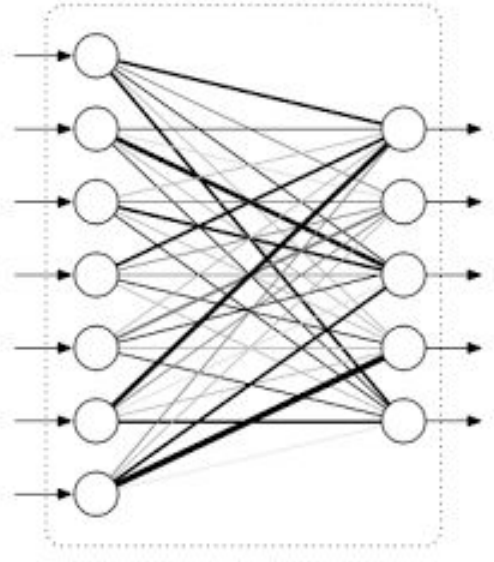




- Neuron or Perceptron
  - Input X is n dimensional, Y is 1 dimensional.
  - Has learnable parameters W = (W_1,W_2,..., W_n) (weights) and b (bias).
  - Y = **σ**(∑$W_i X_i$+ b)
  - **σ** is a non linear activation function.

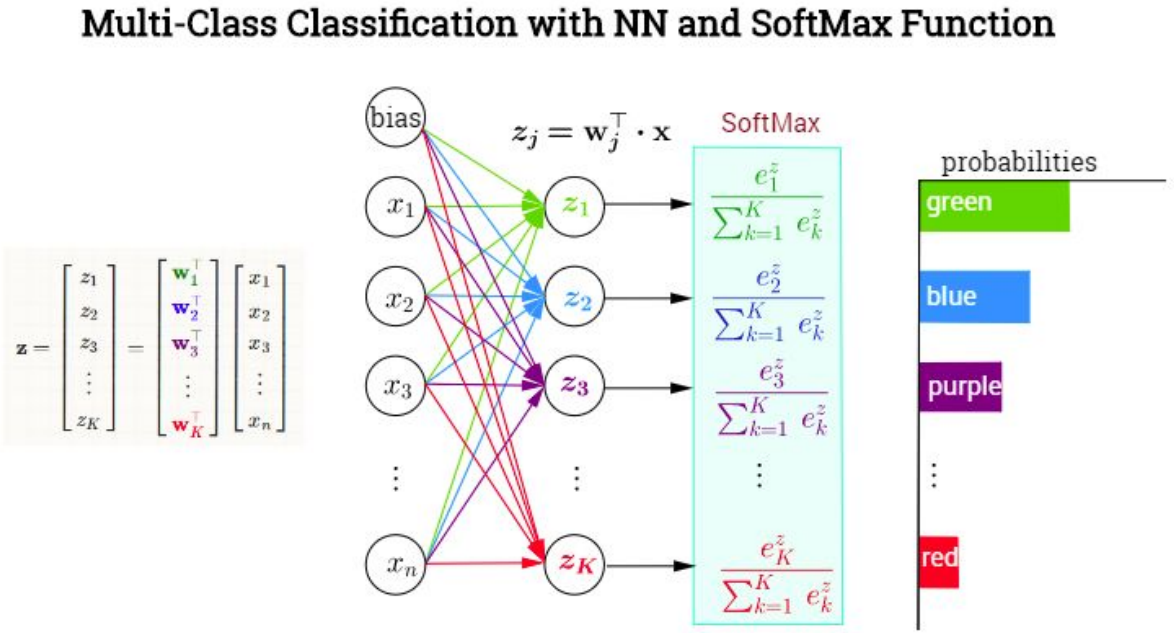
Neural Network Model for Multi-Class

- Fully Connected or Linear
  - Y is also multidimensional (dimension m).
  - Has learnable parameters W = (W_{ij}) and b = (b_j)
    where i <= n, j <= m
  - $Y = \sigma(WX + b)$
- The class predicted for input X is the i with maximum value of Y_i.



Softmax Function

Converts output to prediction probabilities.



Convert vector to positive valued:

$$Z_i \Rightarrow \exp(Z_i)$$

Normalize to get probability distribution

$$\frac{\exp(Z_i)}{\Sigma_j \exp(Z_j)}$$

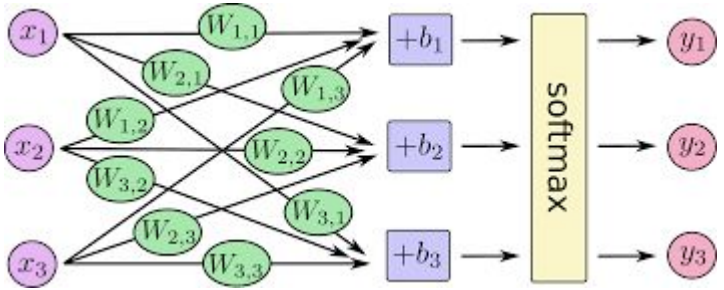
MNIST Classification

Input : x is a [28,28] shaped tensor, giving pixel values of the image

Output : y is a [10] shaped tensor, giving the probabilities of being 0 to 9.

If the dataset gives y as a digit, convert it to probability vector by one hot encoding.

Use Softmax function for converting real valued output to probabilities.



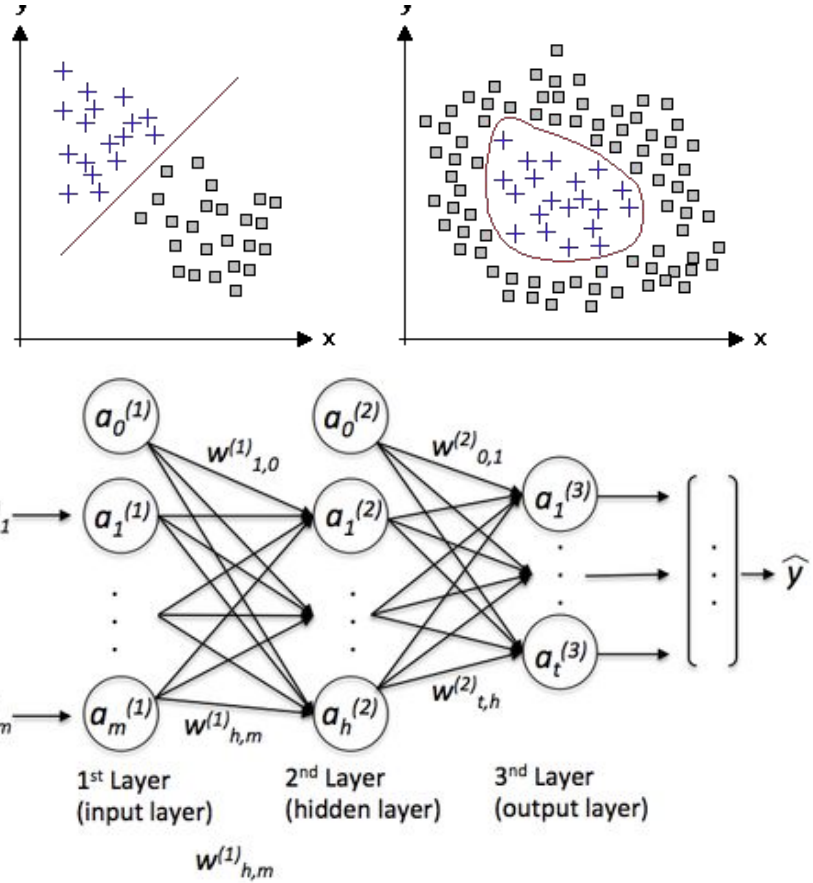
Multilayered Network

Complex data fits only more complex models.

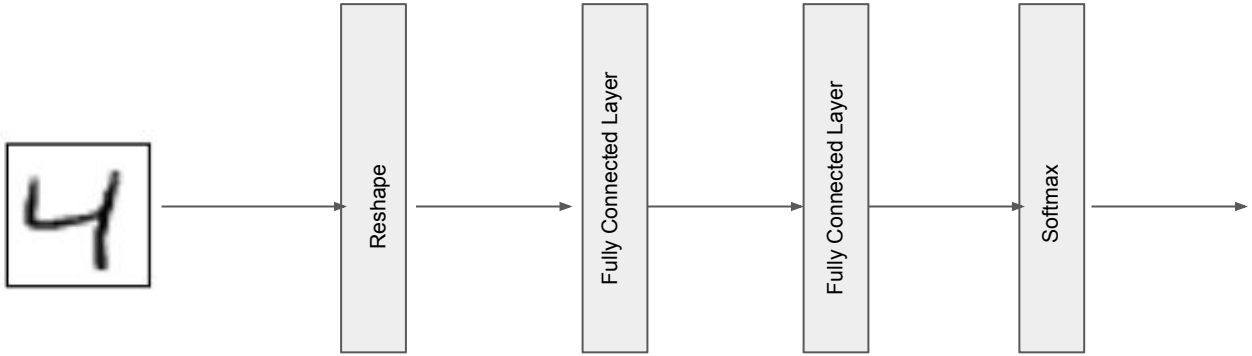
Obtain complex models by layering multiple linear layers.

Multilayered Perceptron (MLP)

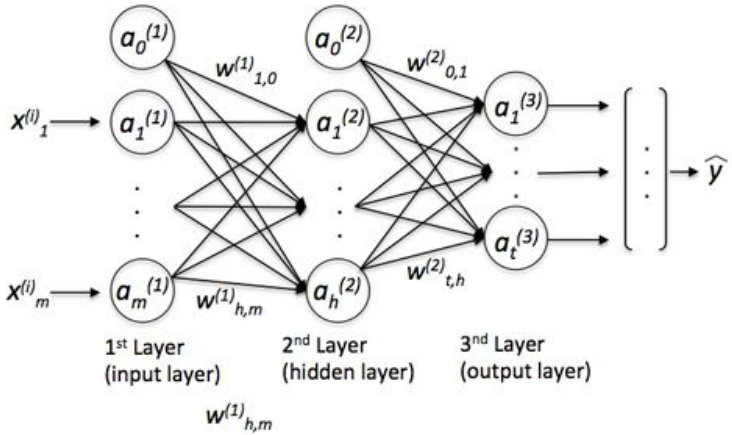
- Multiple Linear layers one following the other.
- Y = **σ**(V **σ**(WX+ b) + c)
- Intermediate outputs are called hidden units.



A MLP model for MNIST



Predicted probabilities for different digits

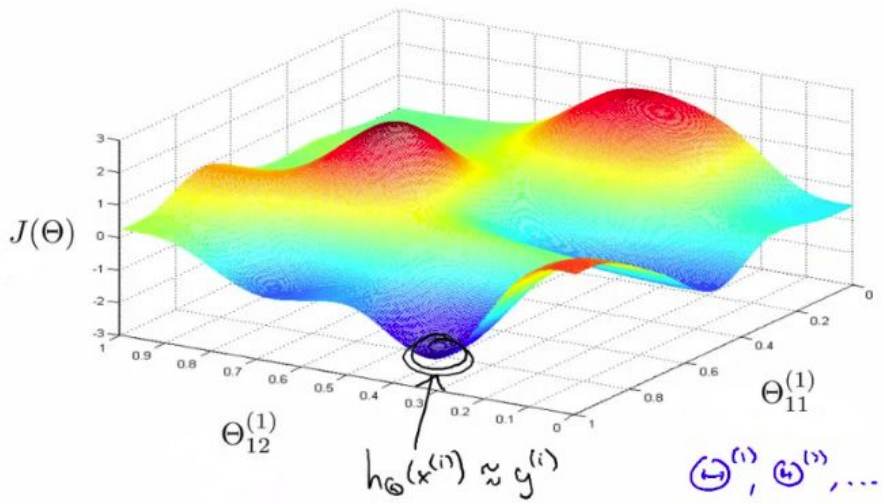


Training a Model

The process of finding the right parameters for the model.

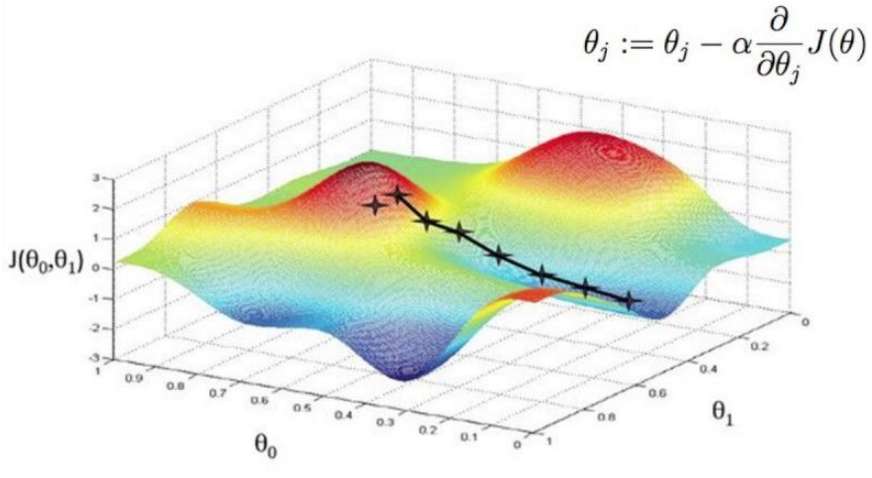
Loss Function

- Loss Function : A function that computes the difference between the predicted output and the correct output.
  - Eg: Mean Squared Error
    - $(f(x) - y_{correct})^2$. $y_{correct}$ is also called the ground truth.
  - Eg: Cross Entropy Loss
    - $\sum_i y_{correct}(i) \log y_{pred}(i)$



Gradient Descent

Gradient Descent : Change the parameters θ slightly such that the loss function decreases. Gradients are the partial derivatives of the loss function wrt. the parameters.



Mathematically

Compute Loss (Forward pass):

$$J = \frac{1}{n}\sum_{i}(y_i - (mx_i + b))^2$$

Compute partial derivatives (Backward pass)

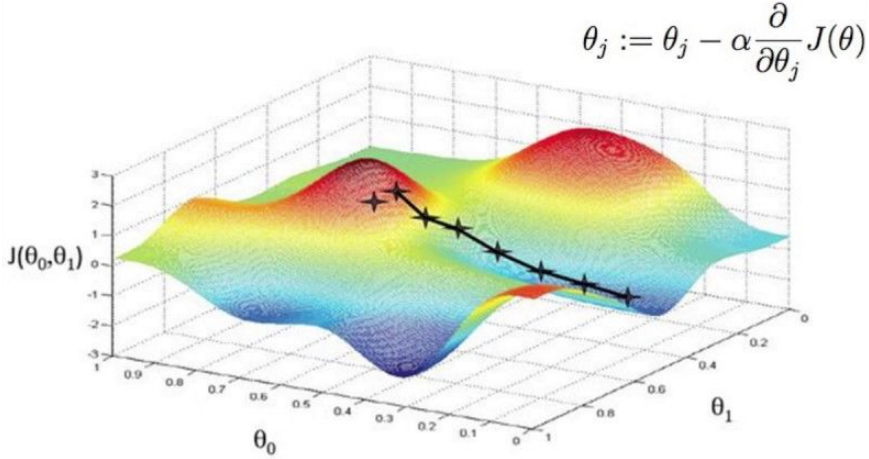
$$\frac{\partial J}{\partial m} = \frac{2}{n}\sum_{i} x_i \cdot (y_i - (mx_i + b))$$

$$\frac{\partial J}{\partial b} = \frac{2}{n}\sum_{i} (y_i - (mx_i + b))$$

Update Learnable parameters

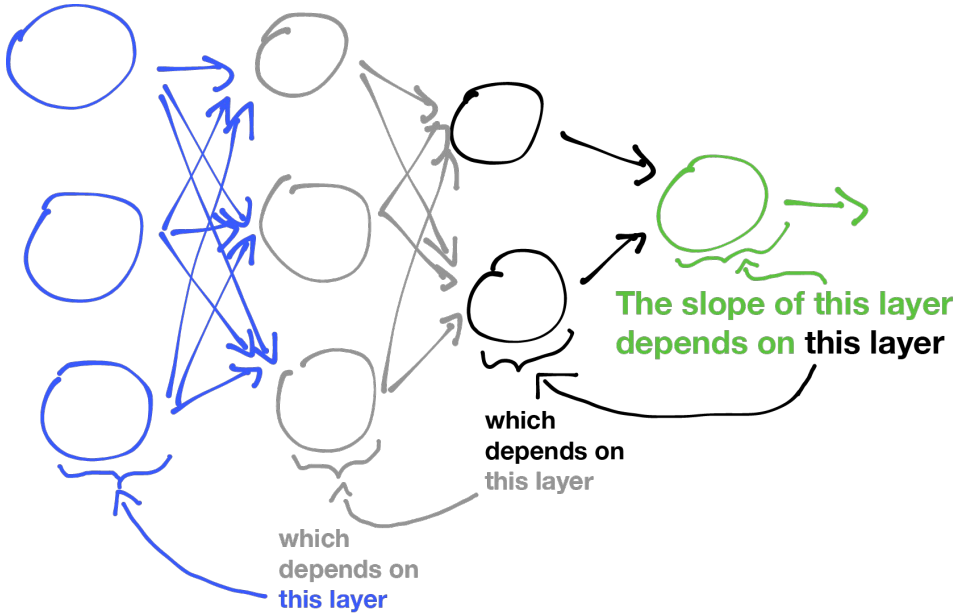
$$m := m - \alpha \cdot \frac{\partial J}{\partial m}$$

$$b := b - \alpha \cdot \frac{\partial J}{\partial b}$$



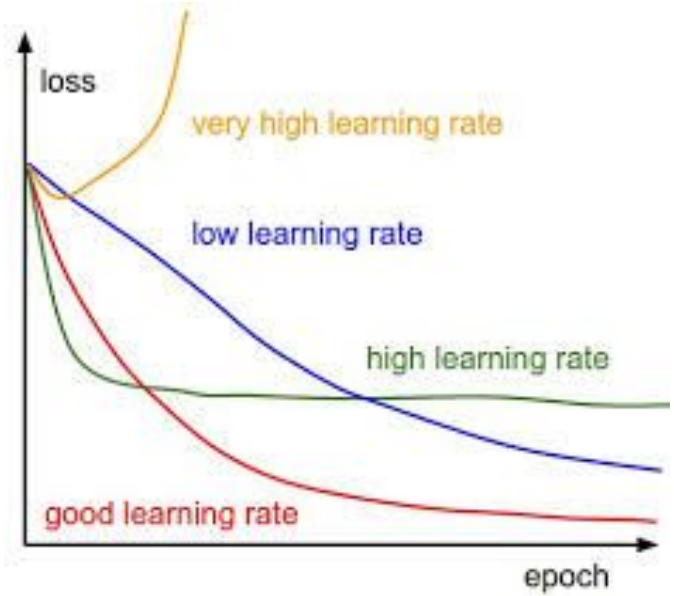
Backpropagation for Multilayered Networks

Backpropagation : The process of finding the gradients of parameters in a multilayered network.

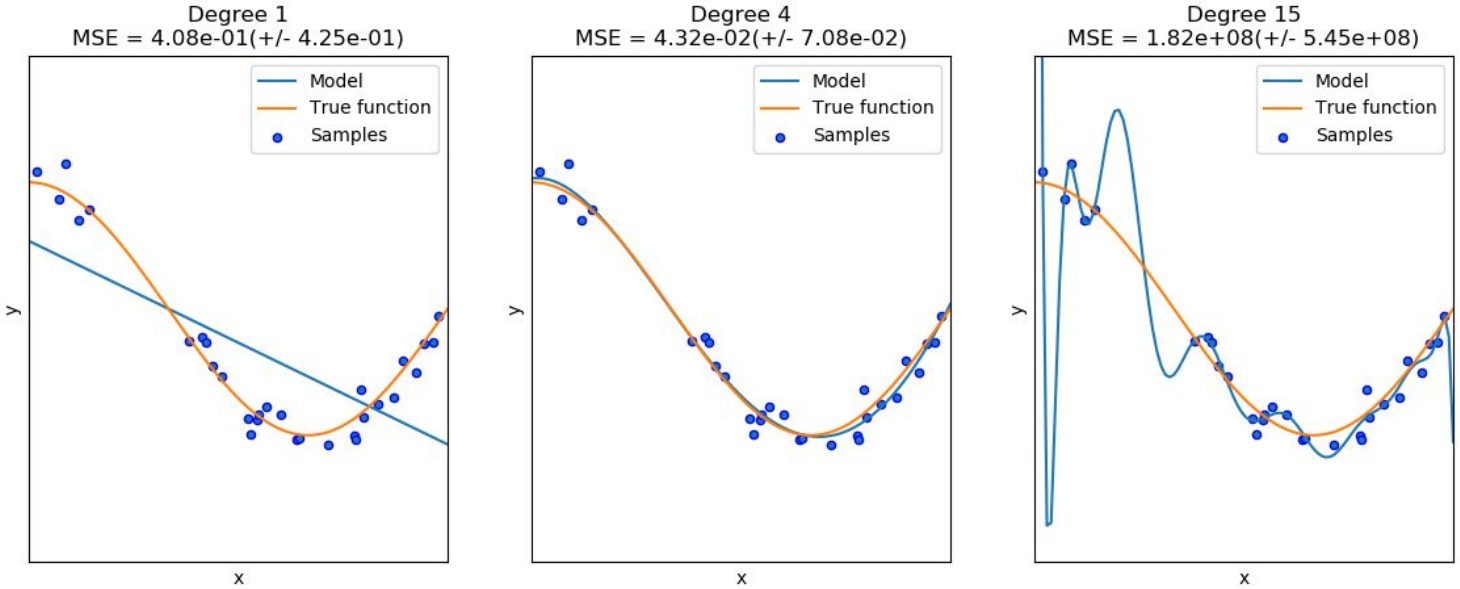


Training Algorithm

1. Initialize model with random parameters.
2. Repeat
   a. Take a small random subset of the dataset that will fit in memory (minibatch).
   b. Forward Pass : pass the subset through the model and obtain predictions
   c. Compute the mean loss function for the subset
   d. Backward Pass: compute the gradients of the parameters, last layer to the first.
   e. Update the gradients using learning rate



Overfitting

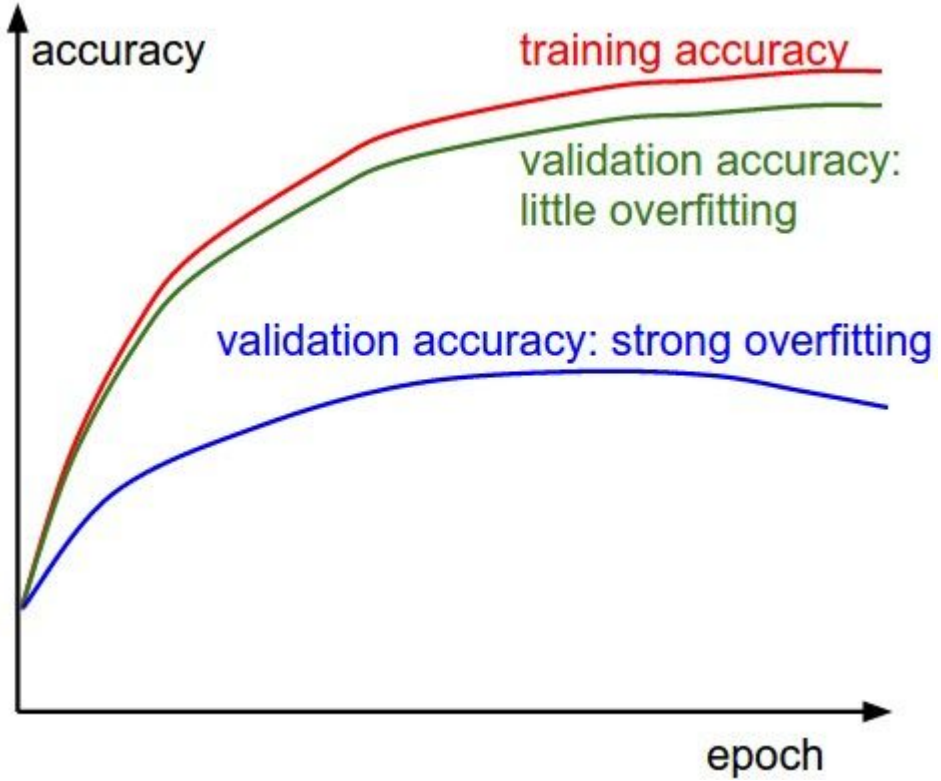


Overfitting during Training

To observe overfitting, we have test data not used for gradient descent.

Accuracy is computed periodically on test data.

If we observe training accuracy and test accuracy diverging, then there is overfitting.



Some References

Must watch video (30 mins)
https://www.youtube.com/watch?v=aircAruvnKk&list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi

Blog (short read)
http://colah.github.io/posts/2015-08-Backprop/
http://www.wildml.com/2015/09/implementing-a-neural-network-from-scratch/
https://ml.berkeley.edu/blog/2016/11/06/tutorial-1/
https://ml.berkeley.edu/blog/2016/12/24/tutorial-2/
https://ml.berkeley.edu/blog/2017/02/04/tutorial-3/

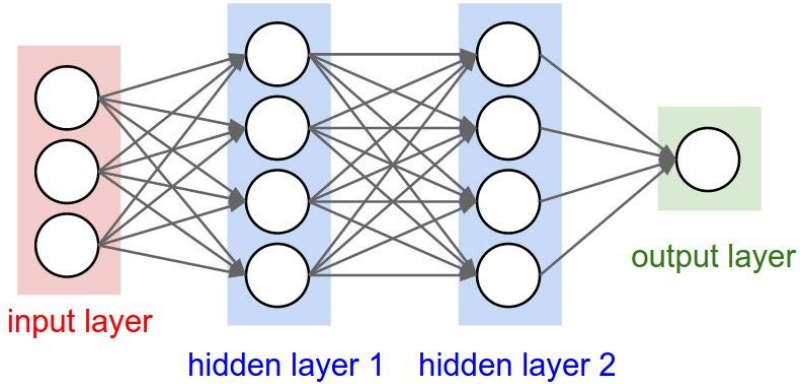
Course Notes: (will require lots of time)
http://cs231n.github.io/ ,
https://www.youtube.com/playlist?list=PL3FW7Lu3i5JvHM8ljYj-zLfQRF3EO8sYv

Deep Learning Theory II

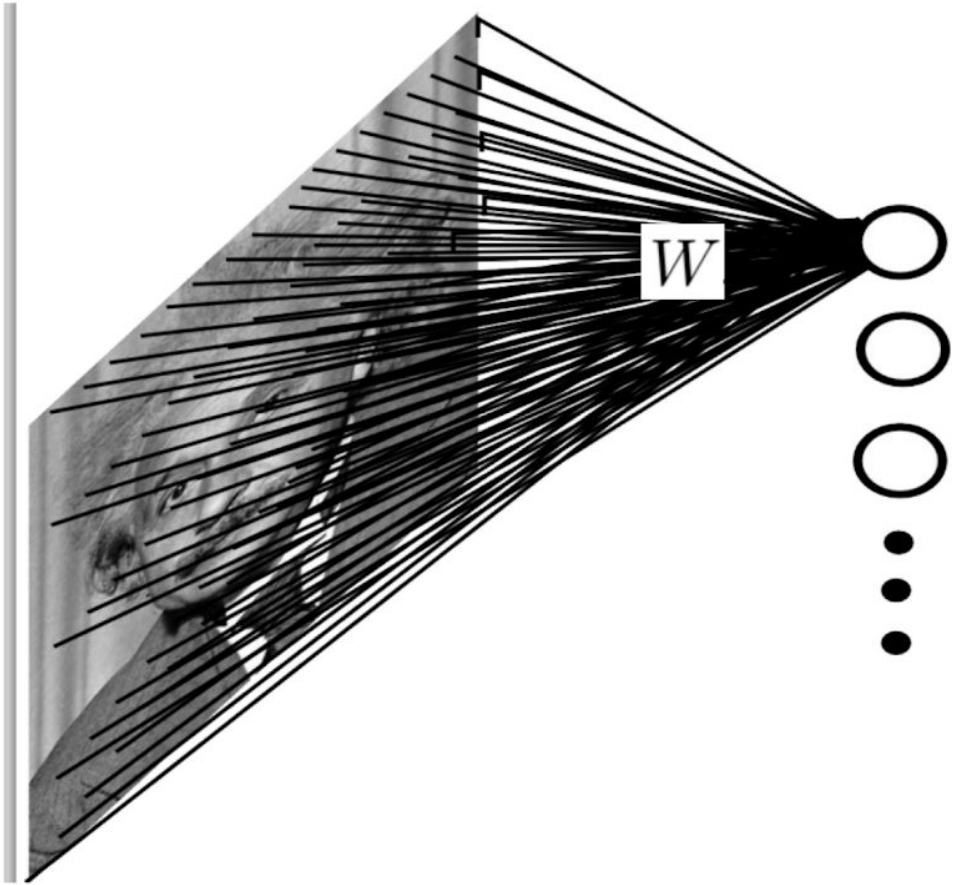
Girish Varma
IIIT Hyderabad

Recap

- Dataset: (x,y) pairs
  - Train, Test splits
- Model
  - Linear Model
  - MLP Model
  - Softmax Function for prediction probabilities
- Gradient Decent/Backpropagation
- Overfitting

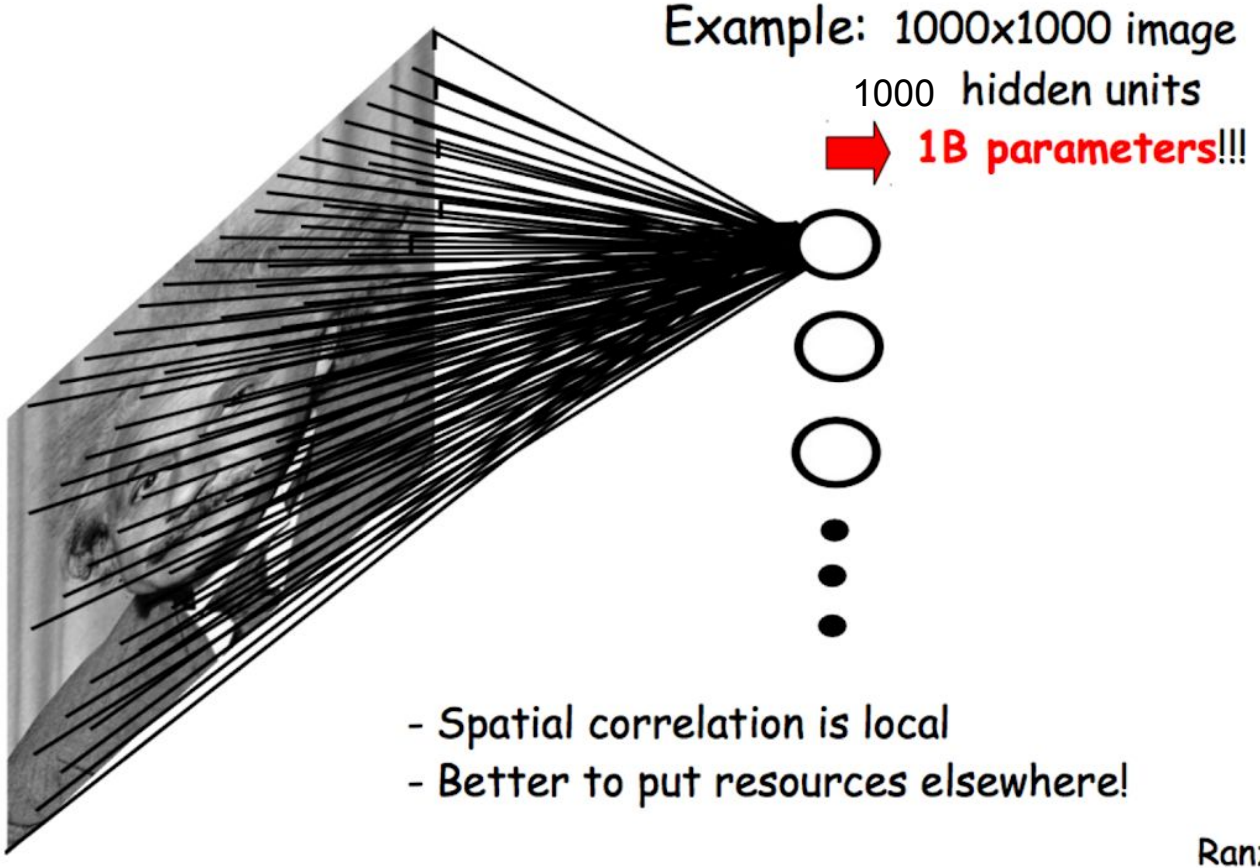


When the input data is an image..

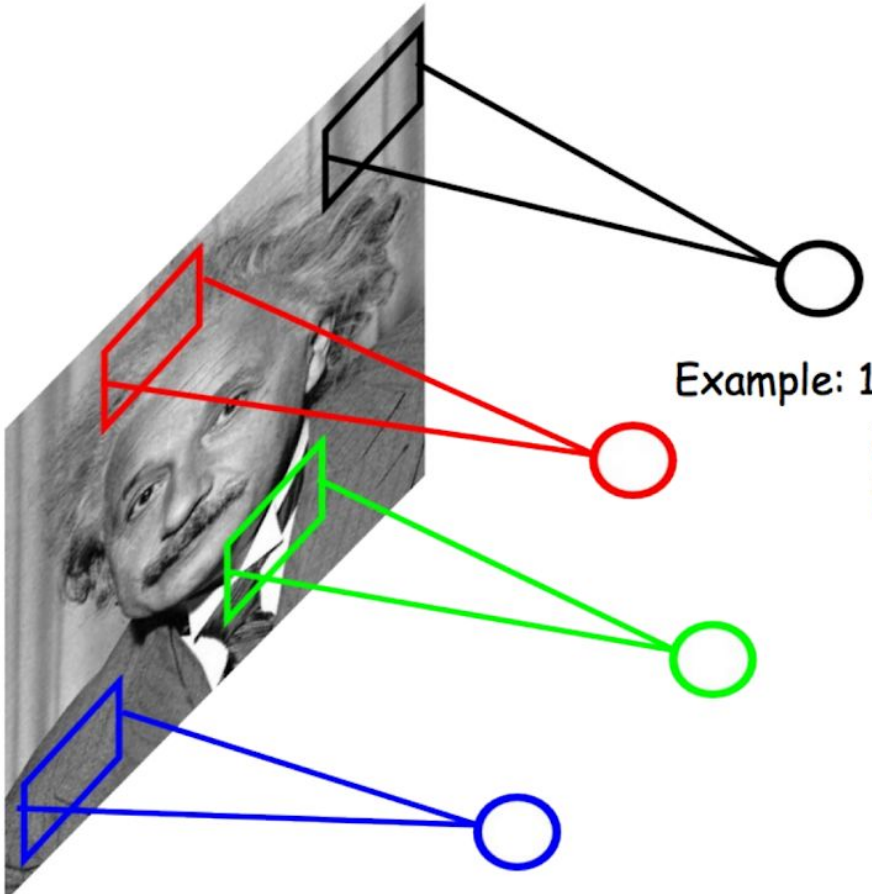


Slide: M. Ranzato

When the input data is an image..

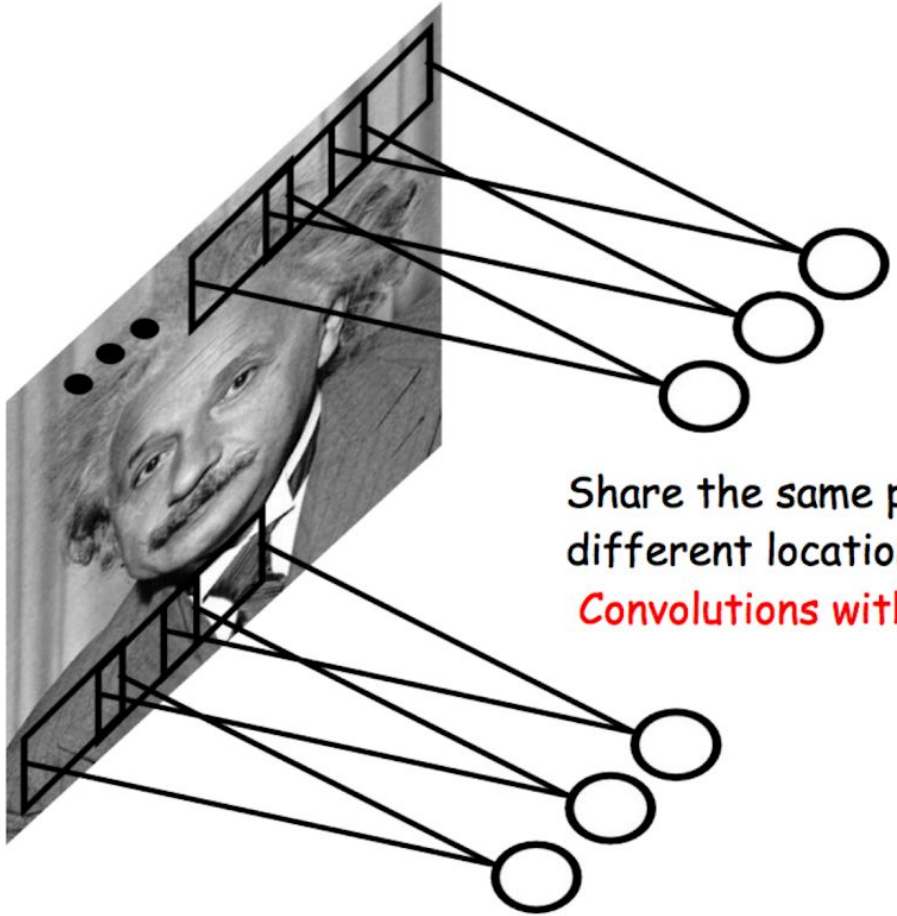


Reduce connection to local regions



Example: 1000x1000 image
1M hidden units
Filter size: 10x10
10M parameters

Reuse the same kernel everywhere

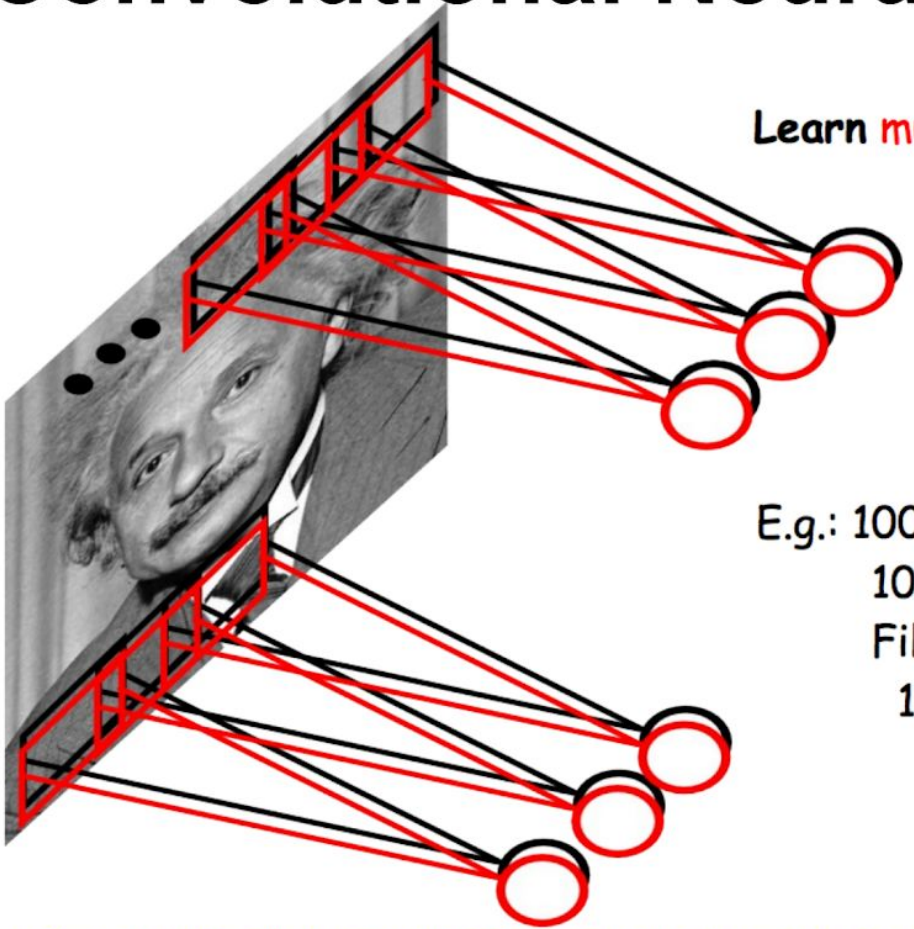


Because interesting features (edges) can happen at anywhere in the image.

Share the same parameters across different locations:

Convolutions with learned kernels

Convolutional Neural Nets



Learn multiple filters.

E.g.: 1000x1000 image
100 Filters
Filter size: 10x10
10K parameters

Ranzato

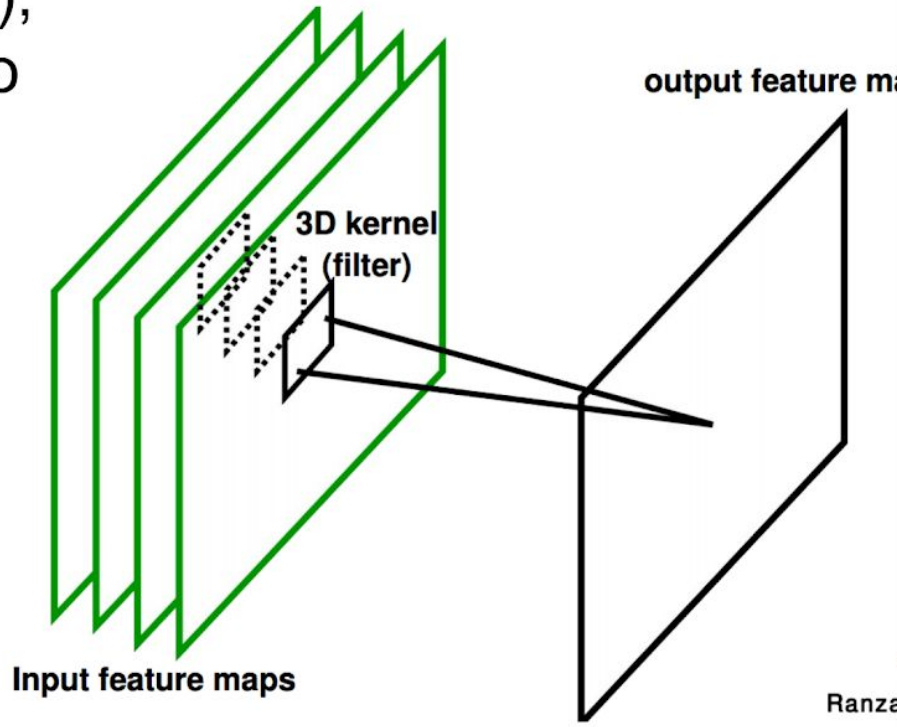
LeCun et al. “Gradient-based learning applied to document recognition” IEEE 1998

Detail

If the input has 3 channels (R,G,B), 3 separate k by k filter is applied to each channel.

Output of convolving 1 feature is called a *feature map*.

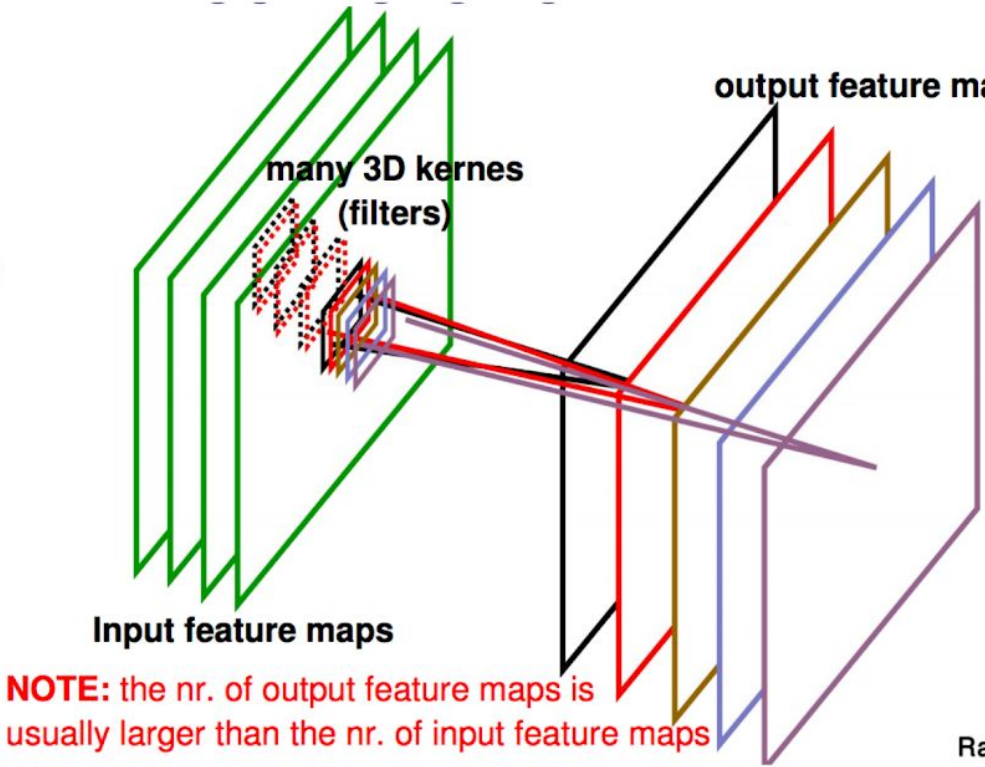
This is just sliding window, ex. the output of one part filter of DPM is a feature map



Using multiple filters

Each filter detects features in the output of previous layer.

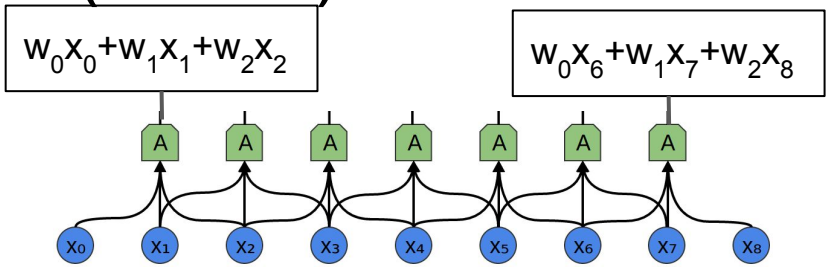
So to capture different features, learn multiple filters.



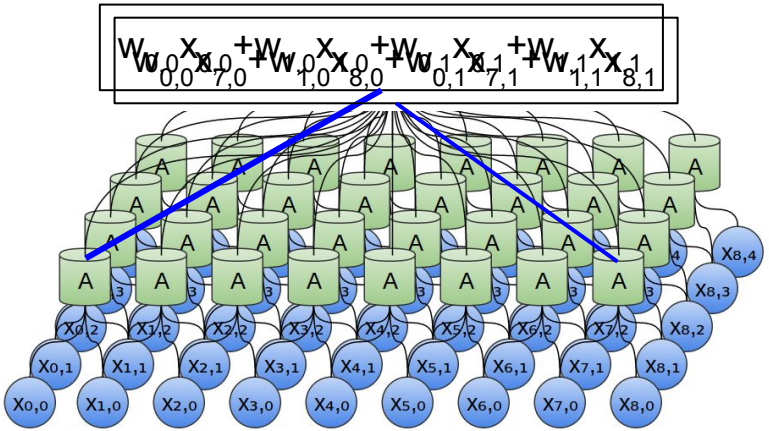
Convolutional Neural Network (CNNs)

For 1D Sequences

weight sharing



For 2D Sequences (Images)

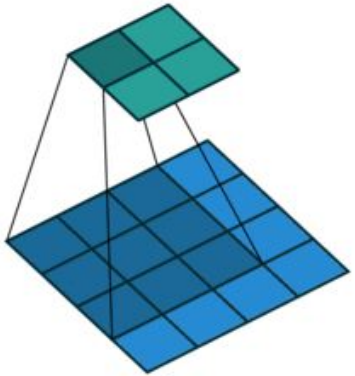


CNNs

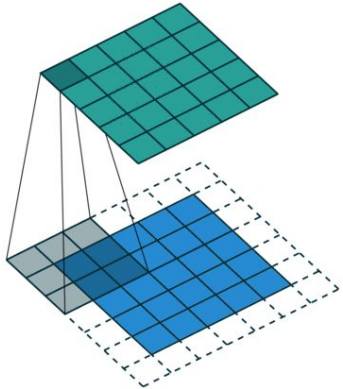
Window size

Stride

Padding



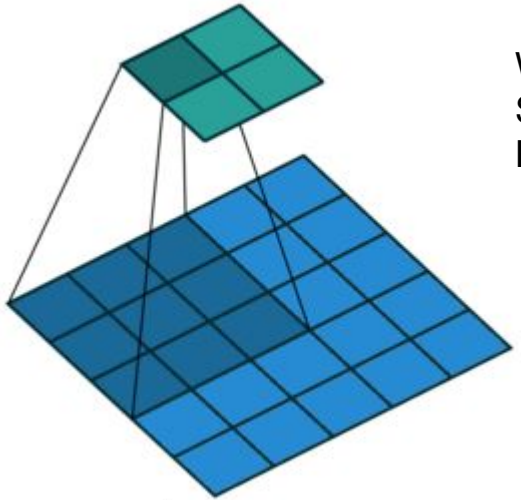
Window size: 3x3
Stride: 1
Padding: 0



Window size: 3x3
Stride: 1
Padding: 1

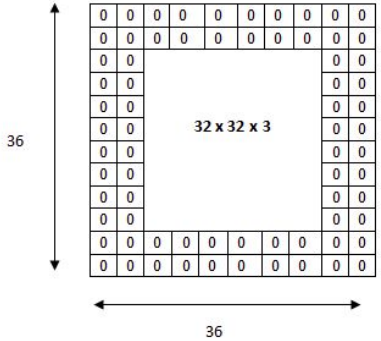
CNNs

Strides reduces dimension

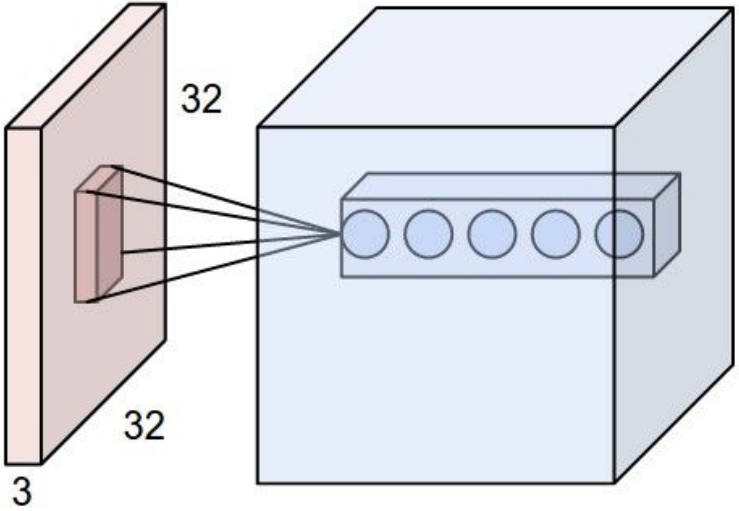


Window size: 3x3
Stride: 2
Padding: 0

$$O = \frac{(W - K + 2P)}{S} + 1$$



Input, Output Channels : Multiple Filters



DEMO: http://cs231n.github.io/convolutional-networks/

Complexity: Parameters

Window Size: F x F

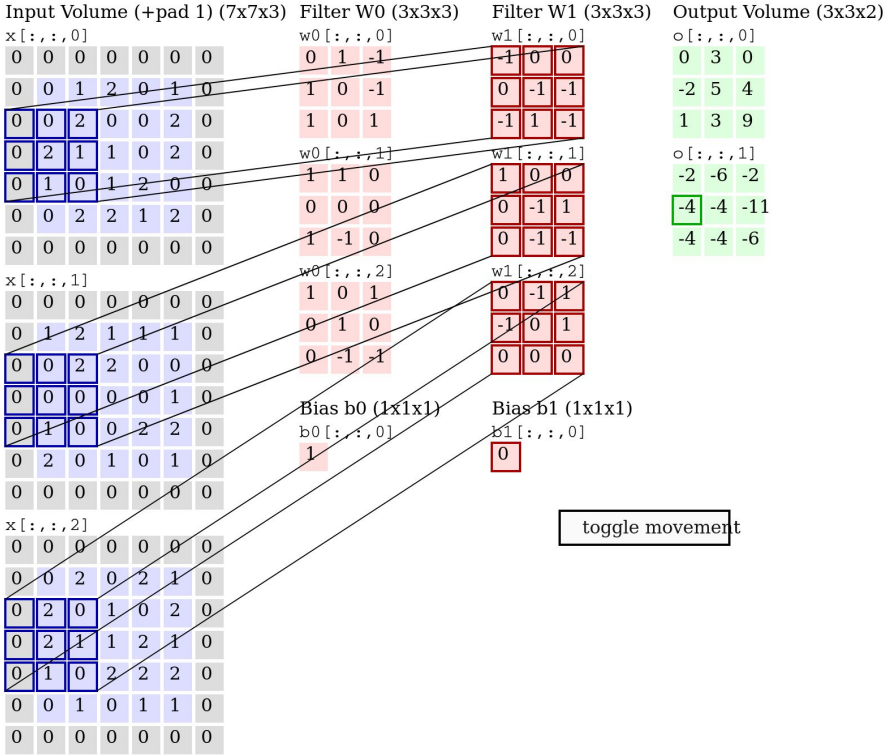
Input Channels: D_1

No. of Filters(Out Channels): K

No. of Parameters =

$F*F*D_1*K$

For every pair of input/output channels ther



Complexity: FLOPs

Floating Point Operations

If a FxF convolutions converts an input of size H_1*W_1*D_1 to output of size is H_2 x W_2xD_2, then FLOPs is

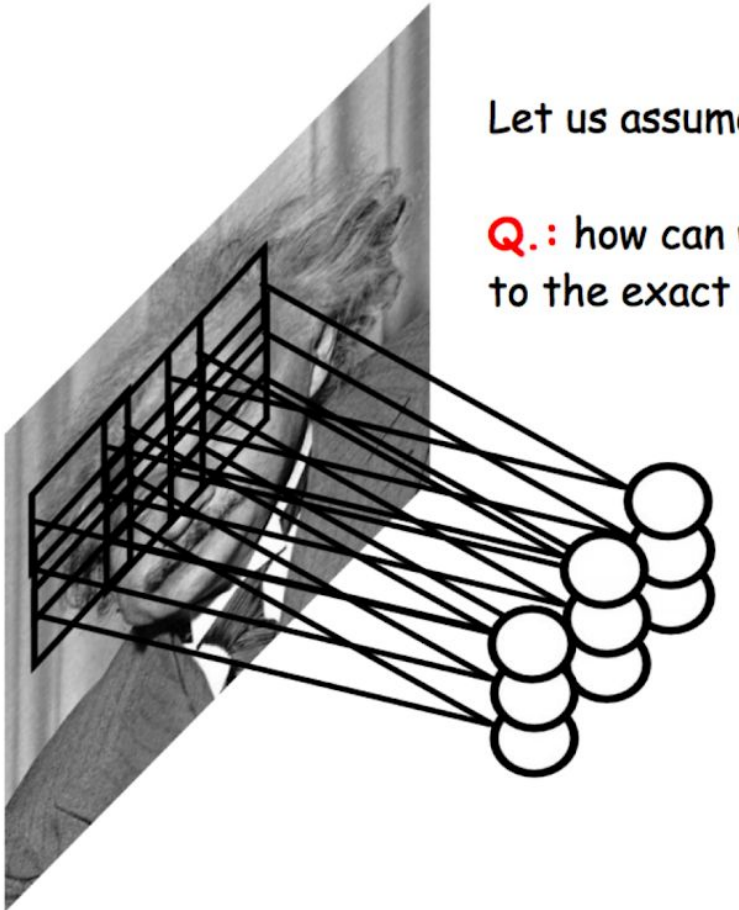
$$H_2*W_2*F*F*D_1*D_2$$

CNN Summary

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

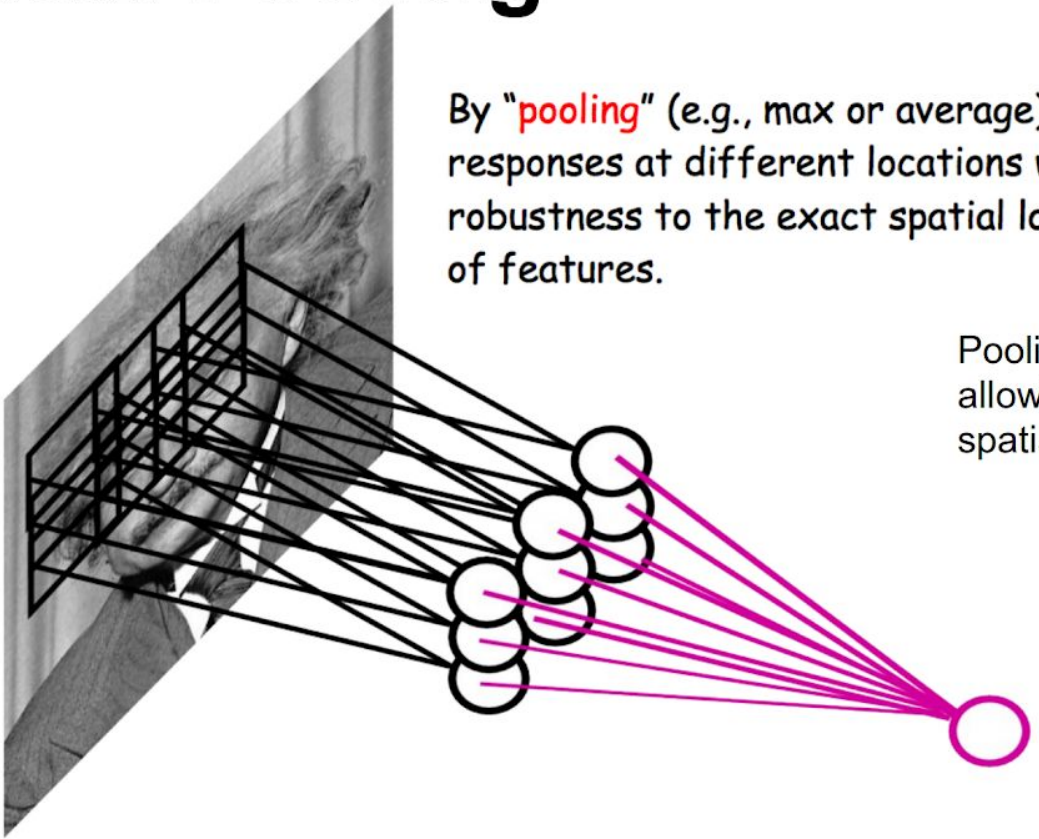
Building Translation Invariance



Let us assume filter is an "eye" detector.

Q.: how can we make the detection robust to the exact location of the eye?

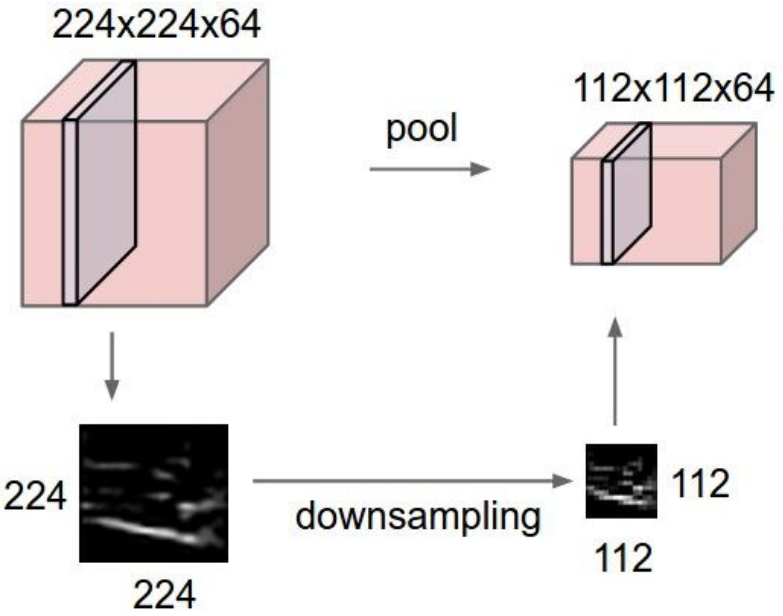
Building Translation Invariance via Spatial Pooling

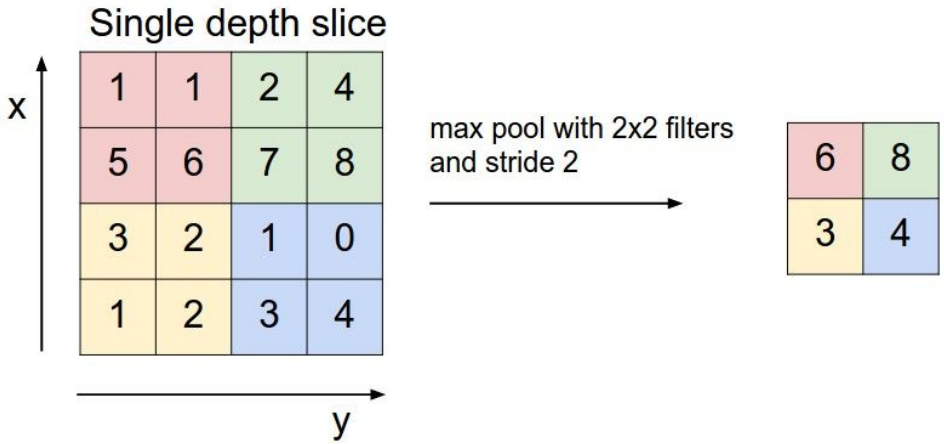


By "pooling" (e.g., max or average) filter responses at different locations we gain robustness to the exact spatial location of features.

Pooling also subsamples the image, allowing the next layer to look at larger spatial regions.

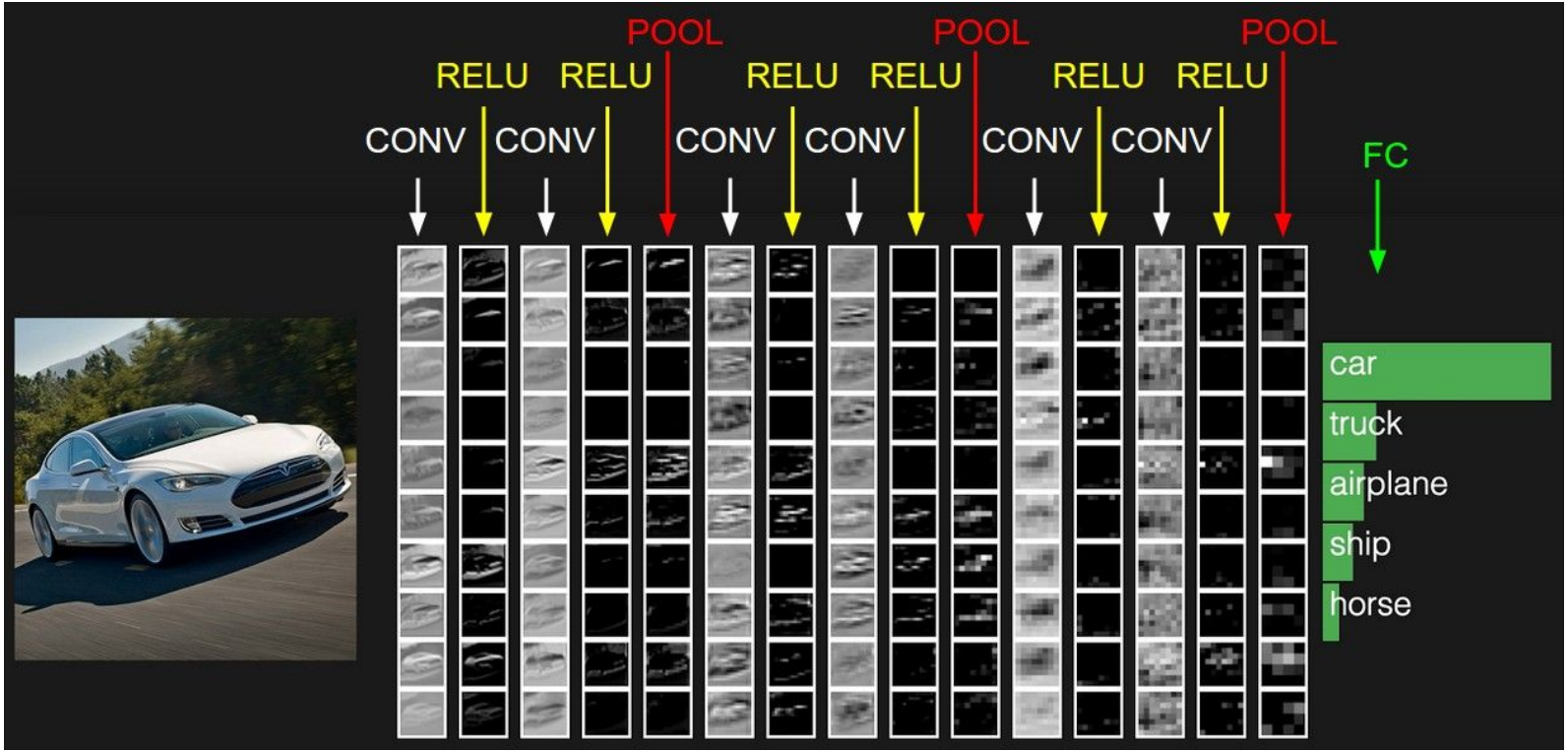
Pooling



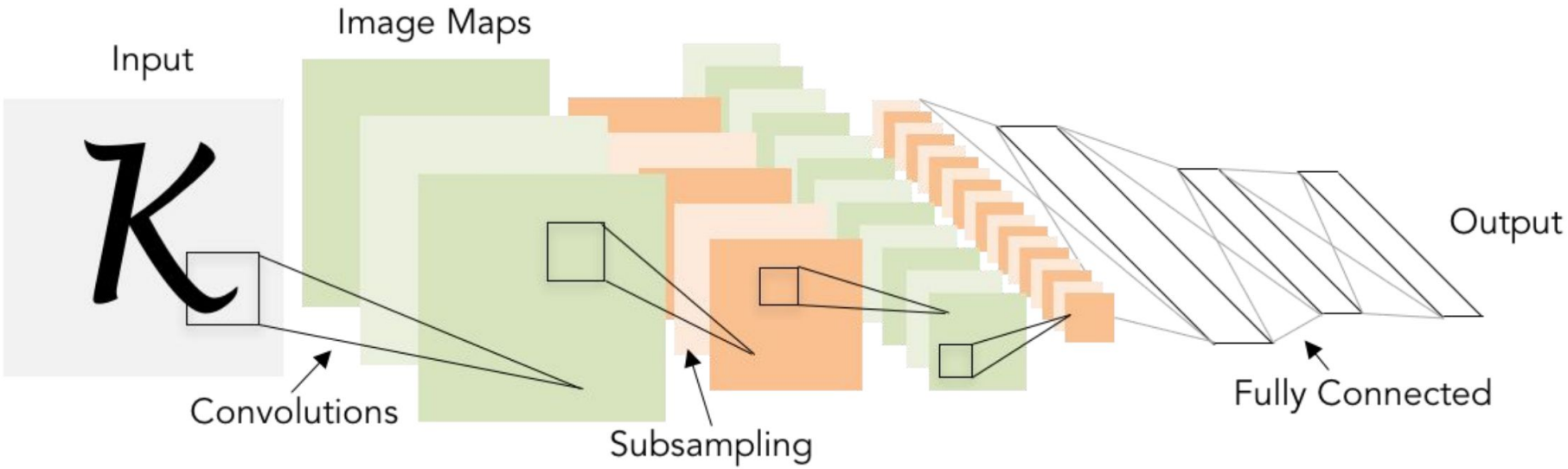


No Learnable Parameters

Deep CNN


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

Review: LeNet

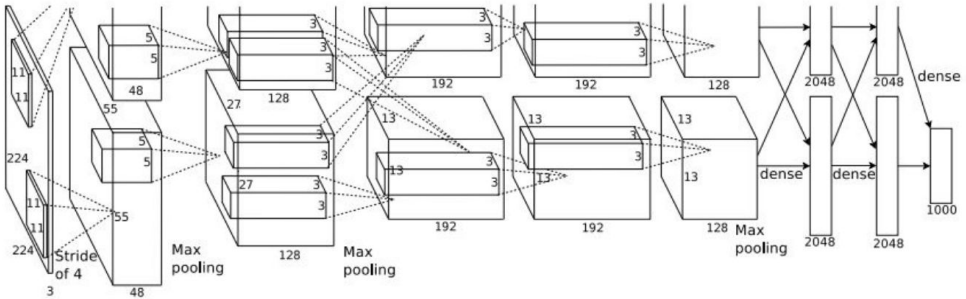


Conv filters were 5x5, applied at stride 1
Subsampling (Pooling) layers were 2x2 applied at stride 2
i.e. architecture is [CONV-POOL-CONV-POOL-FC-FC]

CNN Architectures

Case Study: AlexNet

[Krizhevsky et al. 2012]

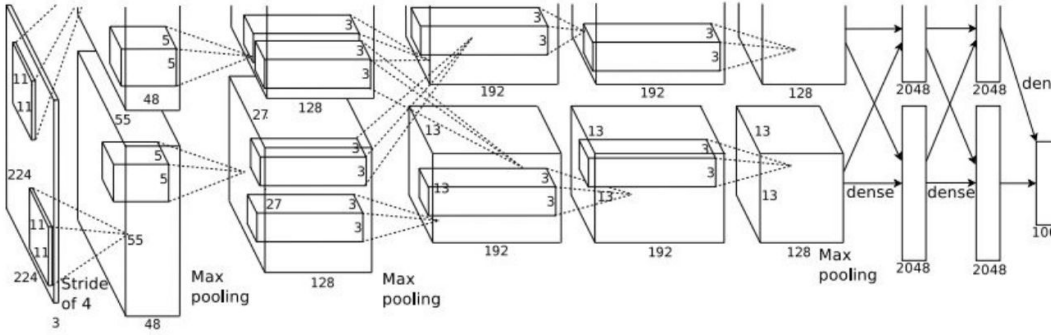


Architecture:

CONV1
MAX POOL1
NORM1
CONV2
MAX POOL2
NORM2
CONV3
CONV4
CONV5
Max POOL3
FC6
FC7
FC8

CNN Architectures
Case Study: AlexNet

[Krizhevsky et al. 2012]



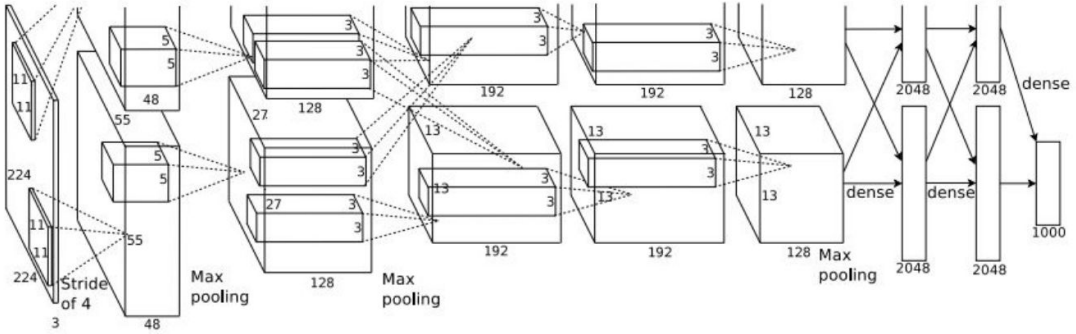
Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>

Q: what is the output volume size? Hint: (227-11)/4+1 = 55

CNN Architectures
Case Study: AlexNet

[Krizhevsky et al. 2012]



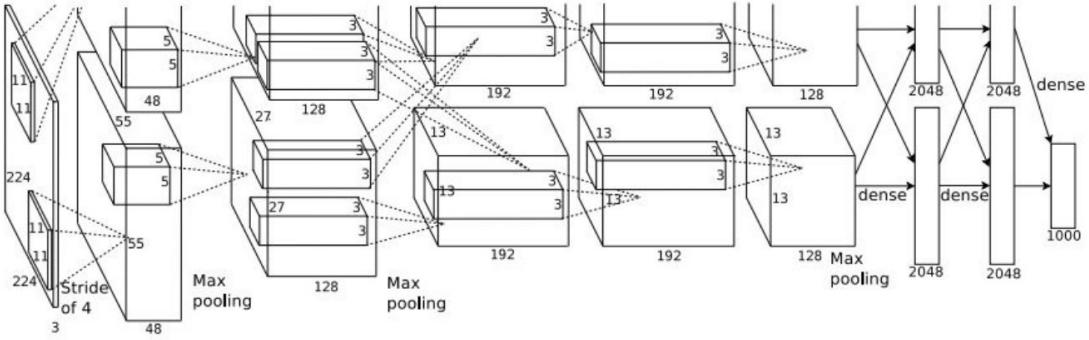
Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**

Q: What is the total number of parameters in this layer?

CNN Architectures
Case Study: AlexNet

[Krizhevsky et al. 2012]



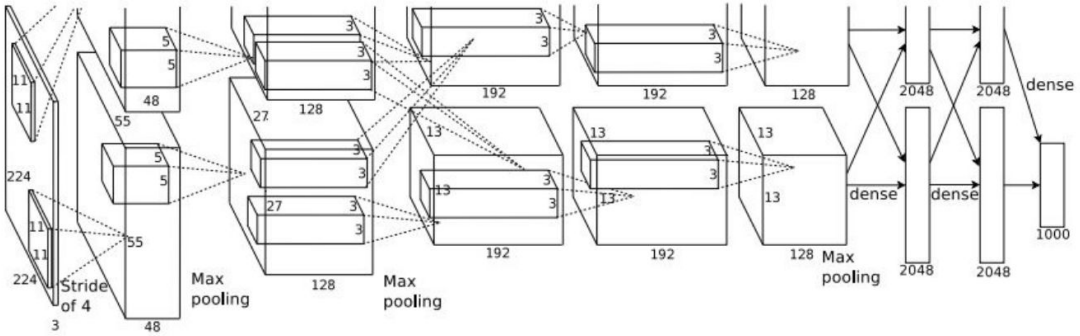
Input: 227x227x3 images

First layer (CONV1): 96 11x11 filters applied at stride 4
=>
Output volume **[55x55x96]**
Parameters: (11*11*3)*96 = **35K**

CNN Architectures

Case Study: AlexNet

[Krizhevsky et al. 2012]



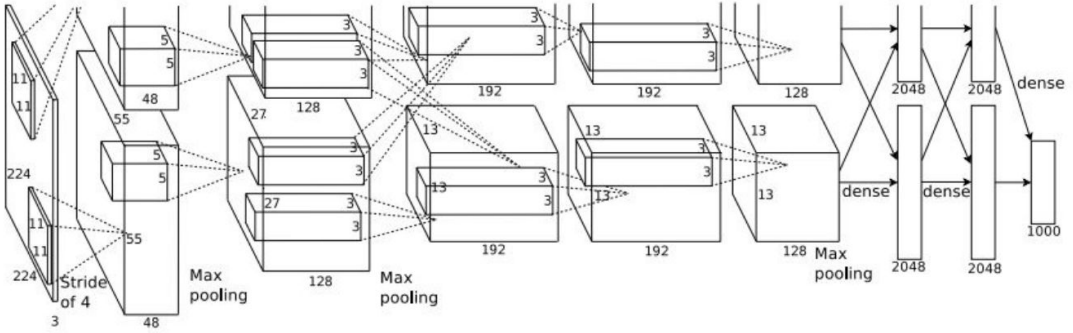
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2

Q: what is the output volume size? Hint: (55-3)/2+1 = 27

CNN Architectures
Case Study: AlexNet

[Krizhevsky et al. 2012]



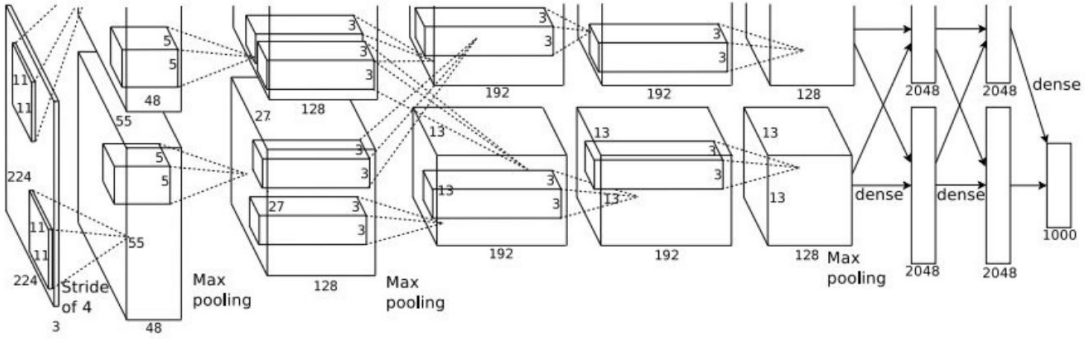
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96

Q: what is the number of parameters in this layer?

CNN Architectures
Case Study: AlexNet

[Krizhevsky et al. 2012]



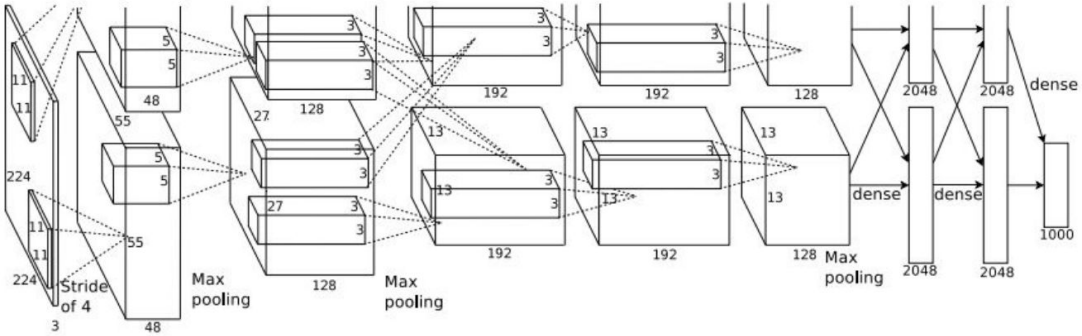
Input: 227x227x3 images
After CONV1: 55x55x96

Second layer (POOL1): 3x3 filters applied at stride 2
Output volume: 27x27x96
Parameters: 0!

CNN Architectures

Case Study: AlexNet

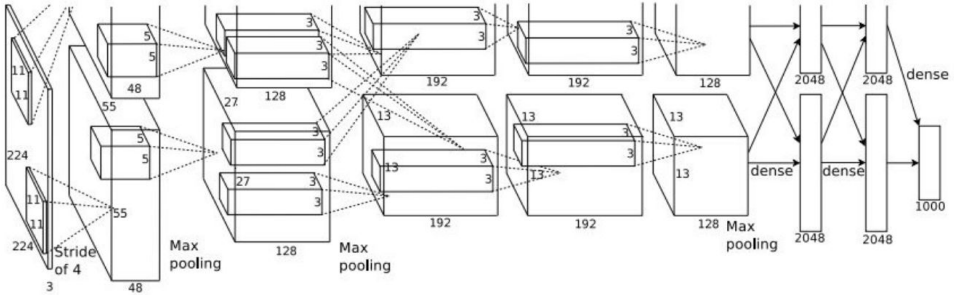
[Krizhevsky et al. 2012]



Input: 227x227x3 images
After CONV1: 55x55x96
After POOL1: 27x27x96
...

CNN Architectures
Case Study: AlexNet

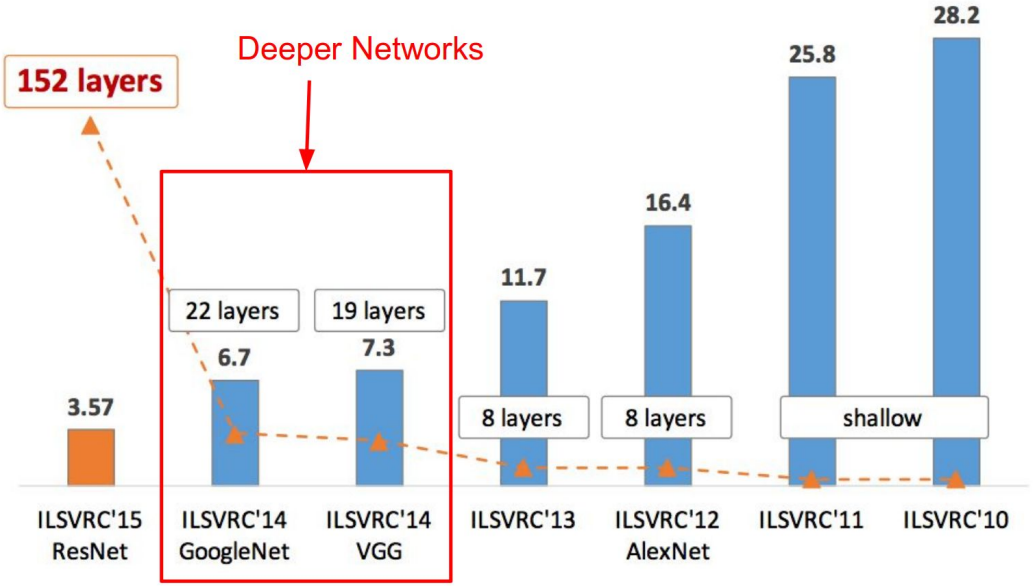
[Krizhevsky et al. 2012]



Full (simplified) AlexNet architecture:
[227x227x3] INPUT
[55x55x96] CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96] MAX POOL1: 3x3 filters at stride 2
[27x27x96] NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256] MAX POOL3: 3x3 filters at stride 2
[4096] FC6: 4096 neurons
[4096] FC7: 4096 neurons
[1000] FC8: 1000 neurons (class scores)

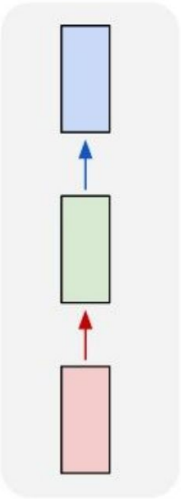
Getting Deeper

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) winners



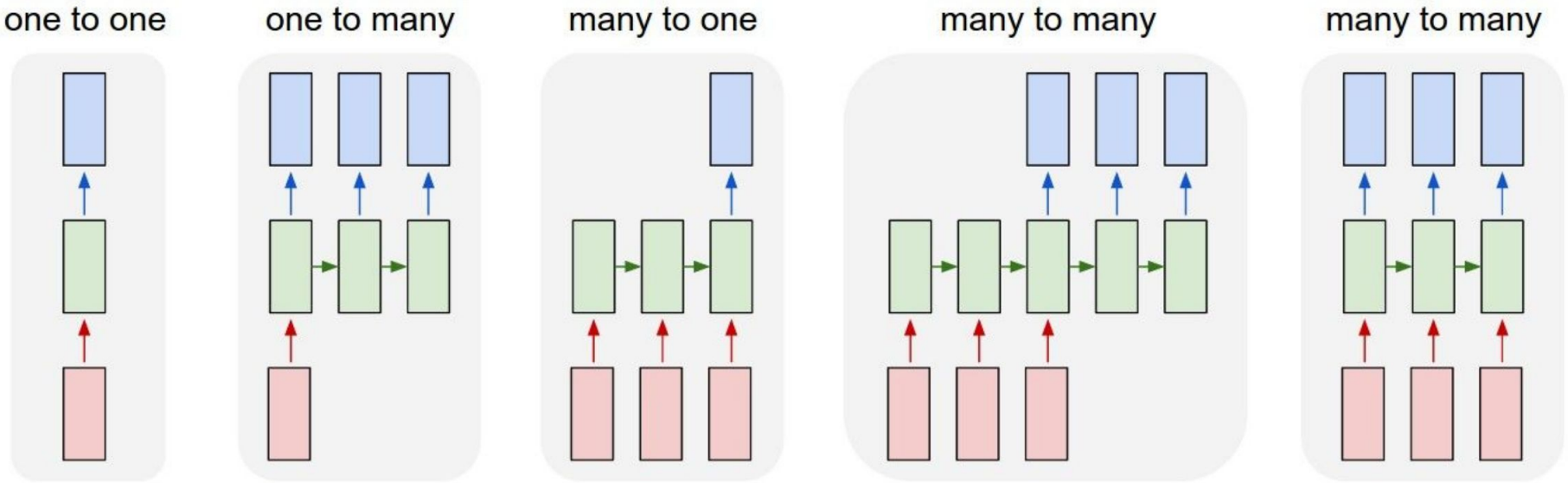
Recurrent Neural Networks (RNNs)

one to one



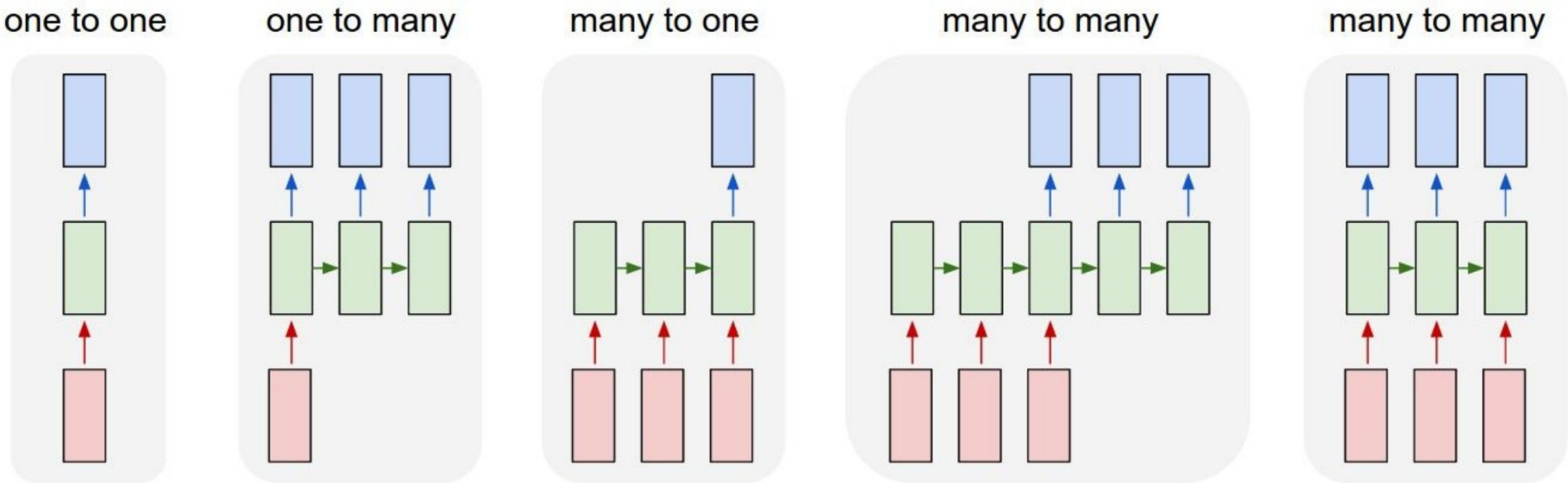
Vanilla Neural Networks

RNNs : Process Sequence



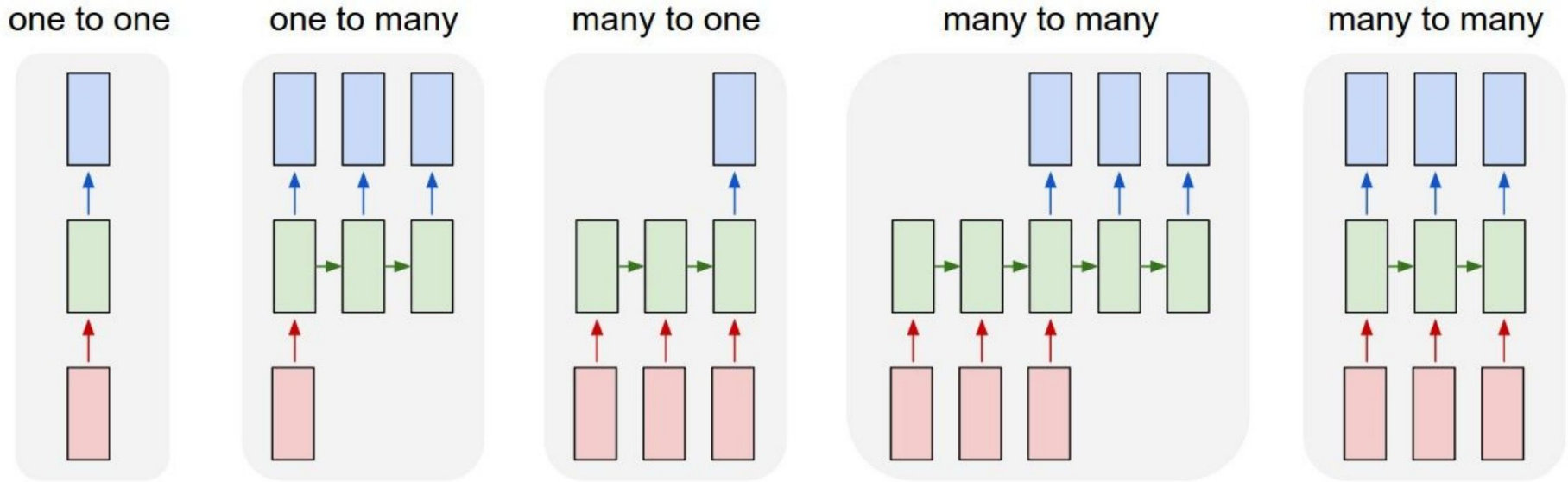
e.g. **Image Captioning**
image -> sequence of words

RNNs : Process Sequence



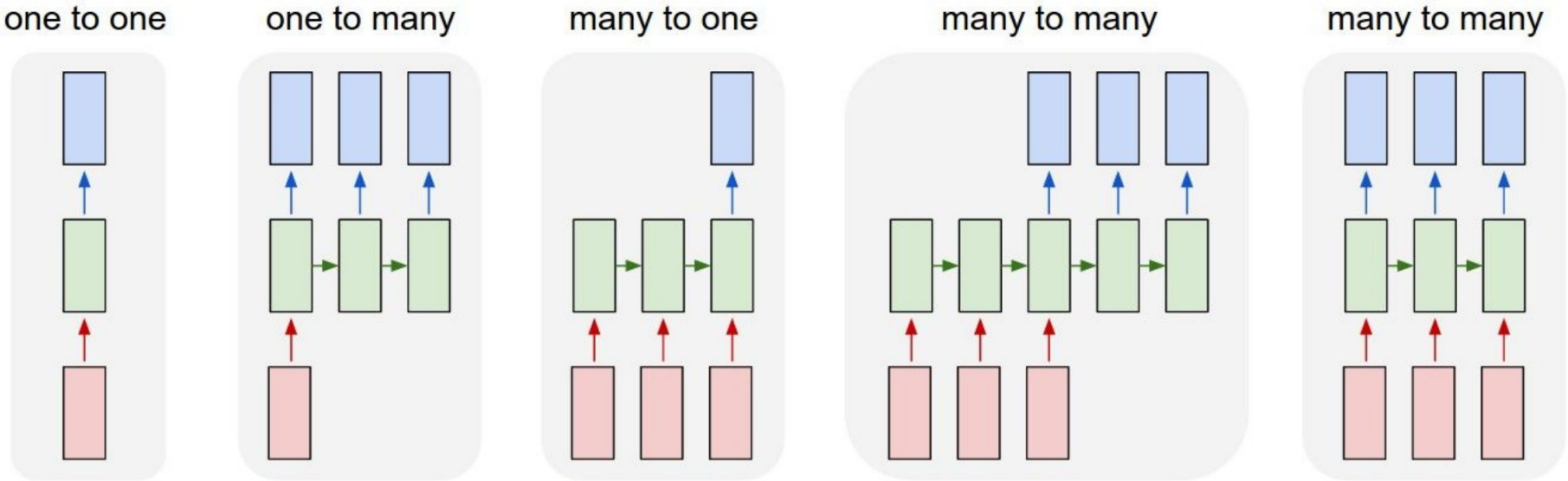
e.g. **Sentiment Classification**
sequence of words -> sentiment

RNNs : Process Sequence



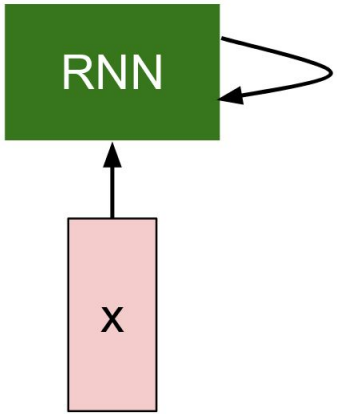
e.g. **Machine Translation**
seq of words -> seq of words

RNNs : Process Sequence

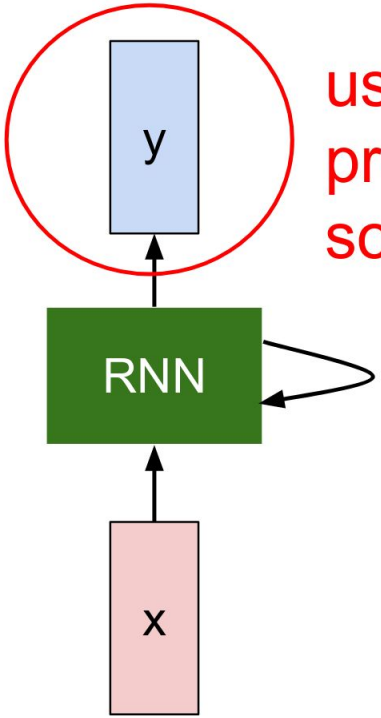


e.g. **Video classification on frame level**

RNNs



RNNs



usually want to predict a vector at some time steps

RNNs

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

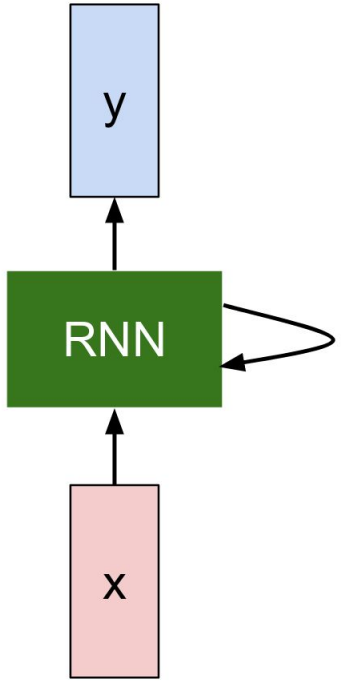
$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

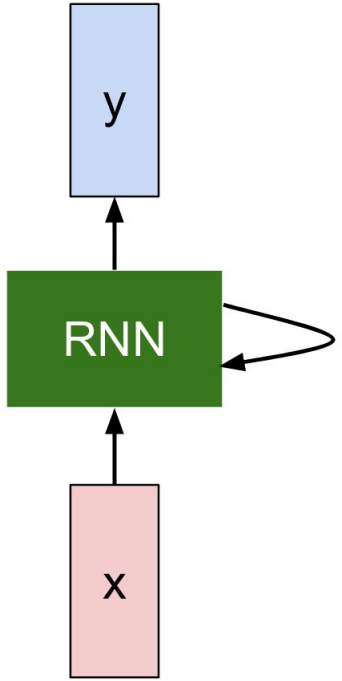
some function with parameters W



RNNs

We can process a sequence of vectors **x** by applying a **recurrence formula** at every time step:

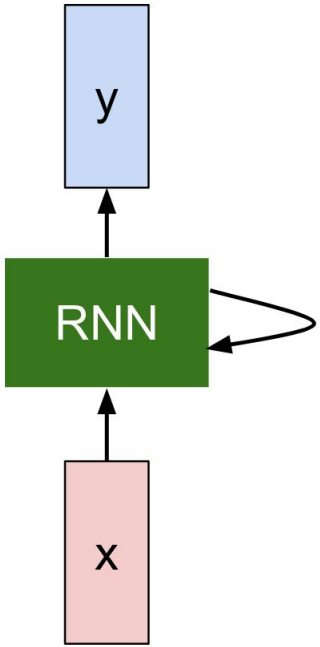
$$h_t = f_W(h_{t-1}, x_t)$$



Notice: the same function and the same set of parameters are used at every time step.

RNNs

The state consists of a single *"hidden"* vector **h**:

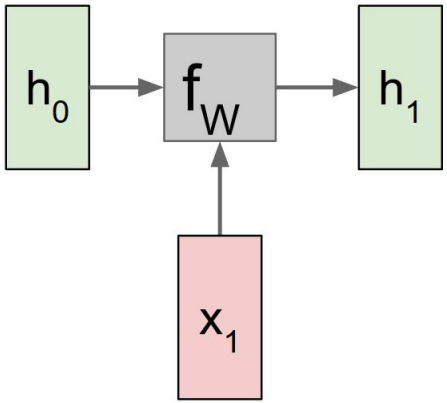


$$h_t = f_W(h_{t-1}, x_t)$$

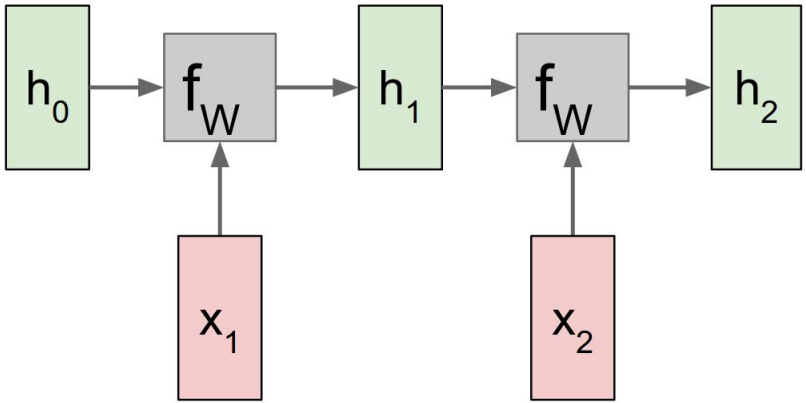
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

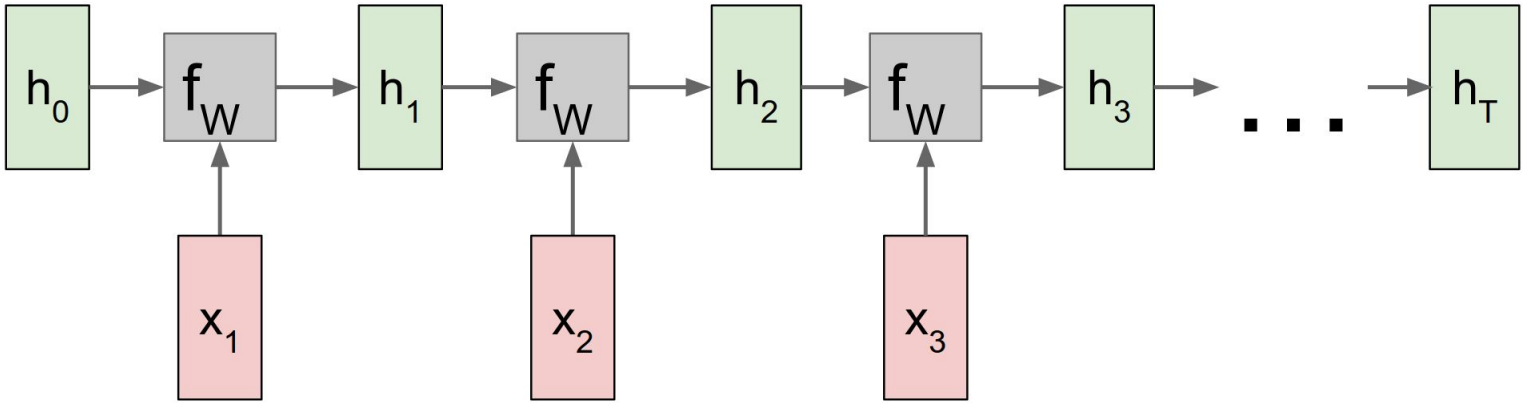
RNN : Computation Graph



RNN : Computation Graph

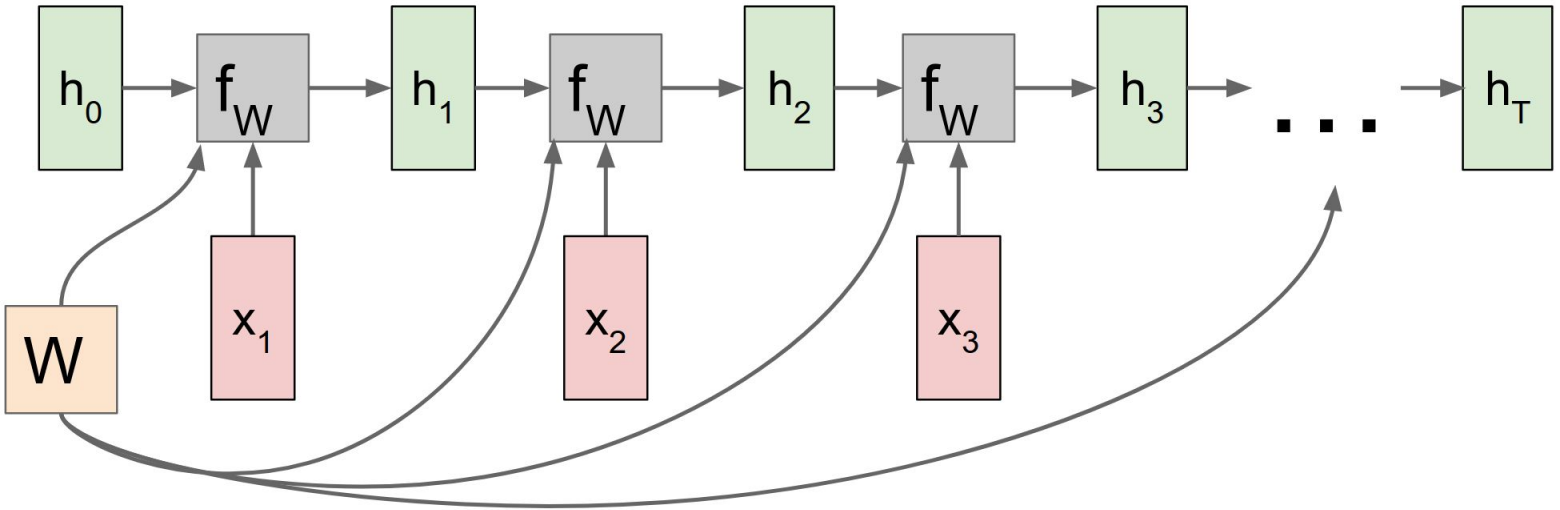


RNN : Computation Graph

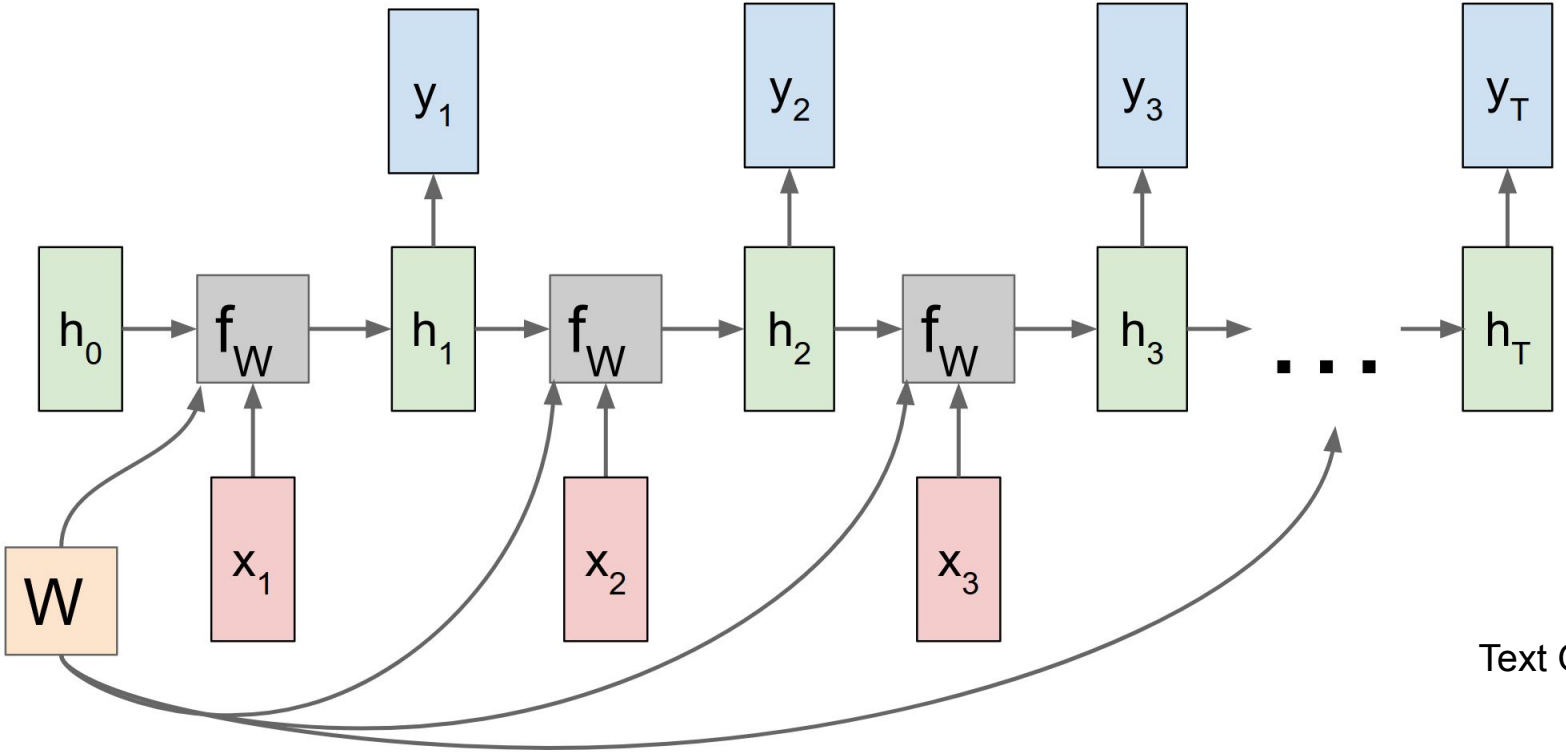


RNN : Computation Graph

Re-use the same weight matrix at every time-step

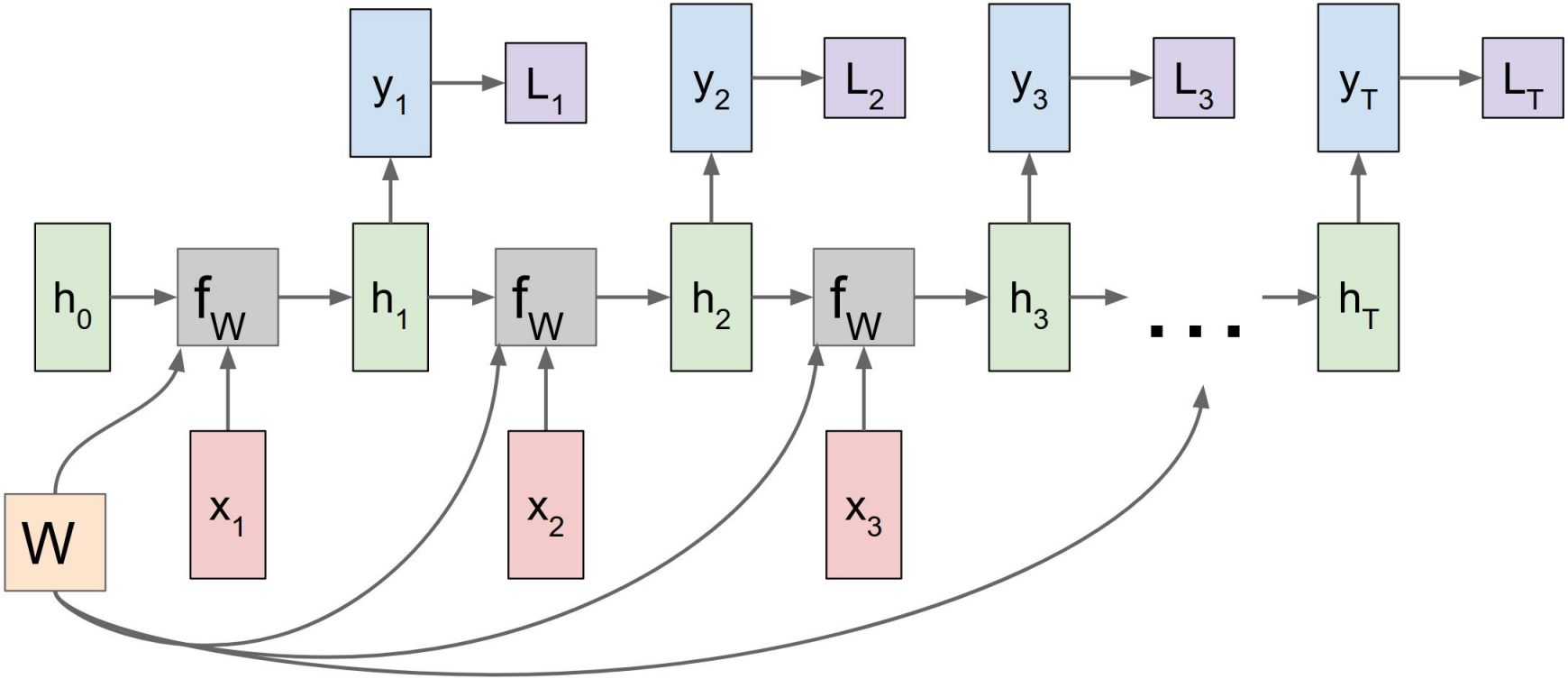


RNN: Many to Many

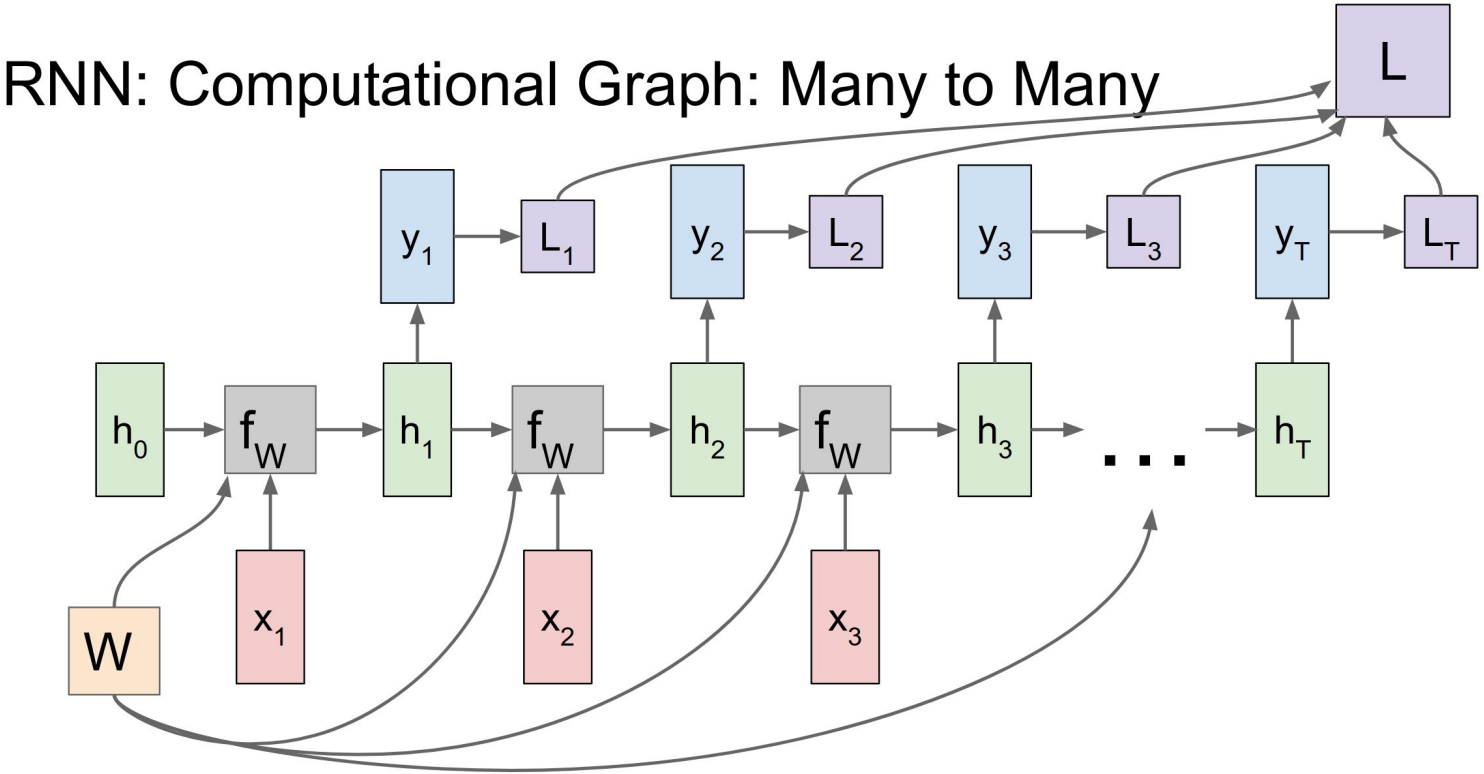


Text Generation

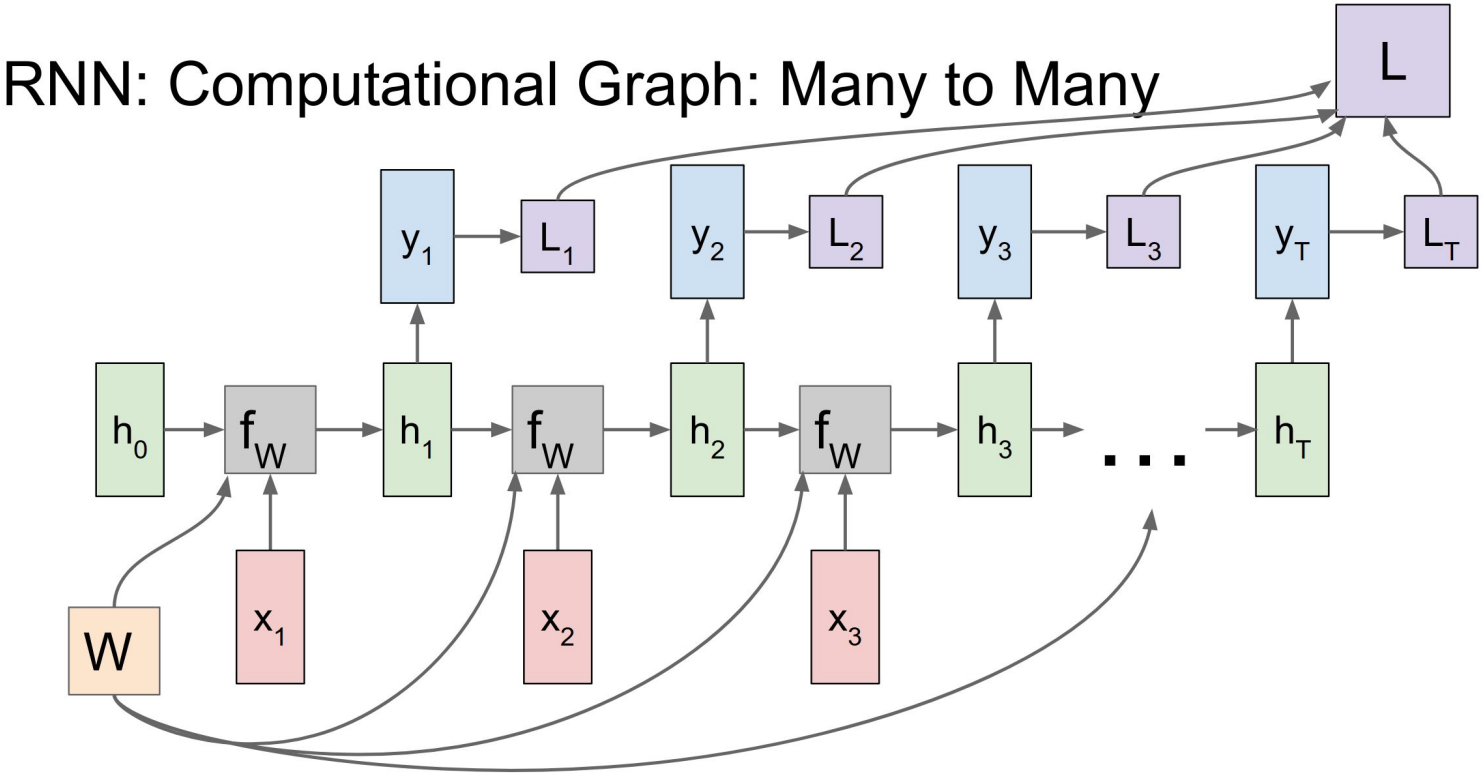
Loss Function for Many to Many



Loss Function for Many to Many

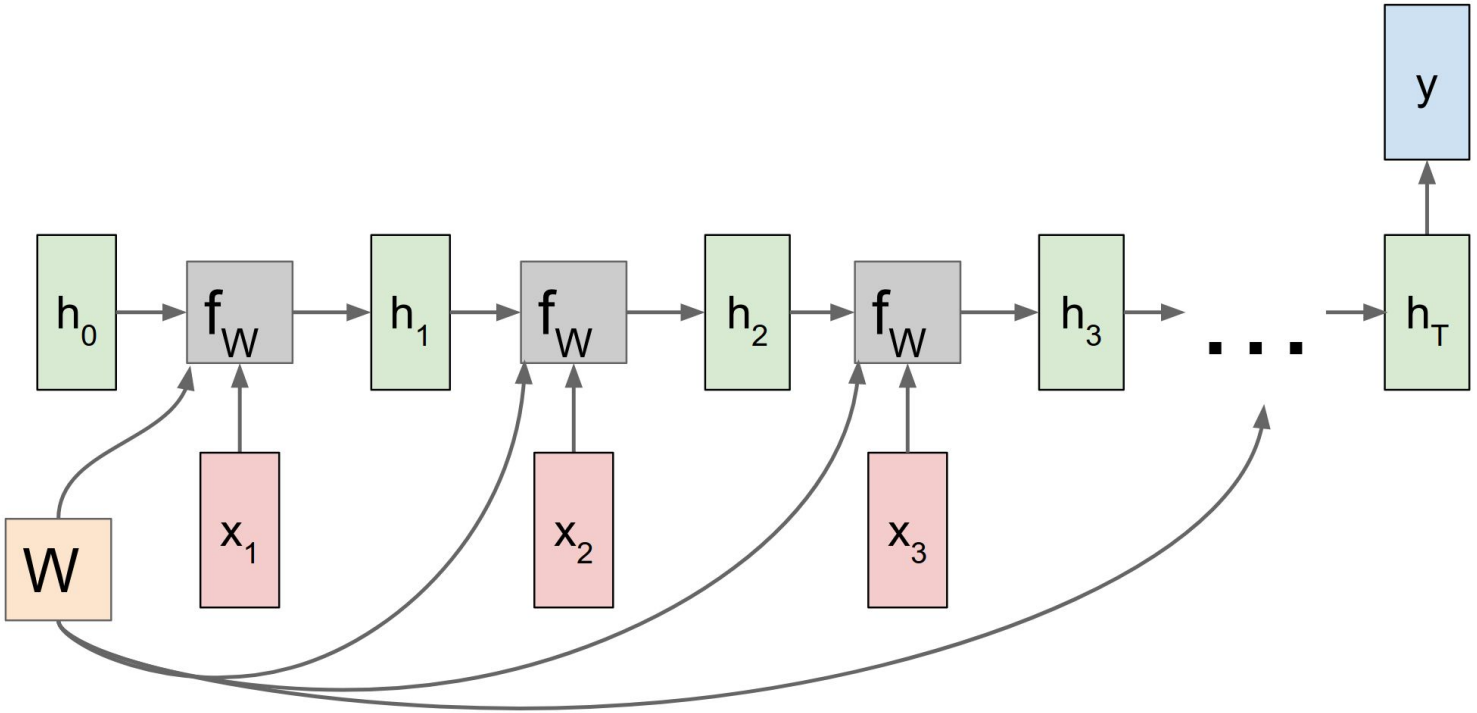


Loss Function for Many to Many



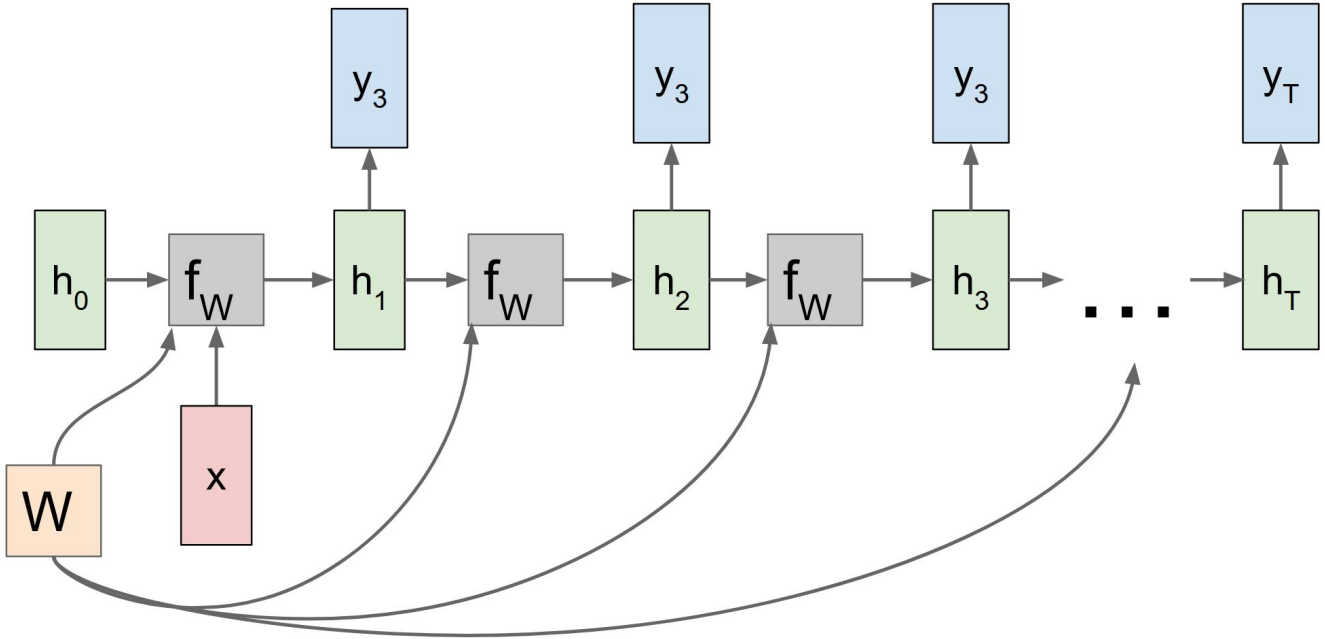
Many to One (Sentiment Analysis)

RNN: Computational Graph: Many to One



One to Many (Image Captioning)

RNN: Computational Graph: One to Many



Summary

- CNNs
  a. Window size, Stride, Padding
  b. Input/Output channels
  c. Parameters and FLOPs
  d. Pooling
- CNN Architectures (LeNet, AlexNet)
- RNN
  a. Definition Formulas
  b. Different Modes of Operation

References

- http://cs231n.github.io/convolutional-networks/
- http://colah.github.io/posts/2014-07-Conv-Nets-Modular/
-

GPU and Deep Learning

CPU vs GPU

GPU and Deep Learning

My computer

Spot the CPU!

(central processing unit)

Spot the GPUs!

(graphics processing unit)

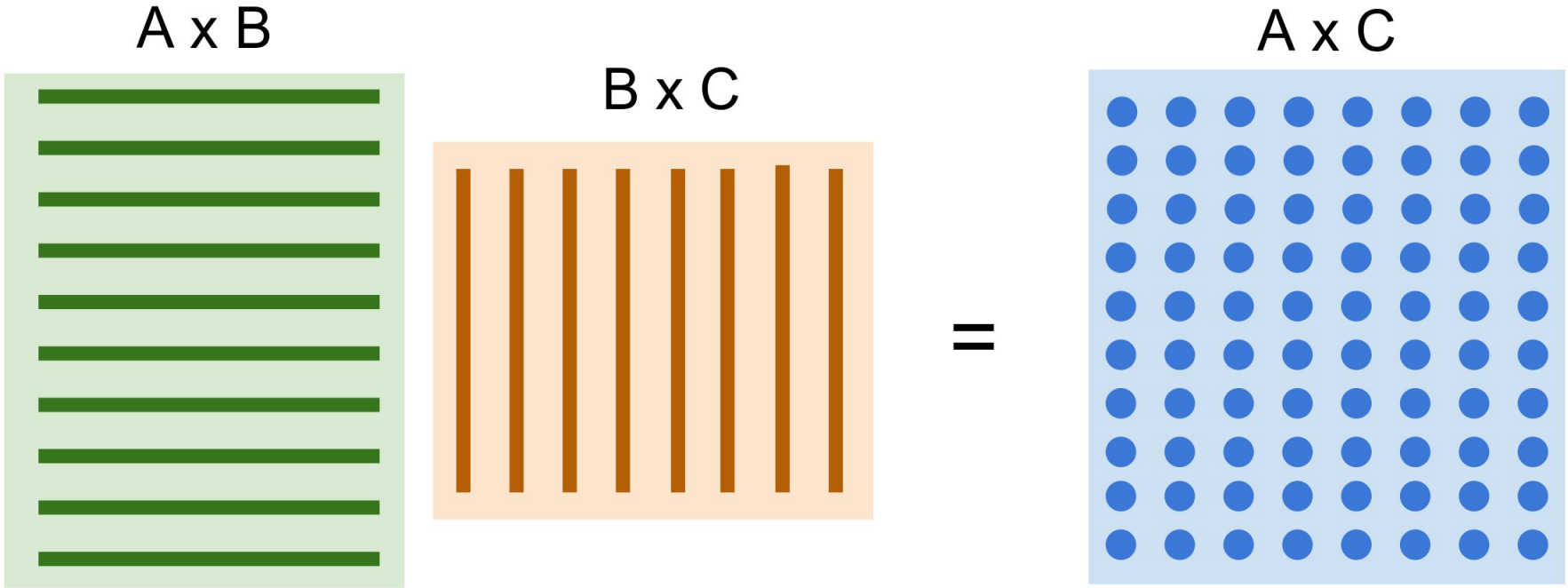
GPU and Deep Learning
CPU vs GPU

	# Cores	Clock Speed	Memory	Price
CPU (Intel Core i7-7700k)	4 (8 threads with hyperthreading)	4.4 GHz	Shared with system	$339
CPU (Intel Core i7-6950X)	10 (20 threads with hyperthreading)	3.5 GHz	Shared with system	$1723
GPU (NVIDIA Titan Xp)	3840	1.6 GHz	12 GB GDDR5X	$1200
GPU (NVIDIA GTX 1070)	1920	1.68 GHz	8 GB GDDR5	$399

CPU: Fewer cores, but each core is much faster and much more capable; great at sequential tasks

GPU: More cores, but each core is much slower and “dumber”; great for parallel tasks

Example: Matrix Multiplication



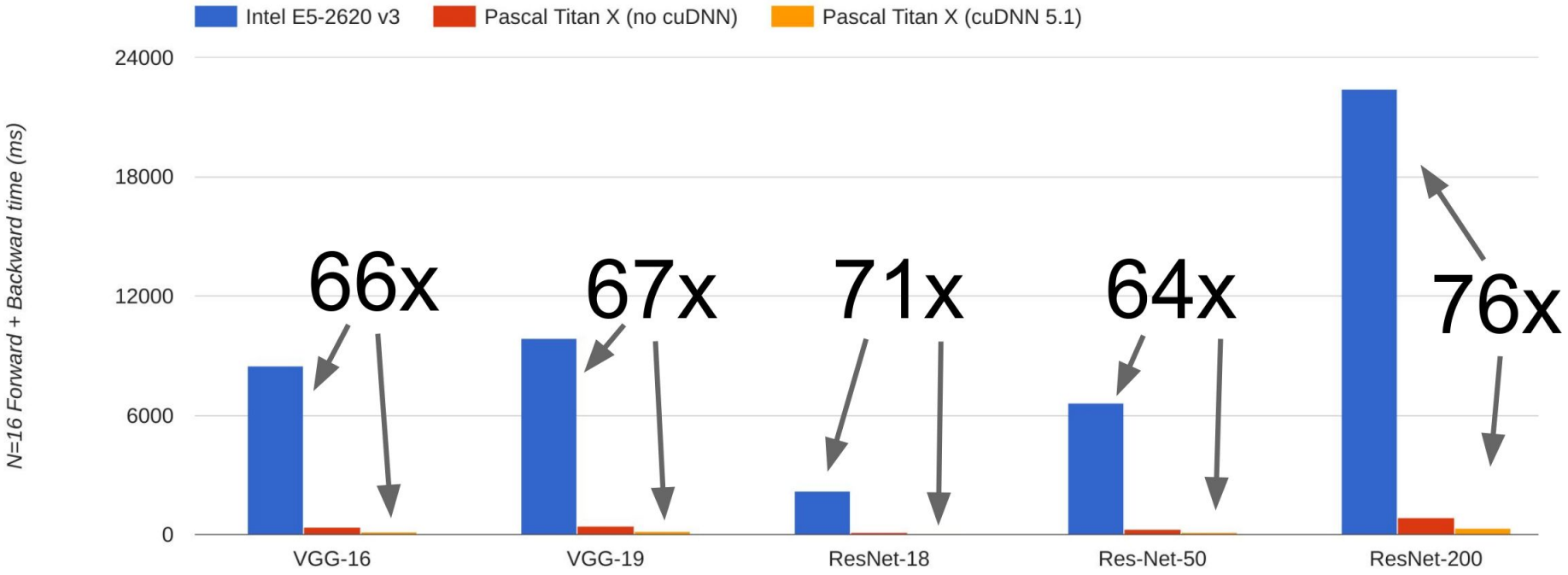
GPU and Deep Learning

Programming GPUs

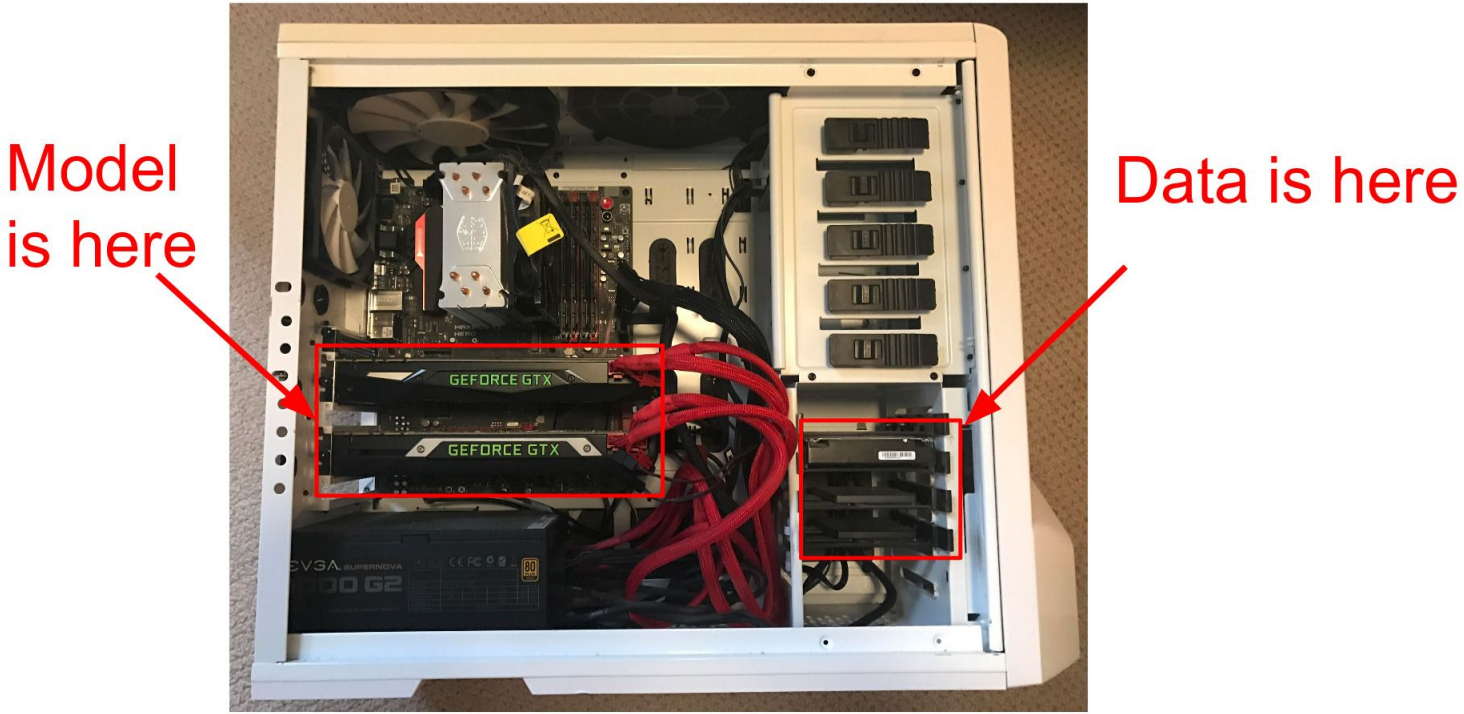
- CUDA (NVIDIA only)
  - Write C-like code that runs directly on the GPU
  - Higher-level APIs: cuBLAS, cuFFT, cuDNN, etc
- OpenCL
  - Similar to CUDA, but runs on anything
  - Usually slower :(

CPU vs GPU in practice

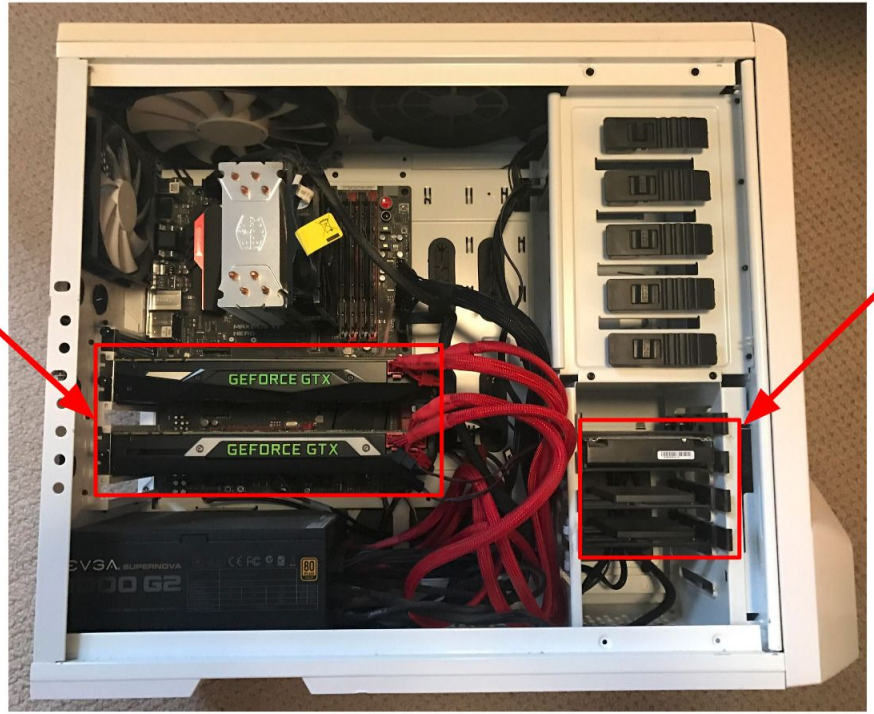
(CPU performance not well-optimized, a little unfair)



CPU / GPU Communication



GPU and Deep Learning
CPU / GPU Communication



If you aren't careful, training can bottleneck on reading data and transferring to GPU!

Solutions:

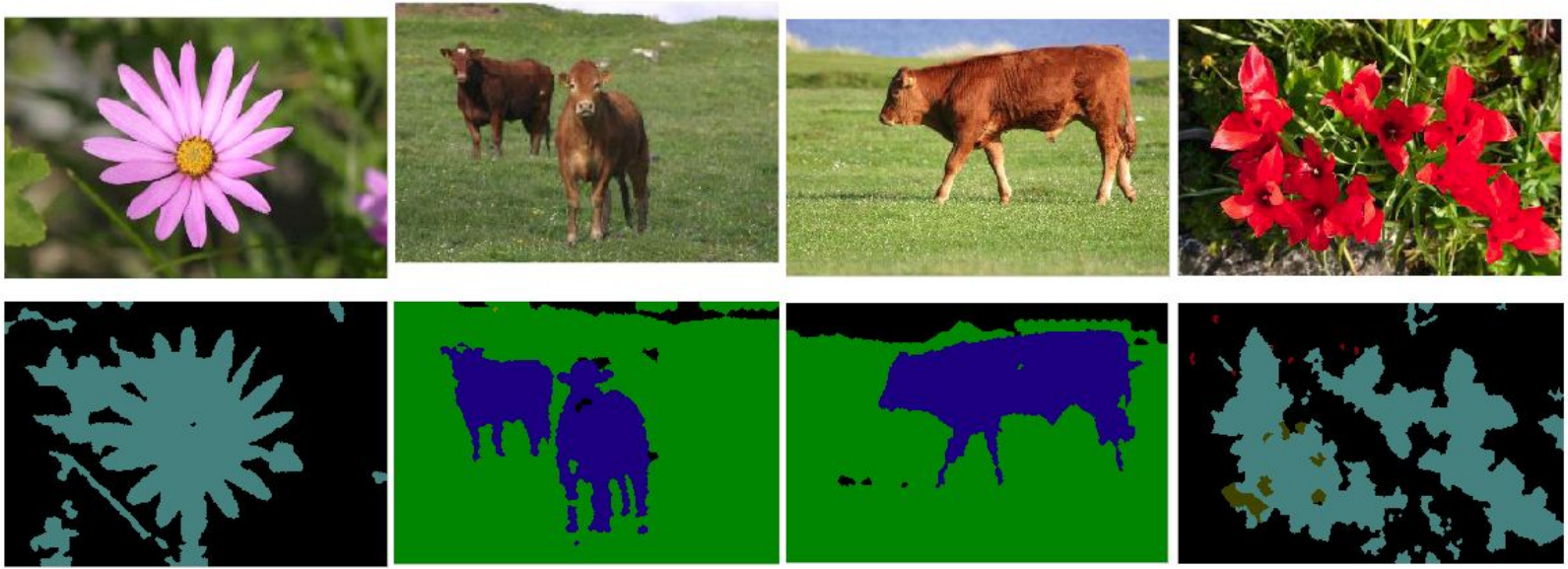
- Read all data into RAM
- Use SSD instead of HDD
- Use multiple CPU threads to prefetch data

Semantic Segmentation

DNNs, Datasets & Trends

Girish Varma
IIIT Hyderabad

Object Segmentation

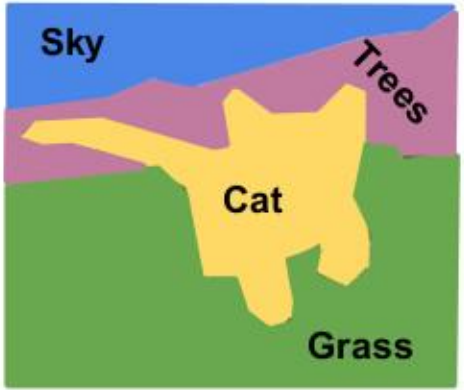


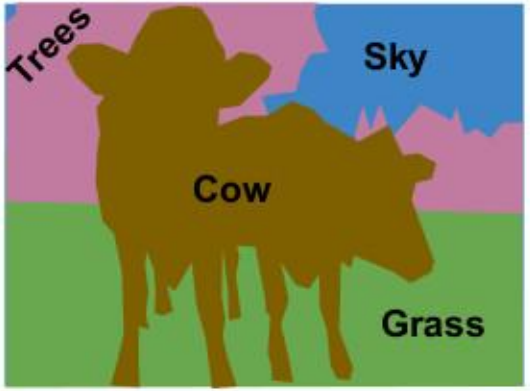
Binary classification for every pixel.

Semantic Segmentation

Label each pixel in the image with a category label

Don't differentiate instances, only care about pixels





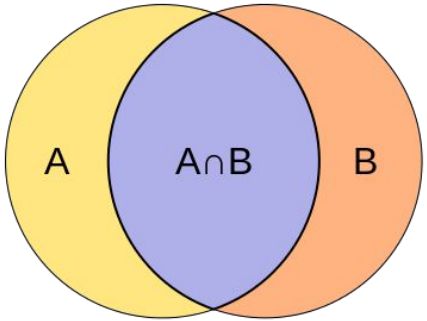
Metric

Pixel Accuracy

- fraction of pixels where prediction is correct.
- Problem: large background, small foreground. Simply predicting background everywhere gives high accuracy.

Jaccard Index of IoU (Intersection over Union)

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}.$$



Multiple classes: mIoU (mean IoU).

- Avg over classes c with equal weight, IoU(c)

Loss Function

Model output shape : NxHxWxC

For every n,h,w,

O[n,h,w] is a probability distribution over C classes.

Compute softmax cross entropy for each n,h,w

Take mean over N, H, W dimensions.

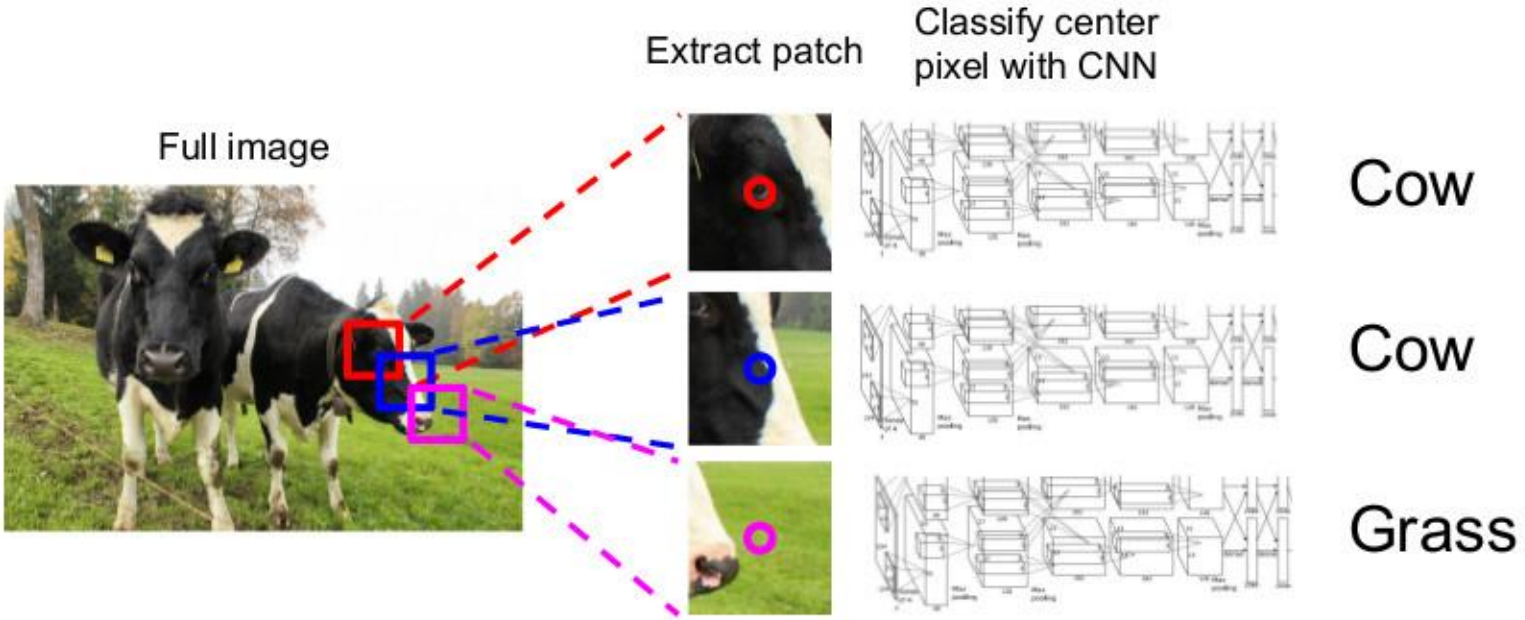
Problem: Above optimizes pixel accuracy

New loss functions:

The Lovász-Softmax loss: A tractable surrogate for the optimization of the intersection-over-union measure in neural networks. CVPR 2018.
Maxim Berman, Amal Rannen Triki, Matthew B. Blaschko. https://arxiv.org/abs/1705.08790v2

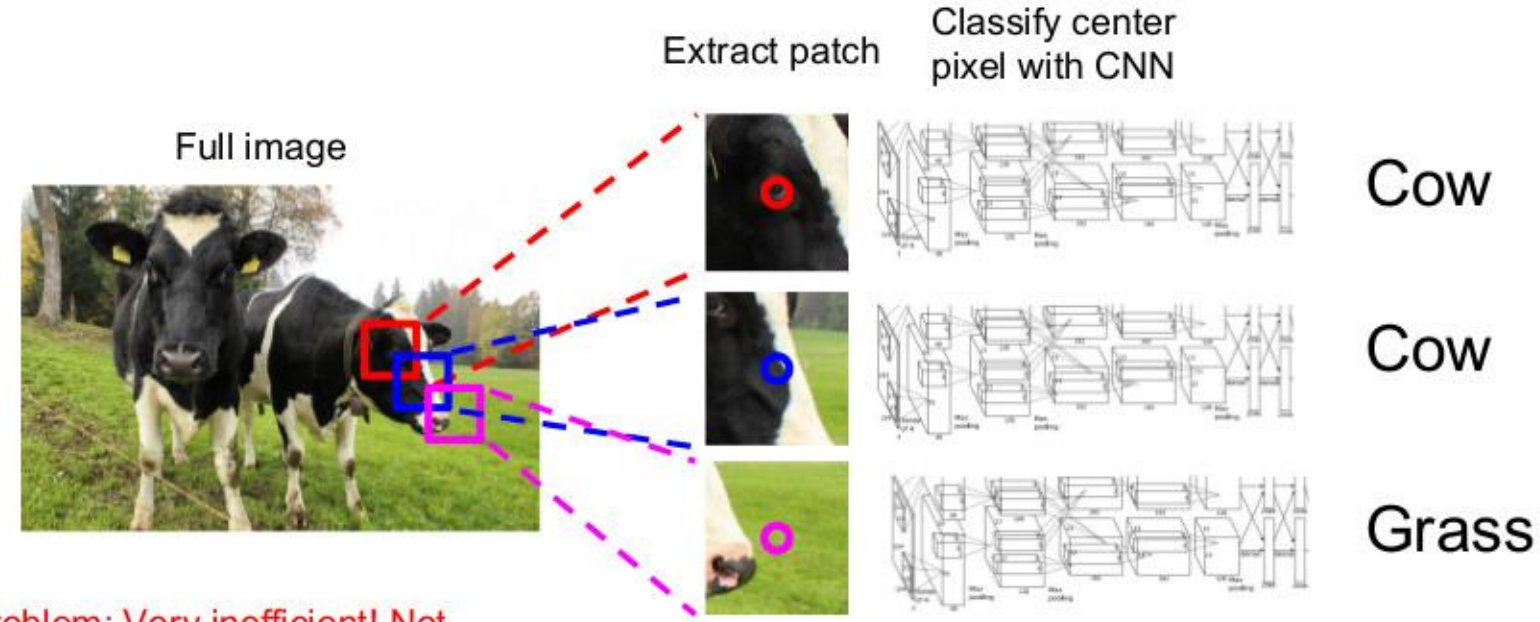
NeuroIoU: Learning a Surrogate Loss for Semantic Segmentation. BMVC 2018
G Nagendar, Digvijay Singh and C. V. Jawahar

Semantic Segmentation Idea: Sliding Window



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 Idea: Sliding Window

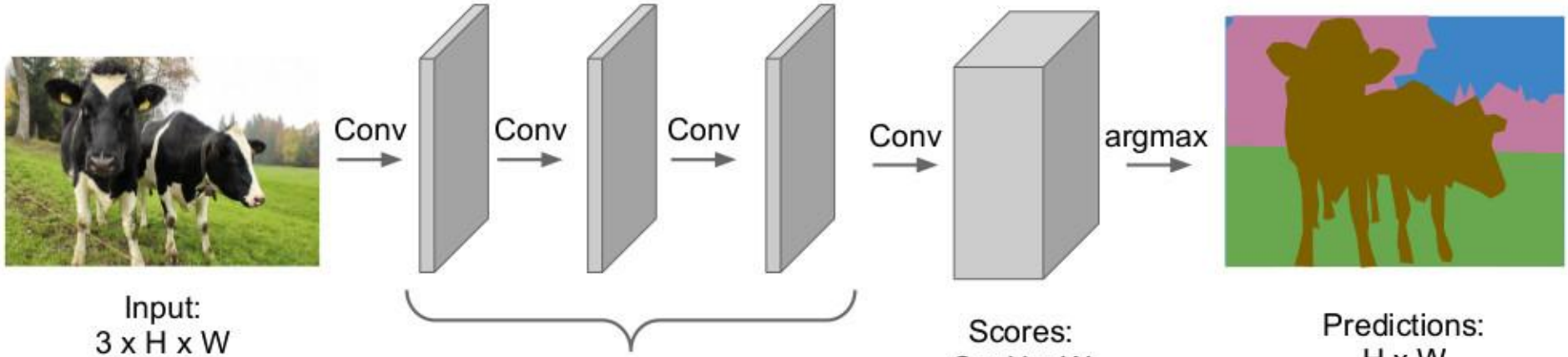


Problem: Very inefficient! Not reusing shared features between overlapping patches

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

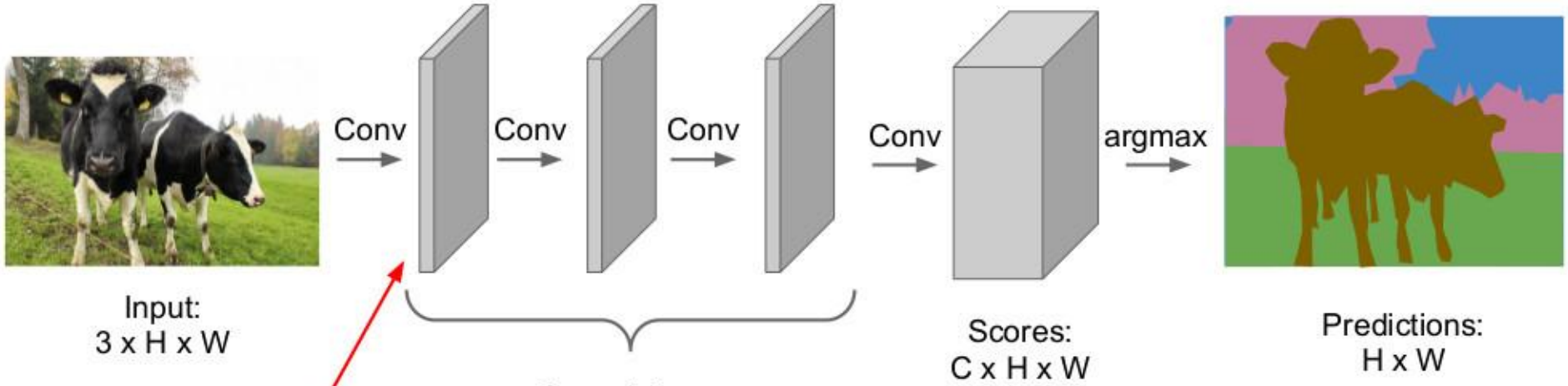
Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



Semantic Segmentation Idea: Fully Convolutional

Design a network as a bunch of convolutional layers to make predictions for pixels all at once!



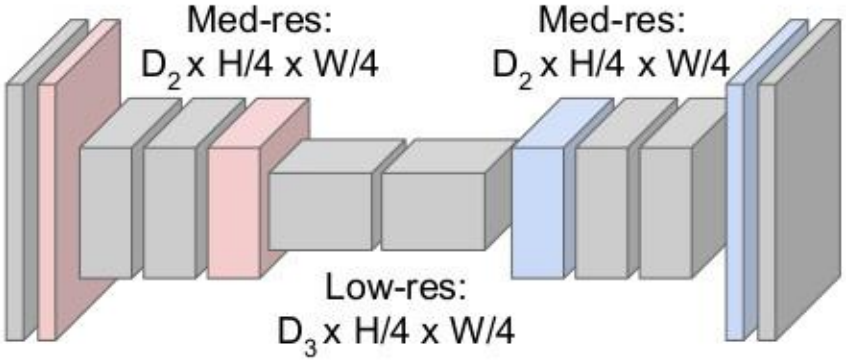
Problem: convolutions at original image resolution will be very expensive ...

Problem 2: Low mixing of information.

Semantic Segmentation Idea: Fully Convolutional

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Input:
3 x H x W



High-res:
D1 x H/2 x W/2

High-res:
D1 x H/2 x W/2

Predictions:
H x W

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

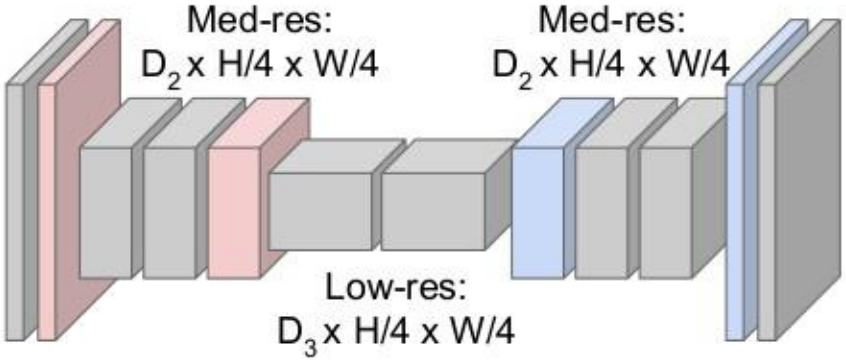
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
???

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

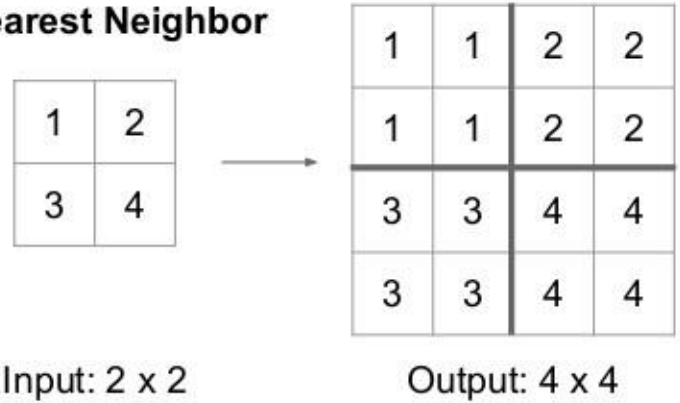
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

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

In-Network upsampling: “Unpooling”

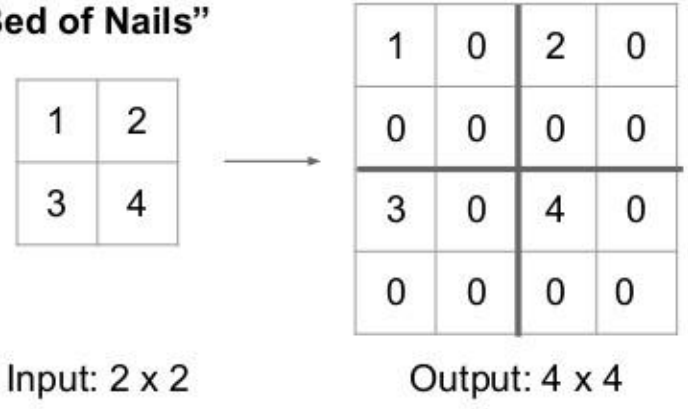
Nearest Neighbor



Input: 2 x 2

Output: 4 x 4

“Bed of Nails”

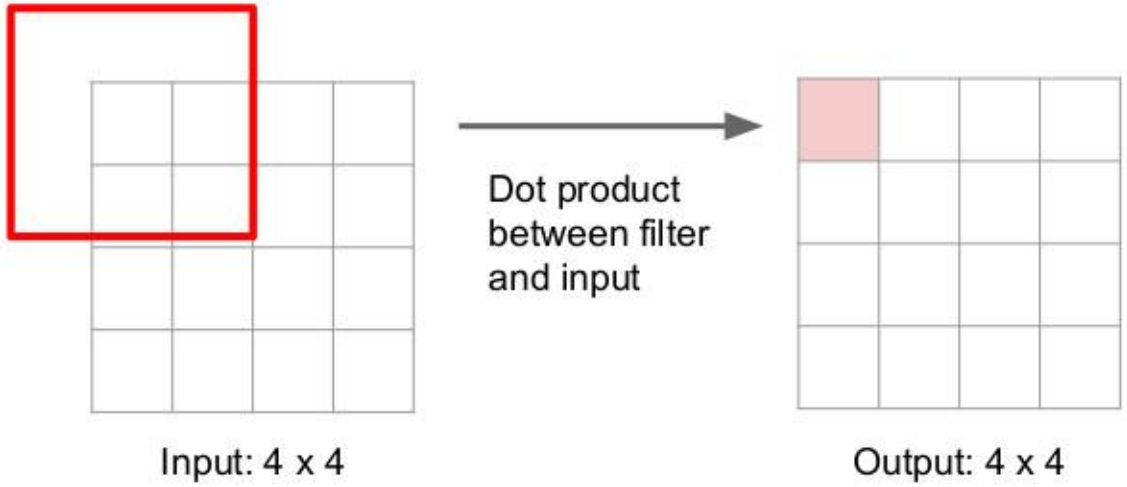


Input: 2 x 2

Output: 4 x 4

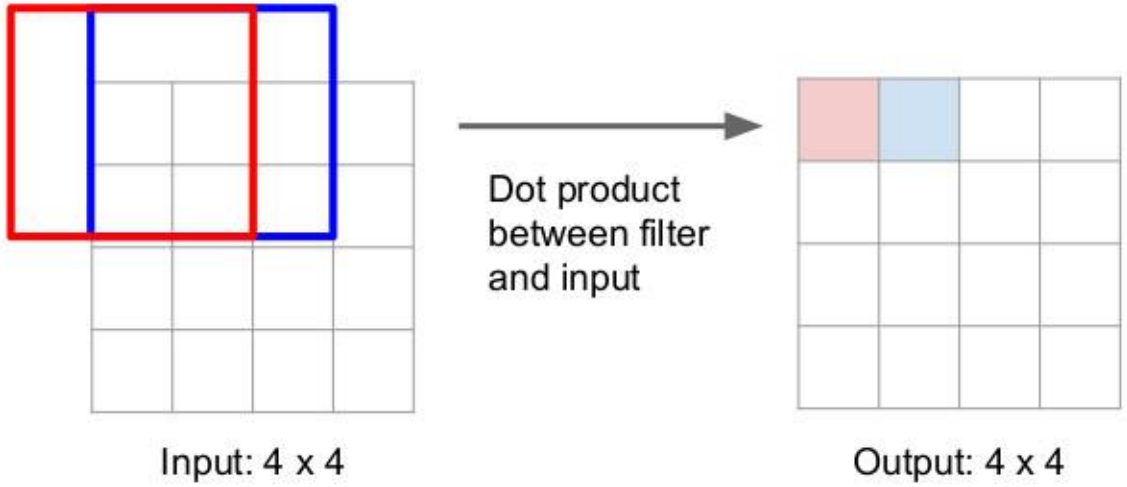
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1



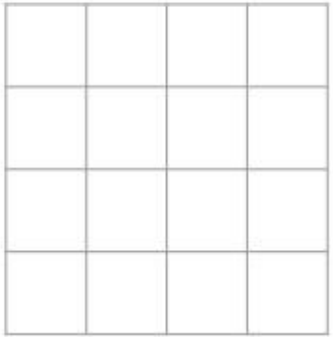
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 1 pad 1

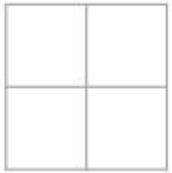


Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



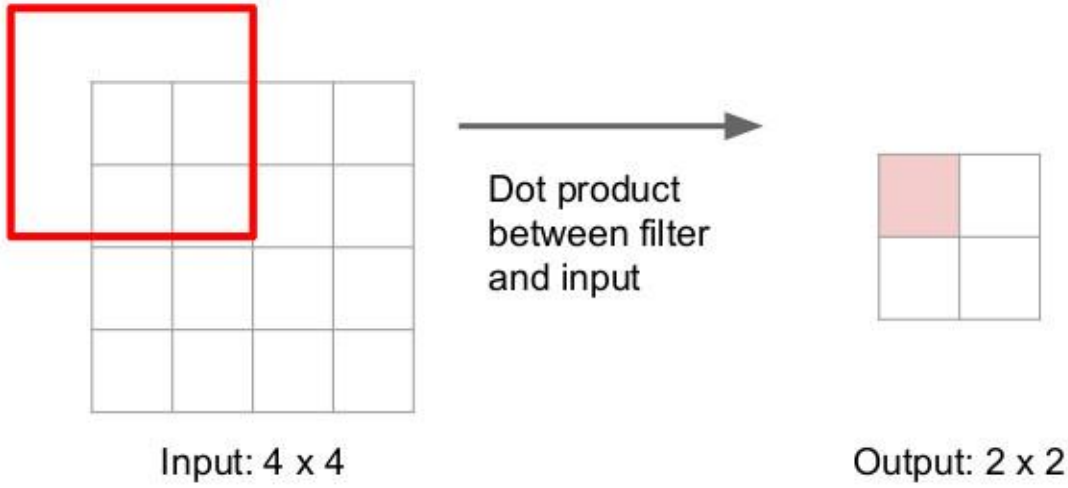
Input: 4 x 4



Output: 2 x 2

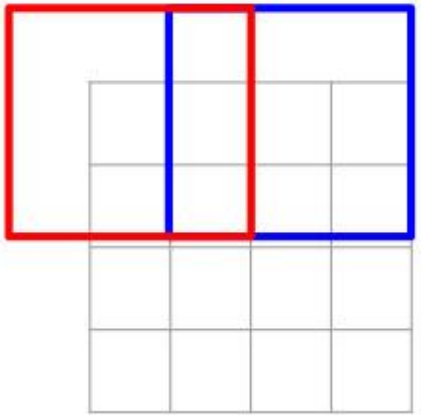
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



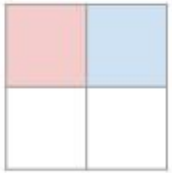
Learnable Upsampling: Transpose Convolution

Recall: Normal 3 x 3 convolution, stride 2 pad 1



Input: 4 x 4

Dot product between filter and input



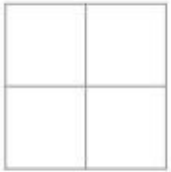
Output: 2 x 2

Filter moves 2 pixels in the input for every one pixel in the output

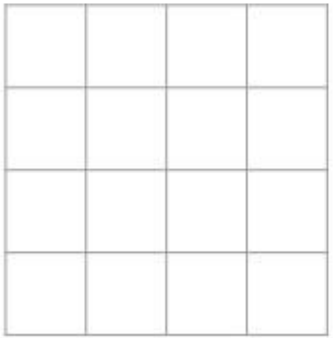
Stride gives ratio between movement in input and output

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



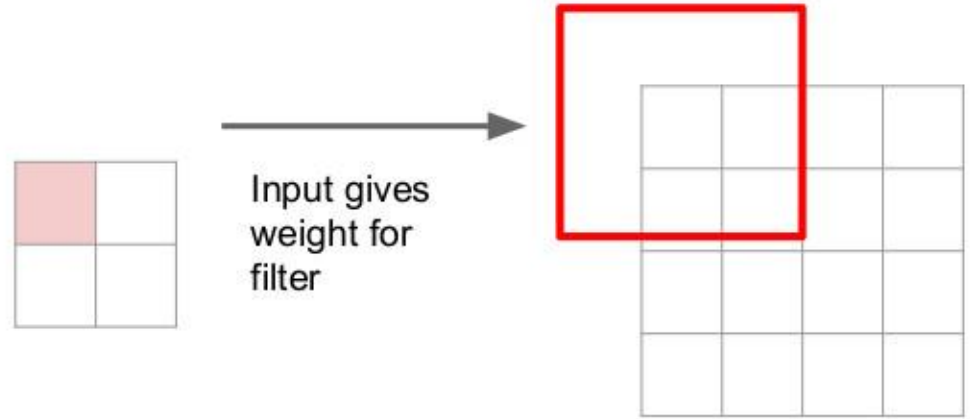
Input: 2 x 2



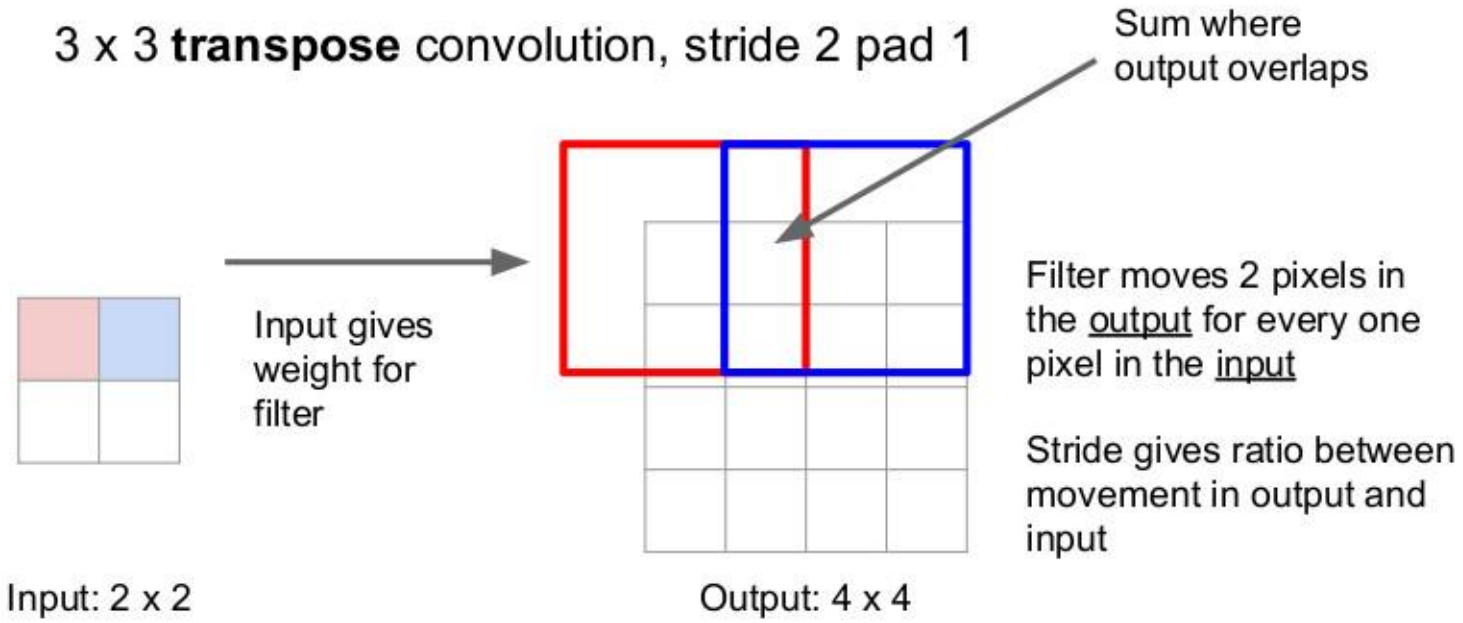
Output: 4 x 4

Learnable Upsampling: Transpose Convolution

3 x 3 **transpose** convolution, stride 2 pad 1



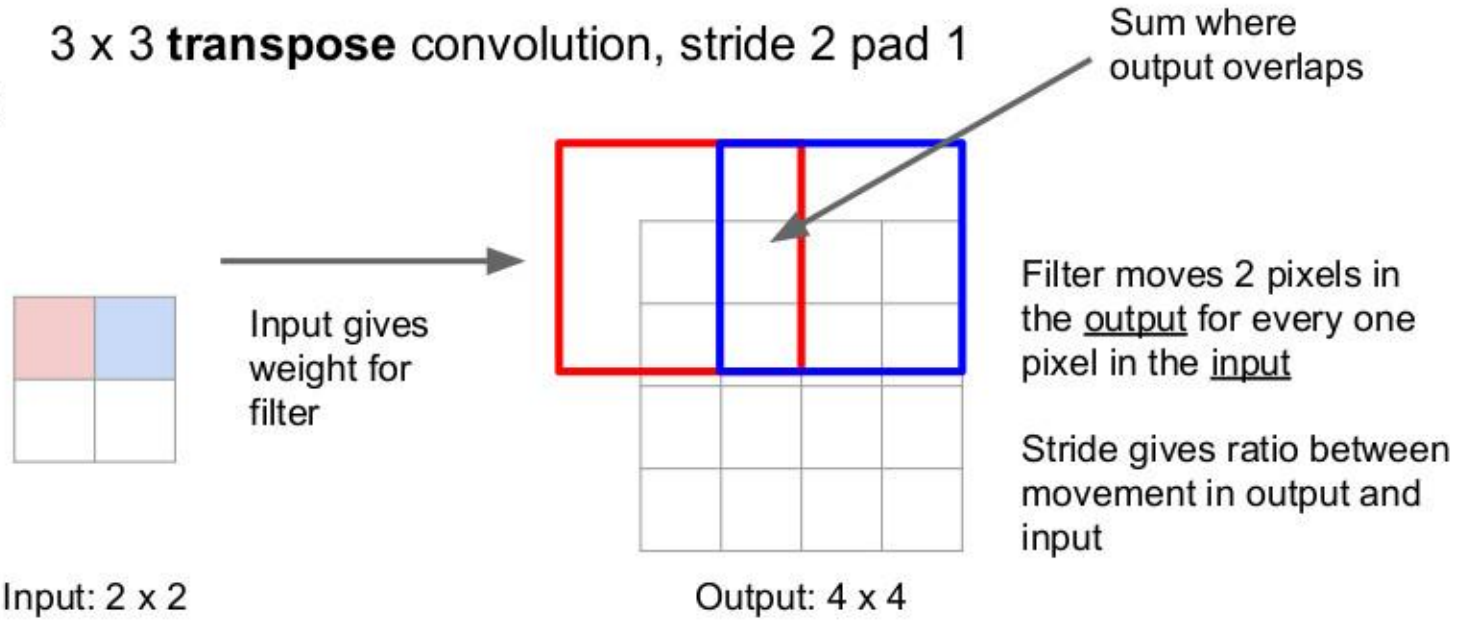
Learnable Upsampling: Transpose Convolution



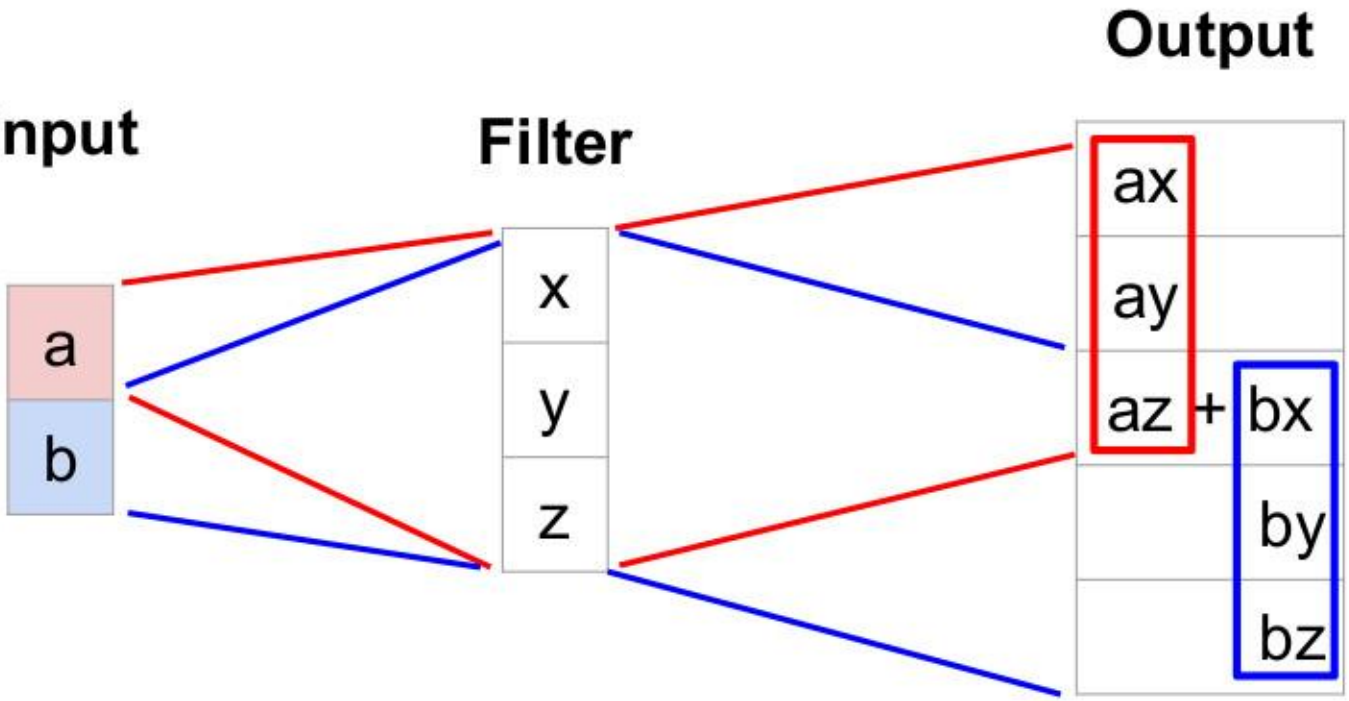
Learnable Upsampling: Transpose Convolution

Other names:
-Deconvolution (bad)
-Upconvolution
-Fractionally strided convolution
-Backward strided convolution

3 x 3 **transpose** convolution, stride 2 pad 1



Learnable Upsampling: 1D Example



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

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

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution as Matrix Multiplication (1D Example)

We can express convolution in terms of a matrix multiplication

$$\vec{x} * \vec{a} = X\vec{a}$$

$$\begin{bmatrix} x & y & x & 0 & 0 & 0 \\ 0 & x & y & x & 0 & 0 \\ 0 & 0 & x & y & x & 0 \\ 0 & 0 & 0 & x & y & x \end{bmatrix} \begin{bmatrix} 0 \\ a \\ b \\ c \\ d \\ 0 \end{bmatrix} = \begin{bmatrix} ay + bz \\ ax + by + cz \\ bx + cy + dz \\ cx + dy \end{bmatrix}$$

Example: 1D conv, kernel size=3, stride=1, padding=1

Convolution transpose multiplies by the transpose of the same matrix:

$$\vec{x} *^T \vec{a} = X^T \vec{a}$$

$$\begin{bmatrix} x & 0 & 0 & 0 \\ y & x & 0 & 0 \\ z & y & x & 0 \\ 0 & z & y & x \\ 0 & 0 & z & y \\ 0 & 0 & 0 & z \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} ax \\ ay + bx \\ az + by + cx \\ bz + cy + dx \\ cz + dy \\ dz \end{bmatrix}$$

When stride=1, convolution transpose is just a regular convolution (with different padding rules)

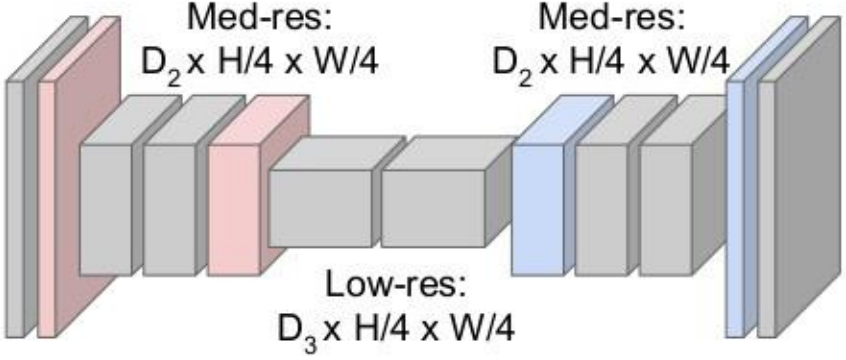
Semantic Segmentation Idea: Fully Convolutional

Downsampling:
Pooling, strided convolution

Design network as a bunch of convolutional layers, with **downsampling** and **upsampling** inside the network!

Upsampling:
Unpooling or strided transpose convolution

Input:
3 x H x W



High-res:
D_1 x H/2 x W/2

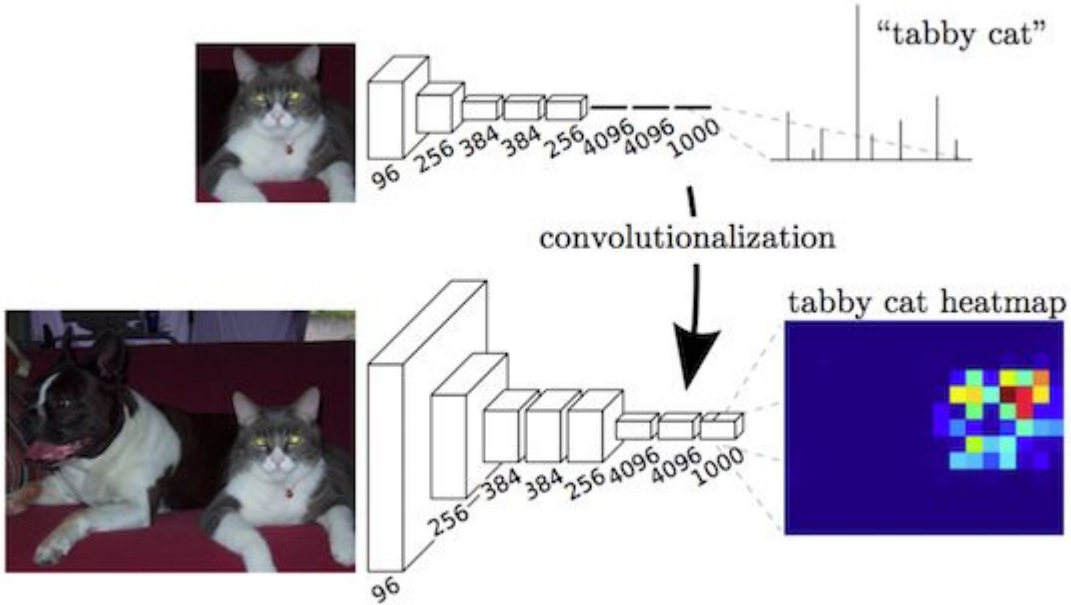
High-res:
D_1 x H/2 x W/2

Predictions:
H x W

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

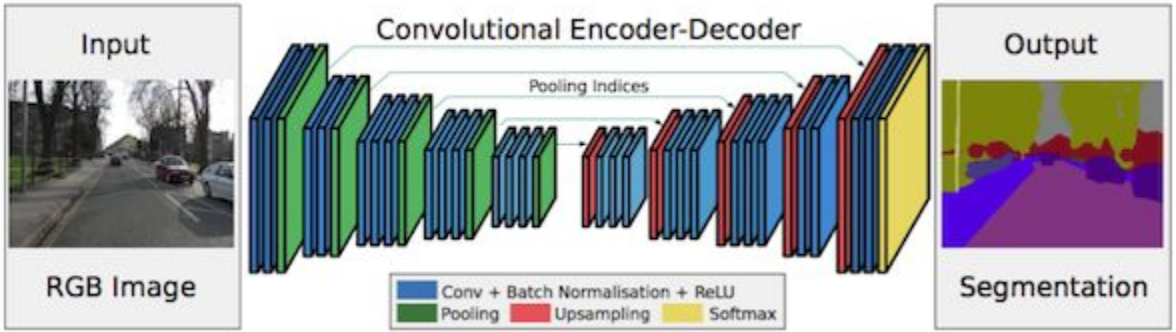
Models

Basic FCNs: Fully Convolutional Neural Networks



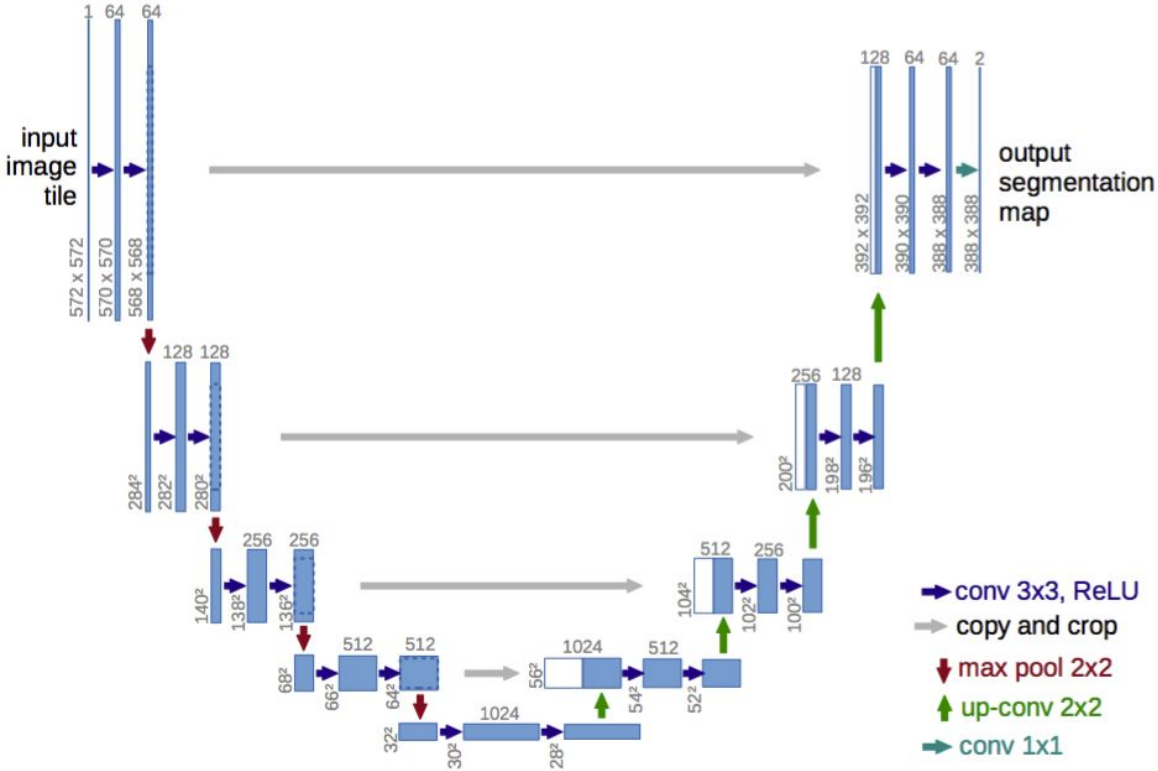
Pros: Can reuse ImageNet trained weights for segmentation networks.

Encoder-Decoder: Segnet

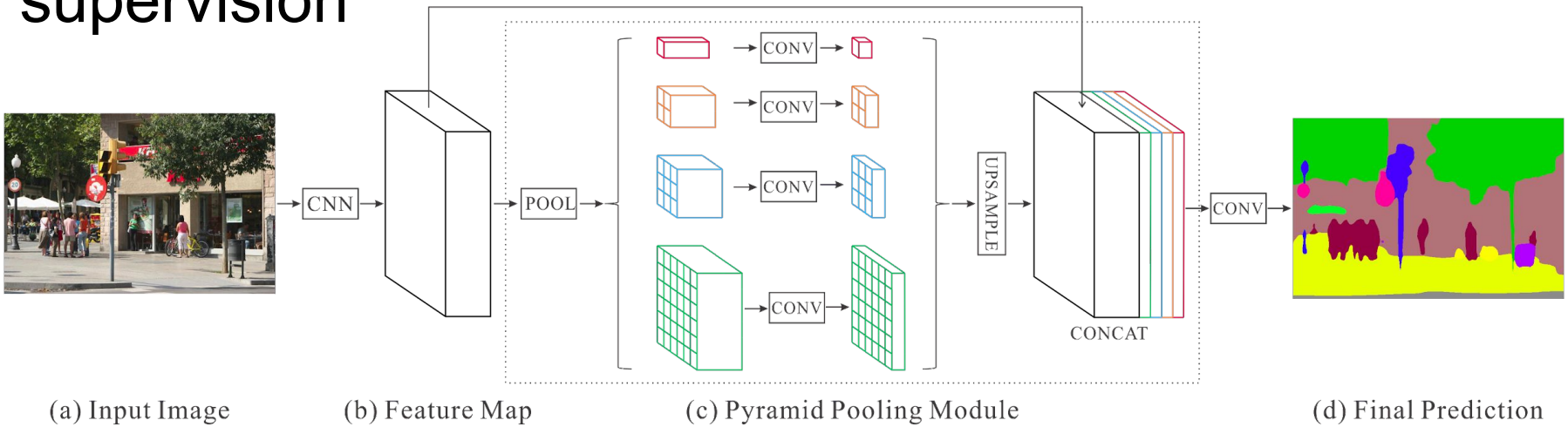


Introduced Decoder

Encoder Decoder: UNet



PSPNet: Multiscale Features and Intermediate supervision



(a) Input Image (b) Feature Map (c) Pyramid Pooling Module (d) Final Prediction

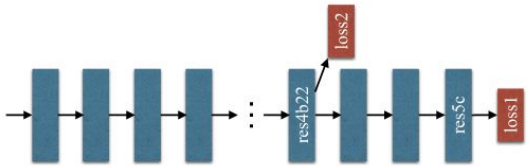
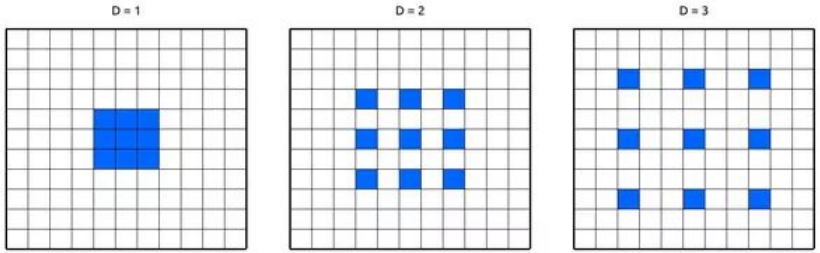
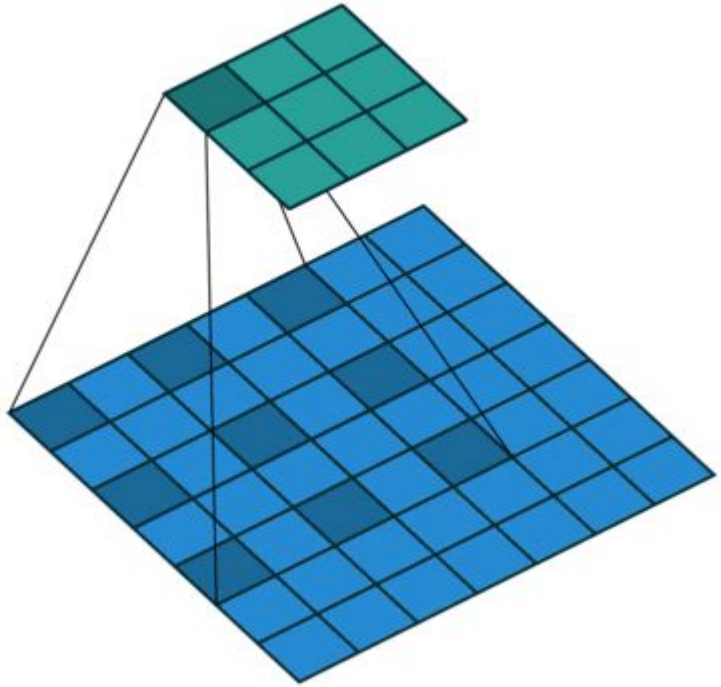


Figure 4. Illustration of auxiliary loss in ResNet101. Each blue box denotes a residue block. The auxiliary loss is added after the res4b22 residue block.

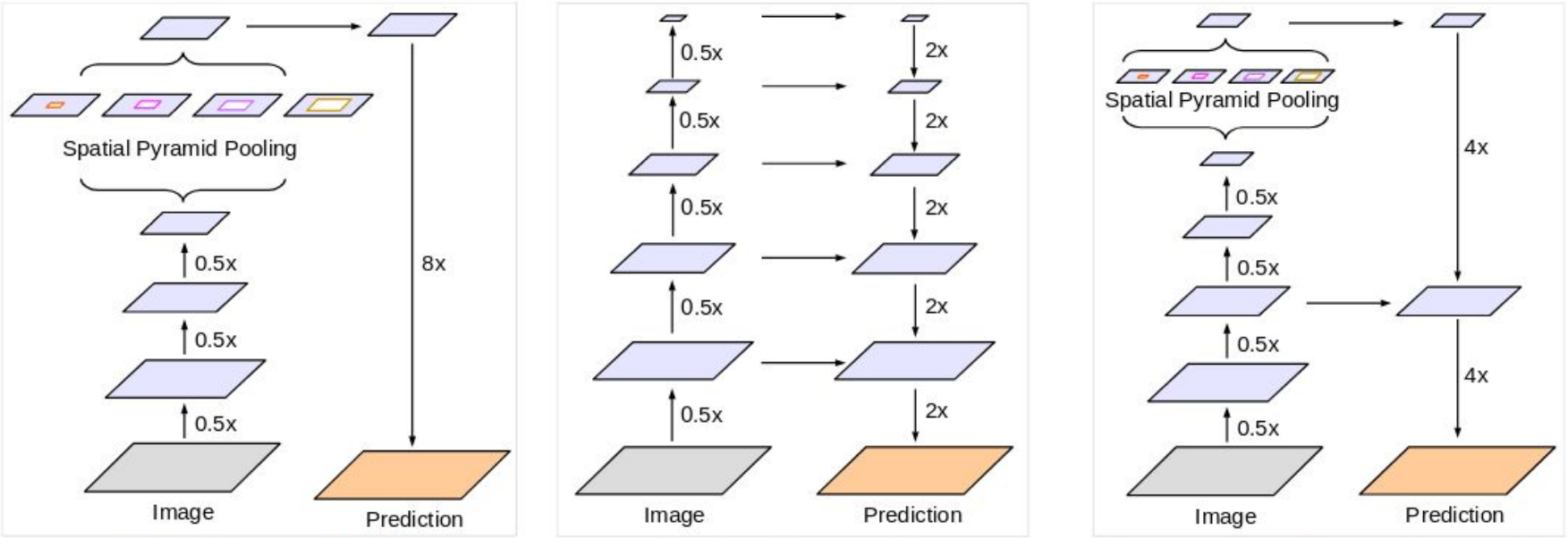
Atrous/Dilated Convolution

For mixing information among pixels, large window size required. Blows up running time!
How to mix pixels without incurring runtime cost?





DeepLab V3+



(a) Spatial Pyramid Pooling (b) Encoder-Decoder (c) Encoder-Decoder with Atrous Conv.

Figure 1. We propose to improve DeepLabv3, which employs the spatial pyramid pooling module (a), with the encoder-decoder structure (b). The proposed model, DeepLabv3+, contains rich semantic information from the encoder module, while the detailed object boundaries are recovered by the simple yet effective decoder module. The encoder module allows us to extract features at an arbitrary resolution by applying atrous convolution.

DeepLab V3+

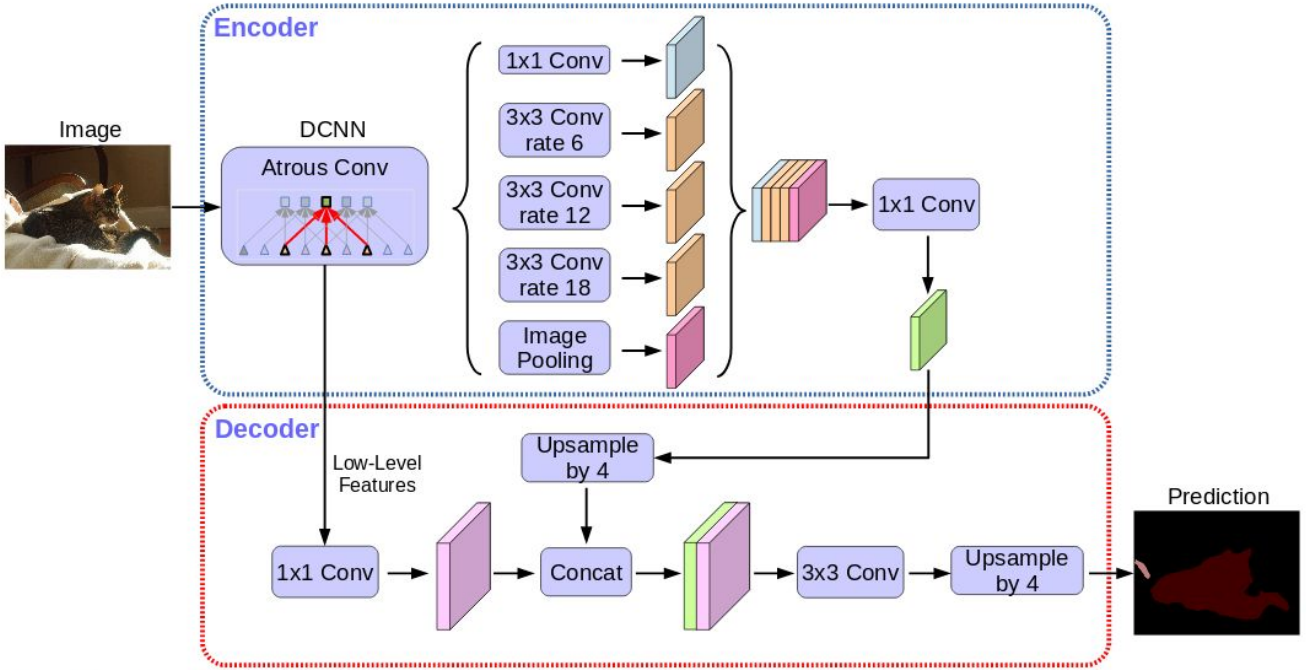
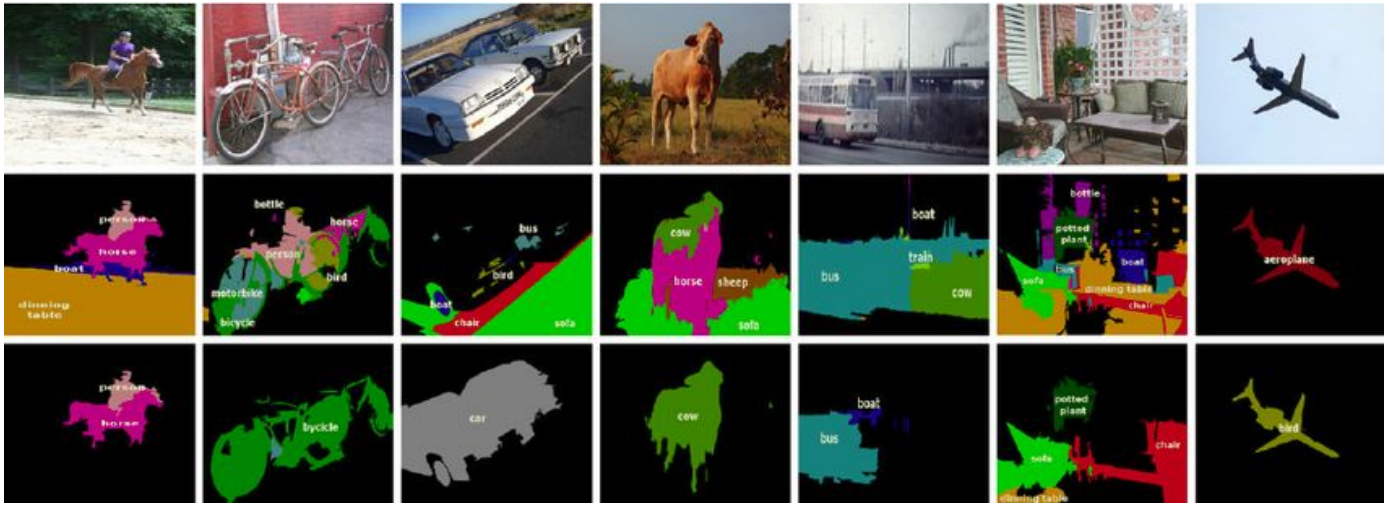


Figure 2. Our proposed DeepLabv3+ extends DeepLabv3 by employing a encoder-decoder structure. The encoder module encodes multi-scale contextual information by applying atrous convolution at multiple scales, while the simple yet effective decoder module refines the segmentation results along object boundaries.

Datasets

Pascal VOC http://host.robots.ox.ac.uk/pascal/VOC/



COCO



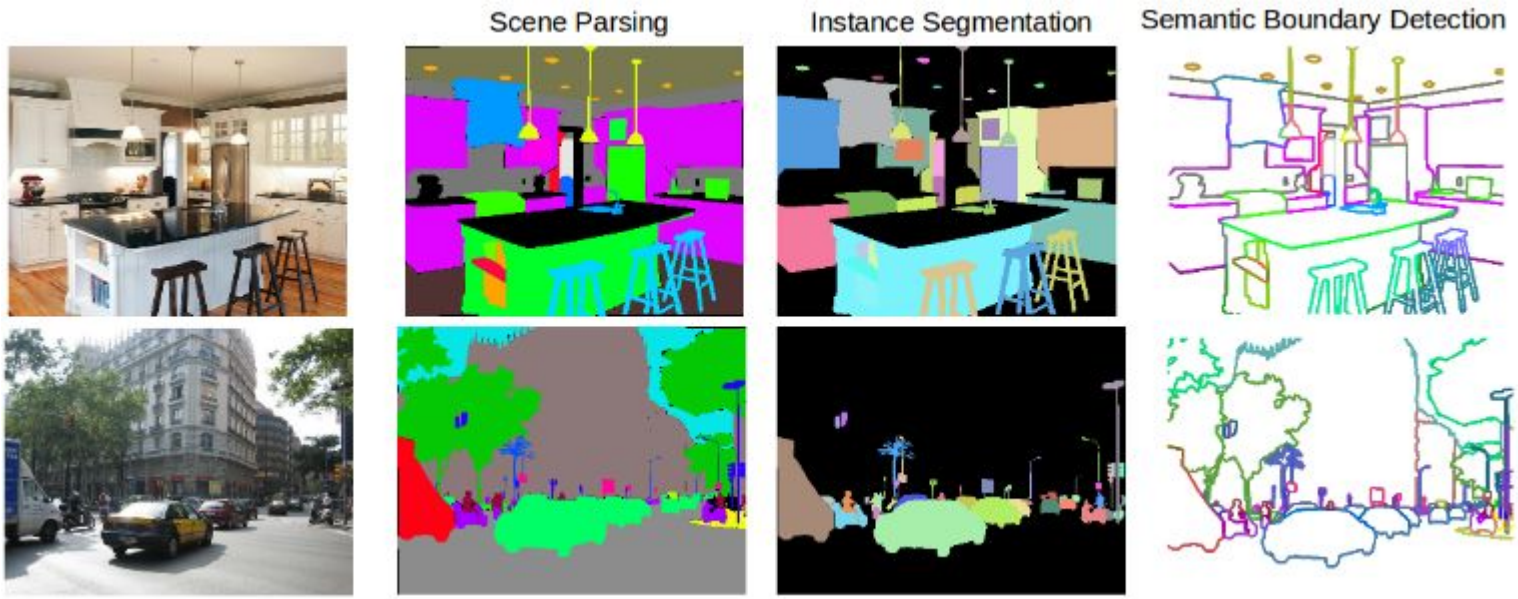
200,000 images and 80 object categories,

Mainly for Instance Segmentation

http://cocodataset.org/#home


ADE20K
Scene Parsing Challenge

ADE20K


Scene Parsing
Instance Segmentation
Semantic Boundary Detection

Cityscapes

19 classes, 3500 image/label pairs.

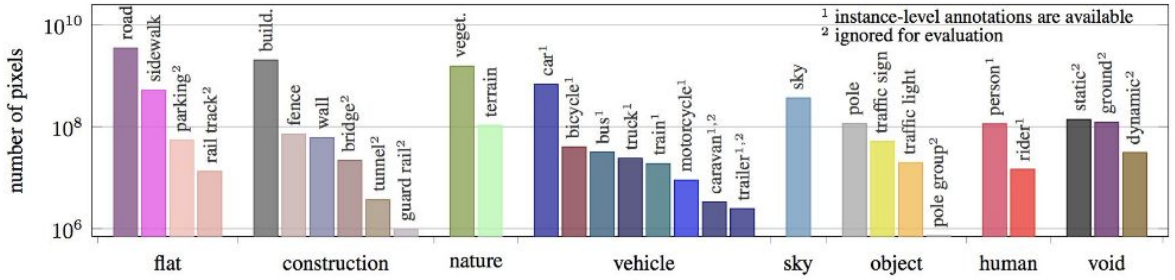


Figure 1. Number of finely annotated pixels (y-axis) per class and their associated categories (x-axis).

Synthia Dataset:

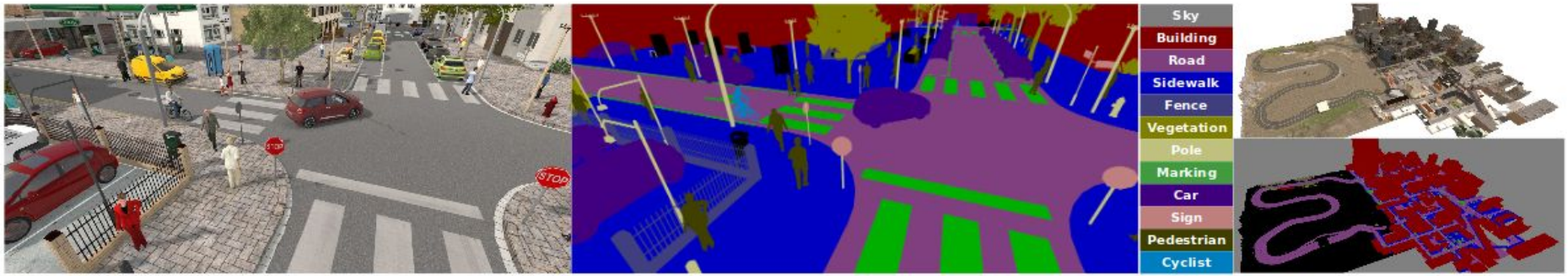
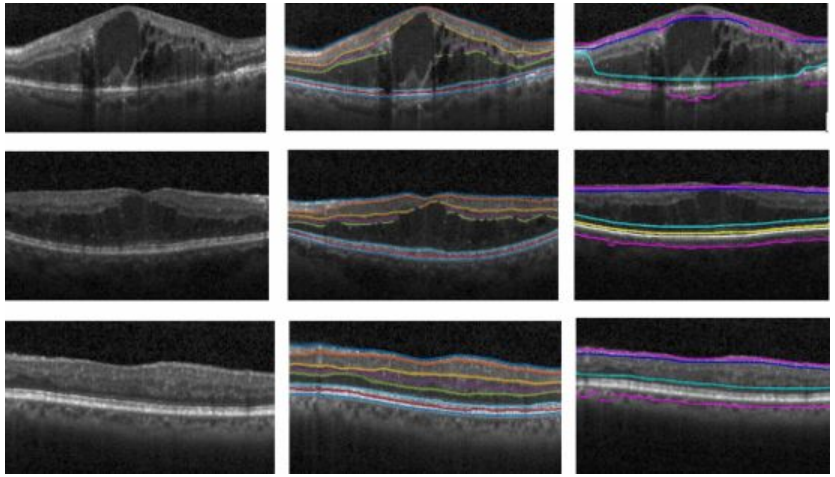
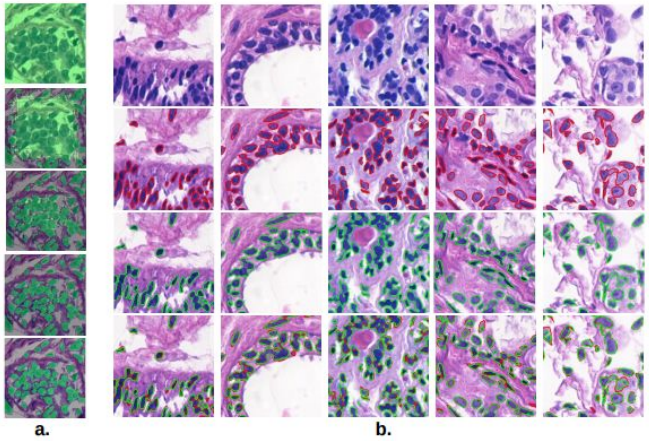


Figure. 1. The SYNTHIA Dataset. A sample frame (Left) with its semantic labels (center) and a general view of the city (right).

Applications: Medical Imaging

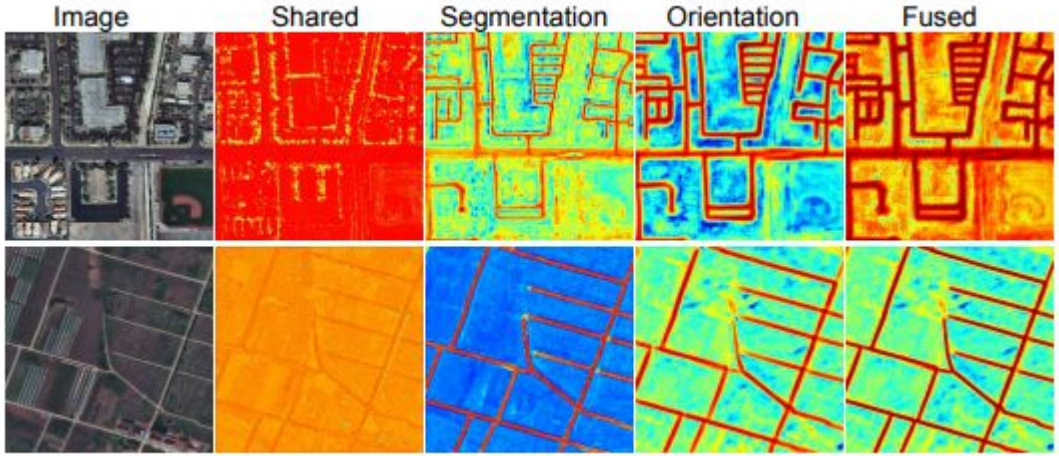
Retinal OCT scan





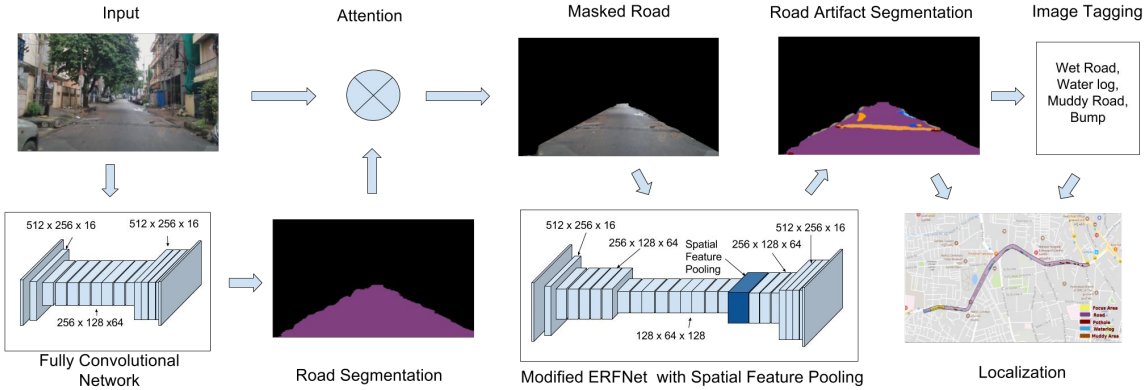
RACE-net: A Recurrent Neural Network for Biomedical Image Segmentation.
Chakravraty A and Jayanthi Sivaswamy
IEEE journal of biomedical and health informatics

Applications: Satellite Imagery



Improved Road Connectivity by Joint Learning of Orientation and Segmentation.
Anil Batra, Suriya Singh, Guan Pang, Saikat Basu, C.V. Jawahar, Manohar Paluri
CVPR 2019

Applications: Road Audit



City-Scale Road Audit System using Deep Learning. IROS 2018
Sudhir Yarram, Girish Varma, C.V. Jawahar
Dataset available: http://insaan.iiit.ac.in/

Instance Segmentation

Instance Segmentation

Detect instances, give category, label pixels

"simultaneous detection and segmentation" (SDS)

Lots of recent work (MS-COCO)

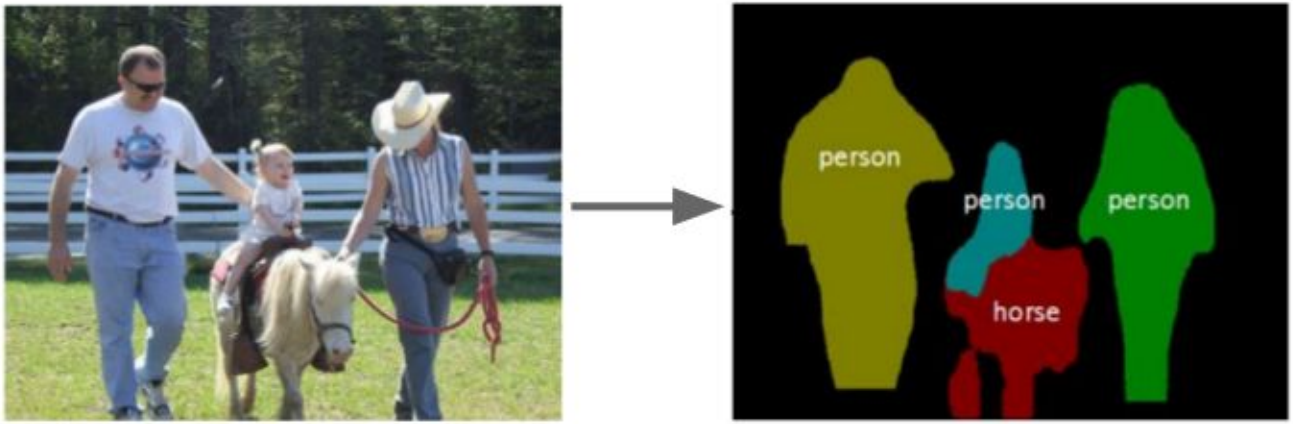
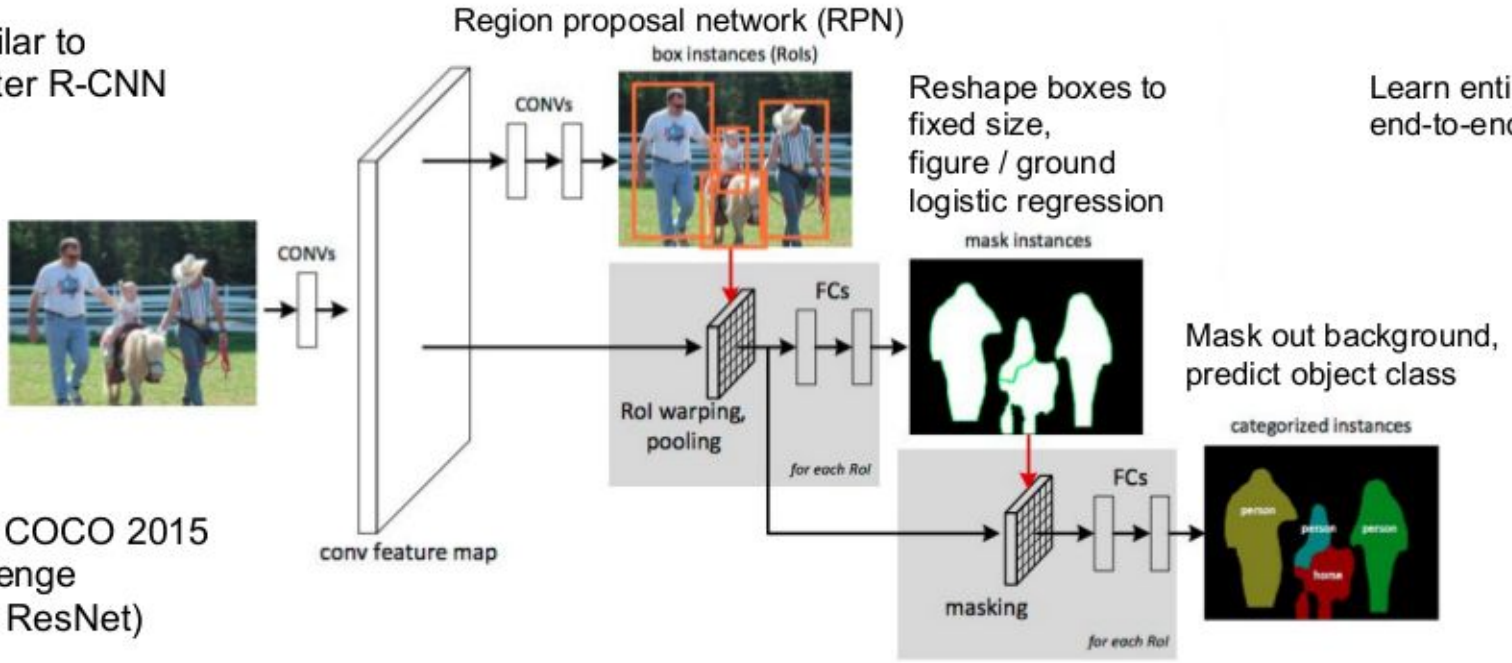


Figure credit: Dai et al, "Instance-aware Semantic Segmentation via Multi-task Network Cascades", arXiv 2015

Instance Segmentation: Cascades

Similar to
Faster R-CNN



Learn entire model
end-to-end!

Won COCO 2015
challenge
(with ResNet)

Dai et al, "Instance-aware Semantic Segmentation via Multi-task Network Cascades", arXiv 2015

Mask R-CNN

CNN

RoI Align

Classification Scores: C
Box coordinates (per class): 4 * C

256 x 14 x 14

Conv

256 x 14 x 14

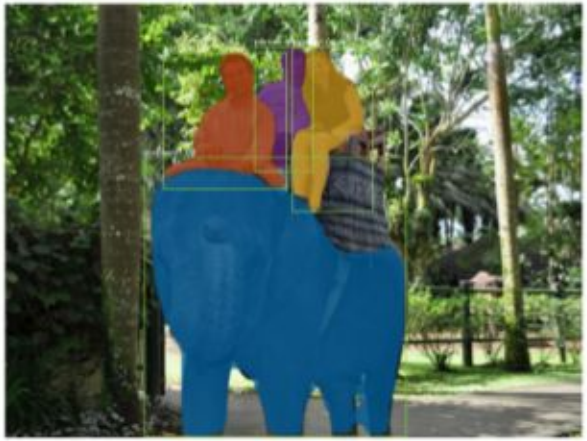
Conv

Predict a mask for each of C classes

C x 14 x 14

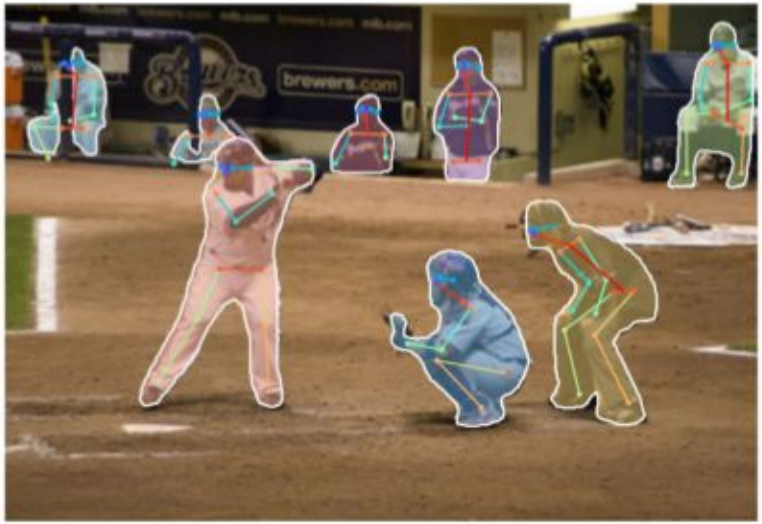
He et al, "Mask R-CNN", arXiv 2017

Mask R-CNN: Very Good Results!


He et al, "Mask R-CNN", arXiv 2017

Mask R-CNN
Also does pose




He et al, "Mask R-CNN", arXiv 2017
Figures copyright Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick, 2017.
Reproduced with permission.

PolygonRNN: Instance Segmentation as sequence prediction

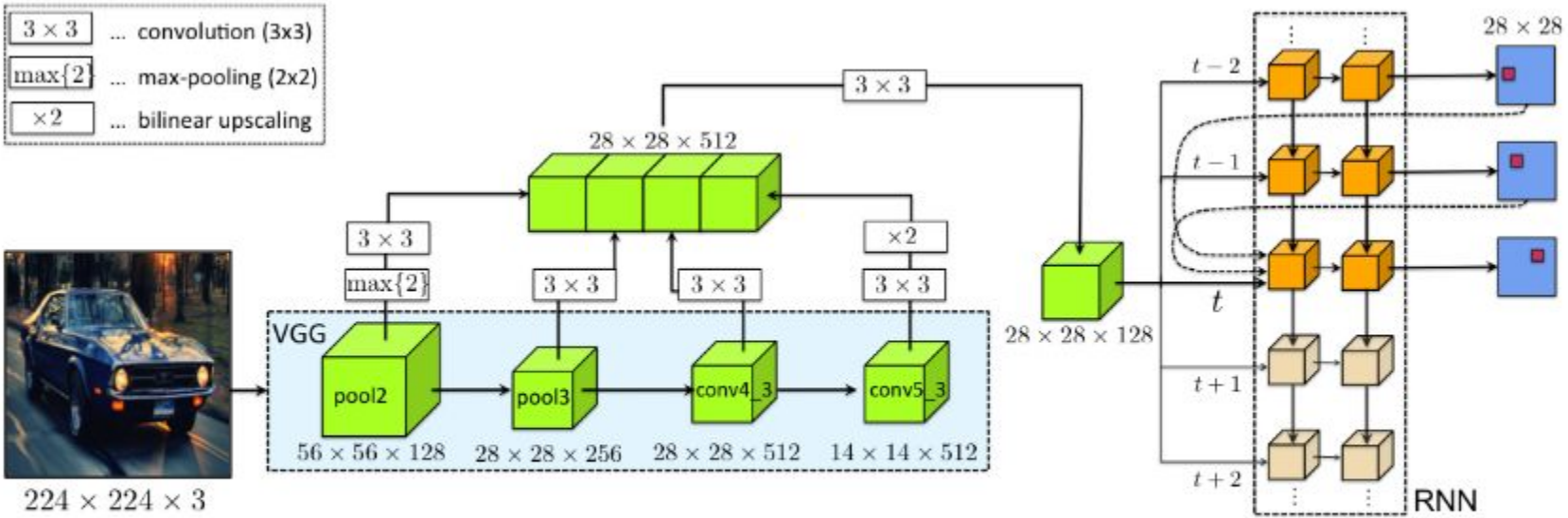
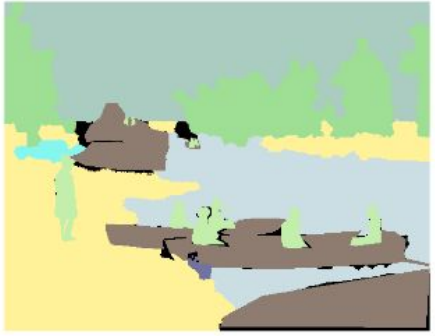


Figure 2. **Our Polygon-RNN model**. At each time step of the RNN-decoder (right), we feed in an image representation using a modified VGG architecture. Our RNN is a two-layer convolutional LSTM with skip-connection from one and two time steps ago. At the output at each time step, we predict the spatial location of the new vertex of the polygon.

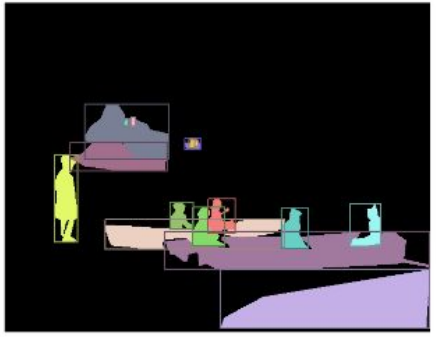
New Frontiers

Unifying Semantic and Instance Segmentation



Semantic Segmentation

- per-pixel annotation
- simple accuracy measure
- instances indistinguishable



Object Detection/Seg

- each object detected and segmented separately
- "stuff" is not segmented

Panoptic Segmentation

For each pixel i predict semantic label l **and instance id** z

➢ no overlaps between segments

- Popular datasets can be used
- We introduce simple, intuitive metric
- Drive novel algorithmic ideas

Joint Semantic + Instance Networks for Panoptic

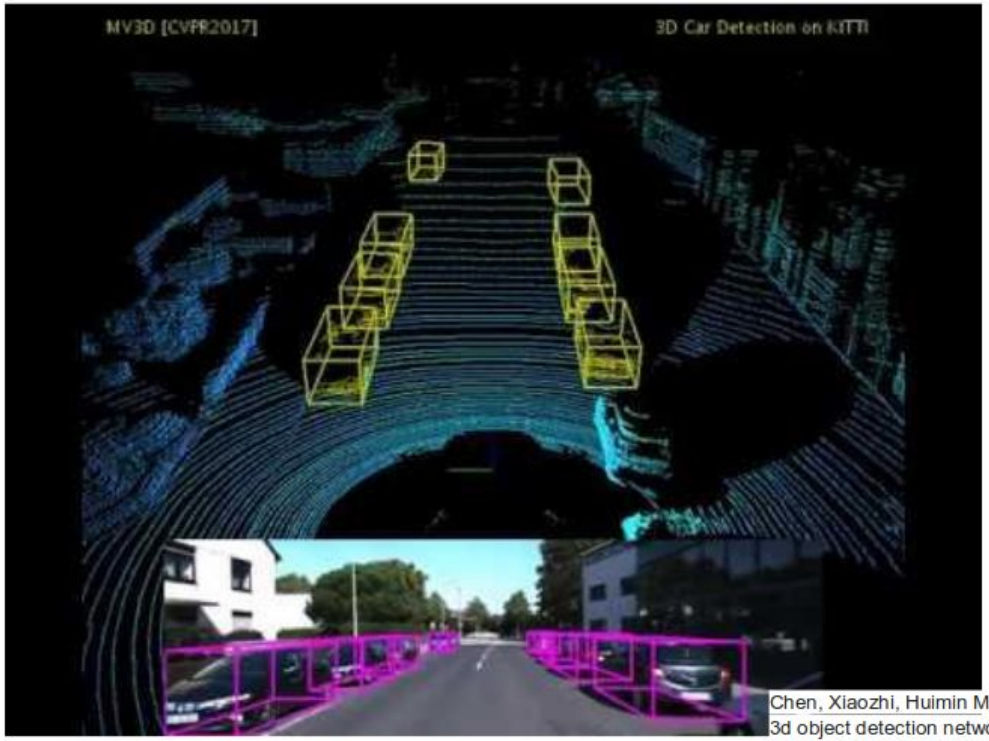
DeeperLab: Single-Shot Image Parser

http://deeperlab.mit.edu/

UPSNet: A Unified Panoptic Segmentation Network

https://arxiv.org/pdf/1901.03784.pdf

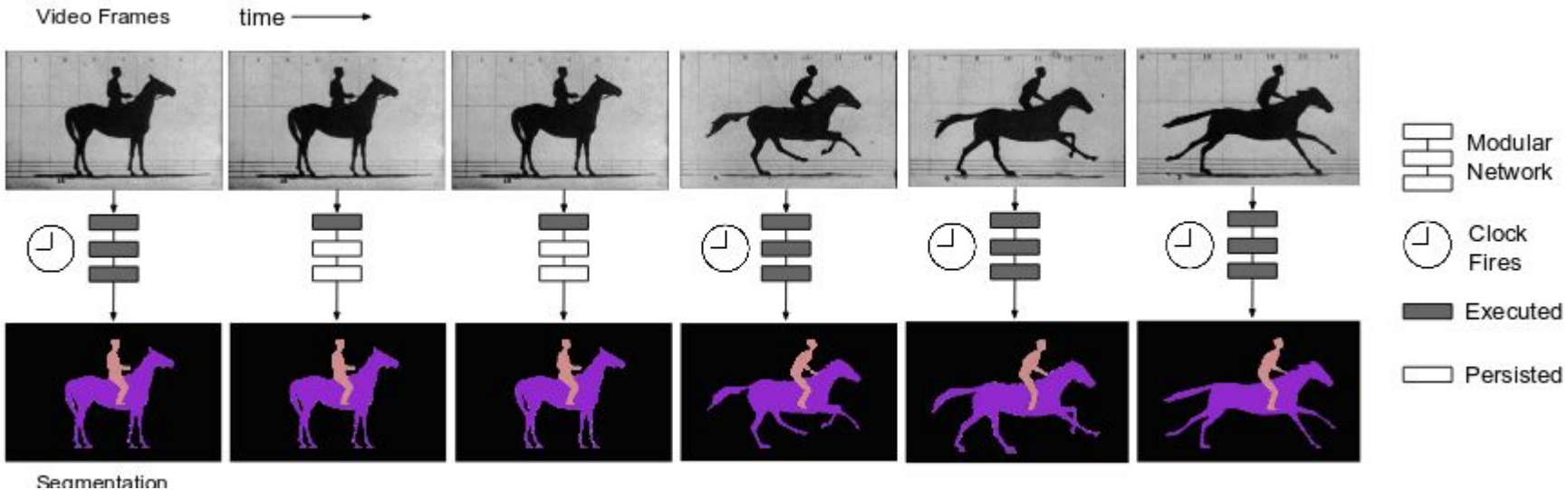
3D Object Detection: Camera + LiDAR



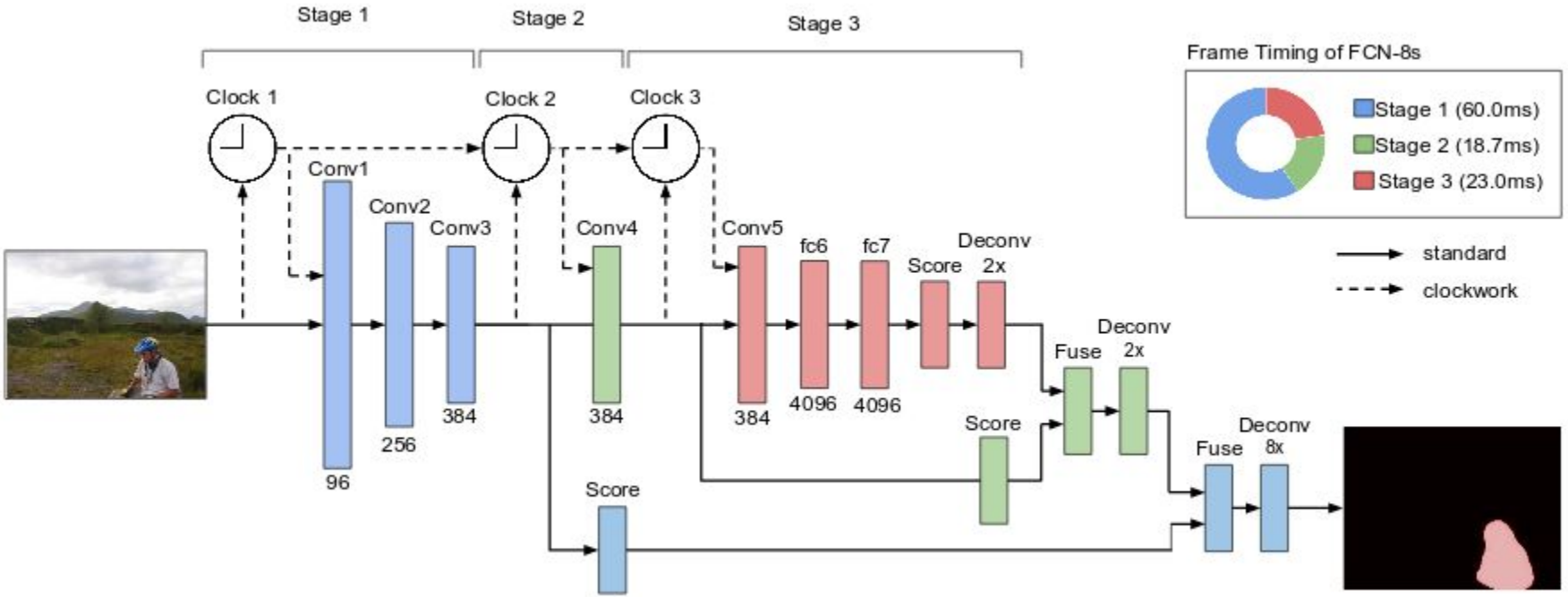
Chen, Xiaozhi, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. "Multi-view 3d object detection network for autonomous driving." *CVPR* 2017

Realtime Models

Clockwork FCN

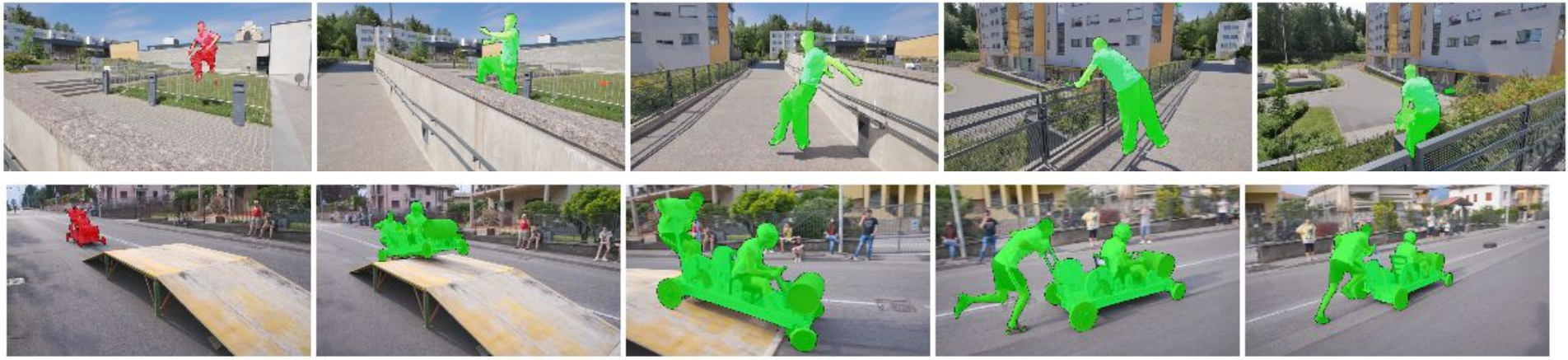


Clockwork FCN



Video One Shot Segmentation

OSVOS: One-Shot Video Object Segmentation



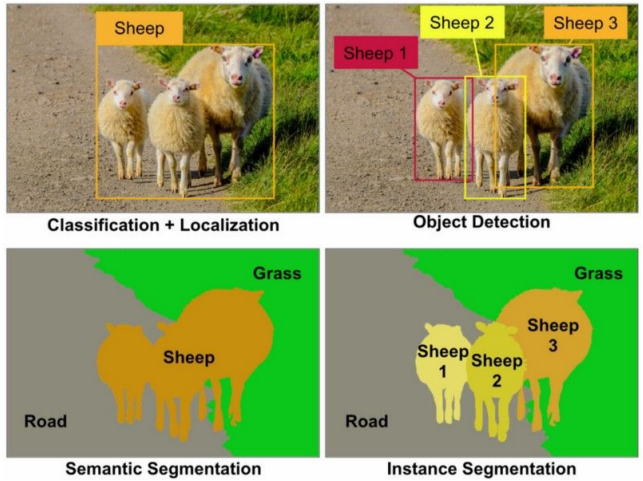
One Shot Segmentation

Figure 2. **Overview of OSVOS**: (1) We start with a pre-trained base CNN for image labeling on ImageNet; its results in terms of segmentation, although conform with some image features, are not useful. (2) We then train a *parent network* on the training set of DAVIS; the segmentation results improve but are not focused on an specific object yet. (3) By fine-tuning on a segmentation example for the specific target object in a single frame, the network rapidly focuses on that target.


IDD
A Dataset for Exploring Problems of
Autonomous Navigation in Unstructured Environments
http://idd.insaan.iiit.ac.in/

Auto. Nav. Dataset

- Images from road scenes
- Pixel level/Bounding box Annotations
- Semantic/instance segmentation, Detection
- A basic primitive for Auto. Nav.



Existing Datasets

- Camvid (2008)
  - ETHZ, Cambridge
- KITTI (2012-14)
  - Comprehensive set of data
  - MPI, TTI, KIT
- Cityscapes (2015-16)
  - Daimler, MPI, TU Darmstad
- Mapillary (2017-18)
  - User uploaded image from around the world
- BDD100K (2018)
  - Berkeley, Dashboard Cam

Other Segmentation Datasets

- MS COCO
- ADE20K
- SUN

Some examples from Cityscapes

Diversity & Unstructured Conditions

Odd-shaped vehicles, challenging drivable areas

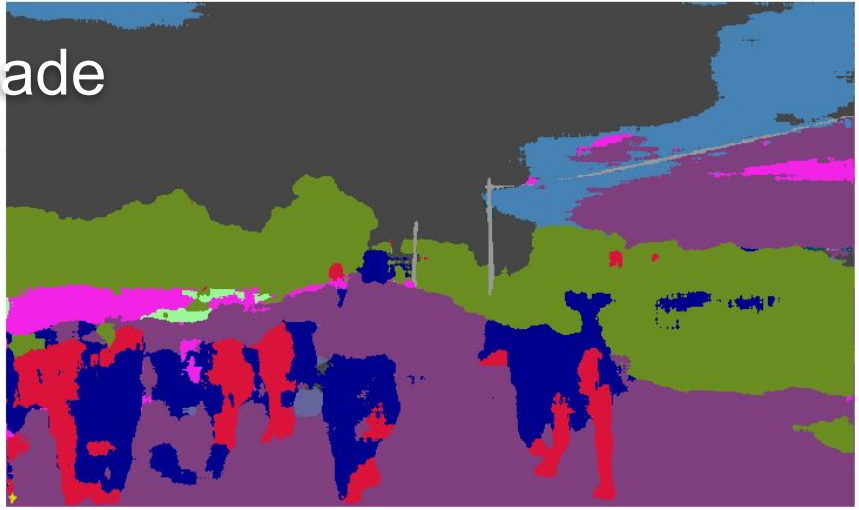


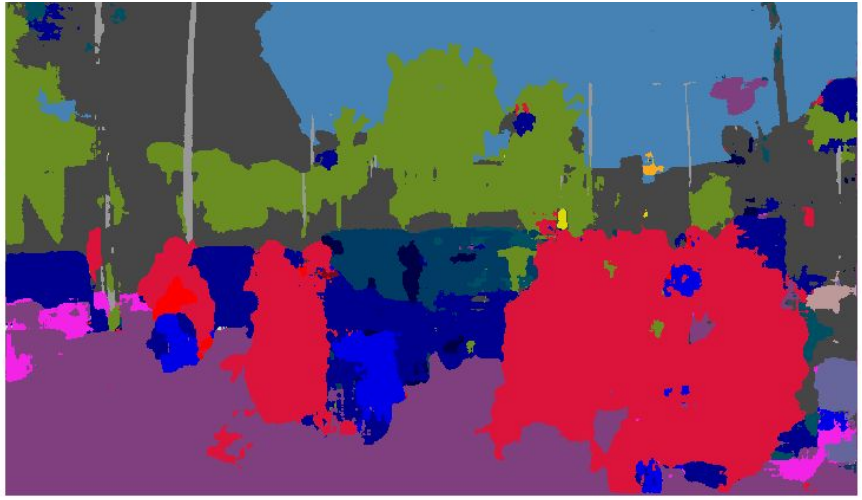
Pedestrians & jay-walkers



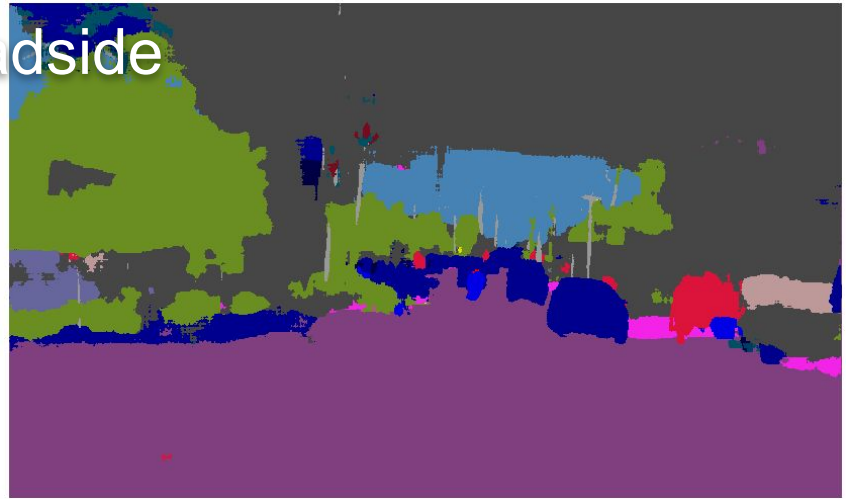


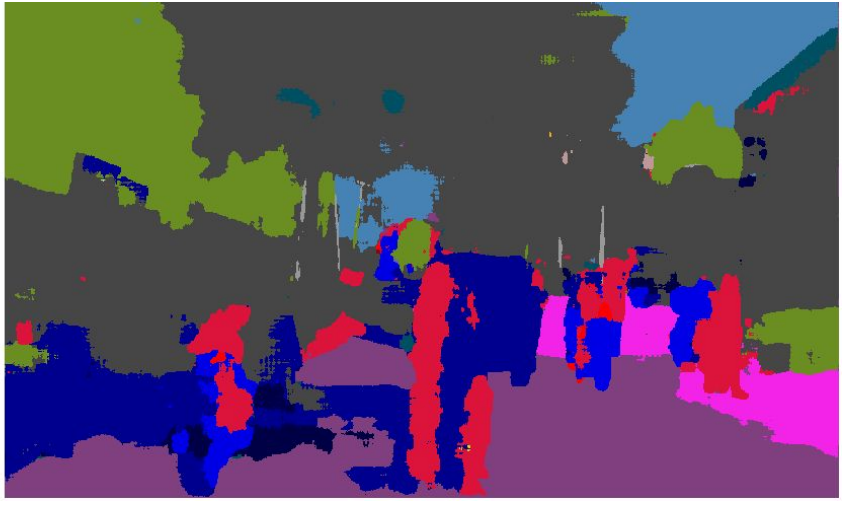

Shade



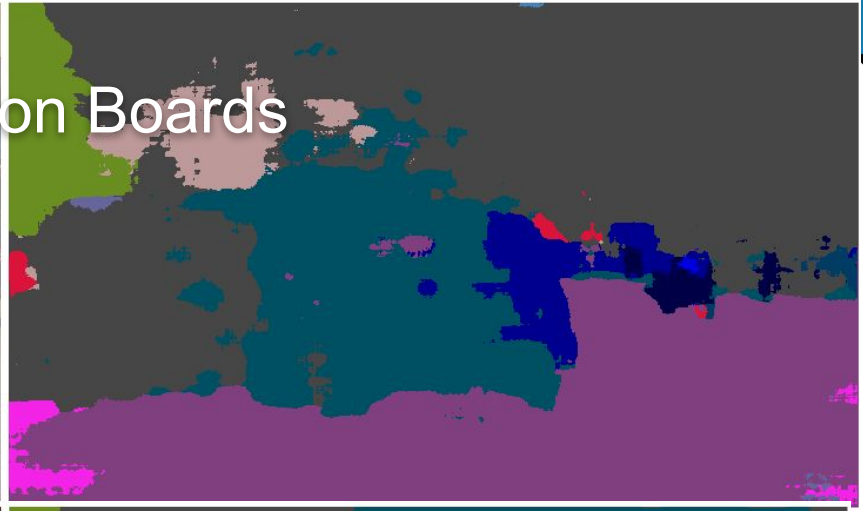


Roadside

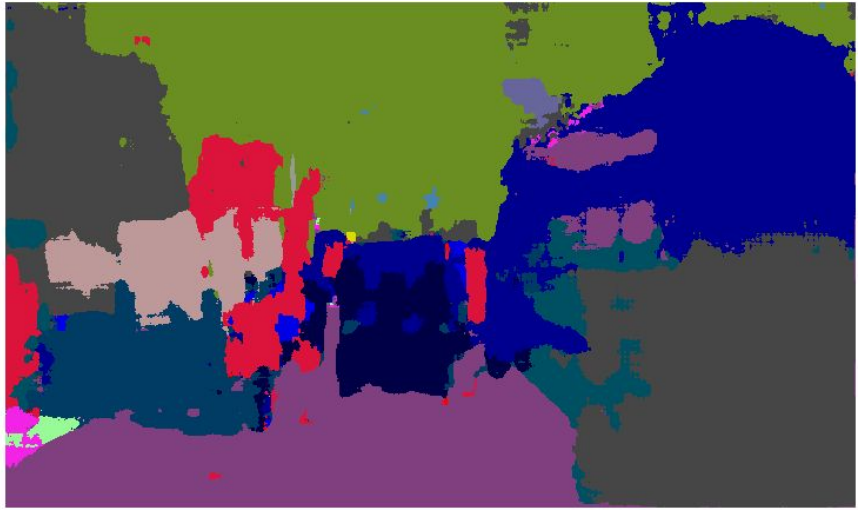




Information Boards



Unstructured Traffic



IDD

IDD: Vital Stats

- 10,004/50,000 images from 182 drive sequences annotated With pixel level/bounding box labels.
- From Bangalore and Hyderabad

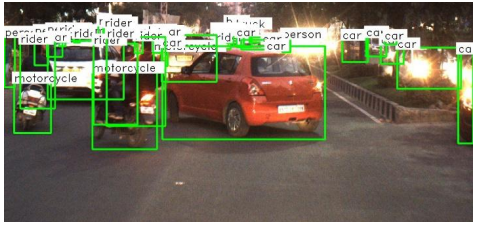
Dataset	Calibration	Nearby frames / Video	Distortion /Night	#Images/ #Sequences	#Labels Train/Total	Average Resolution
Cityscapes [5]	✓	✓		5K / 50	19/34	2048x1024
IDD	✓	✓		10K / 180	30/34	1678x968
BDD100K [26]		✓	✓	10K / 10K	19/30	1280x720
MVD [16]				25K / -	65/66	>1920x1080

Table 1. Comparison of semantic segmentation datasets for autonomous navigation.

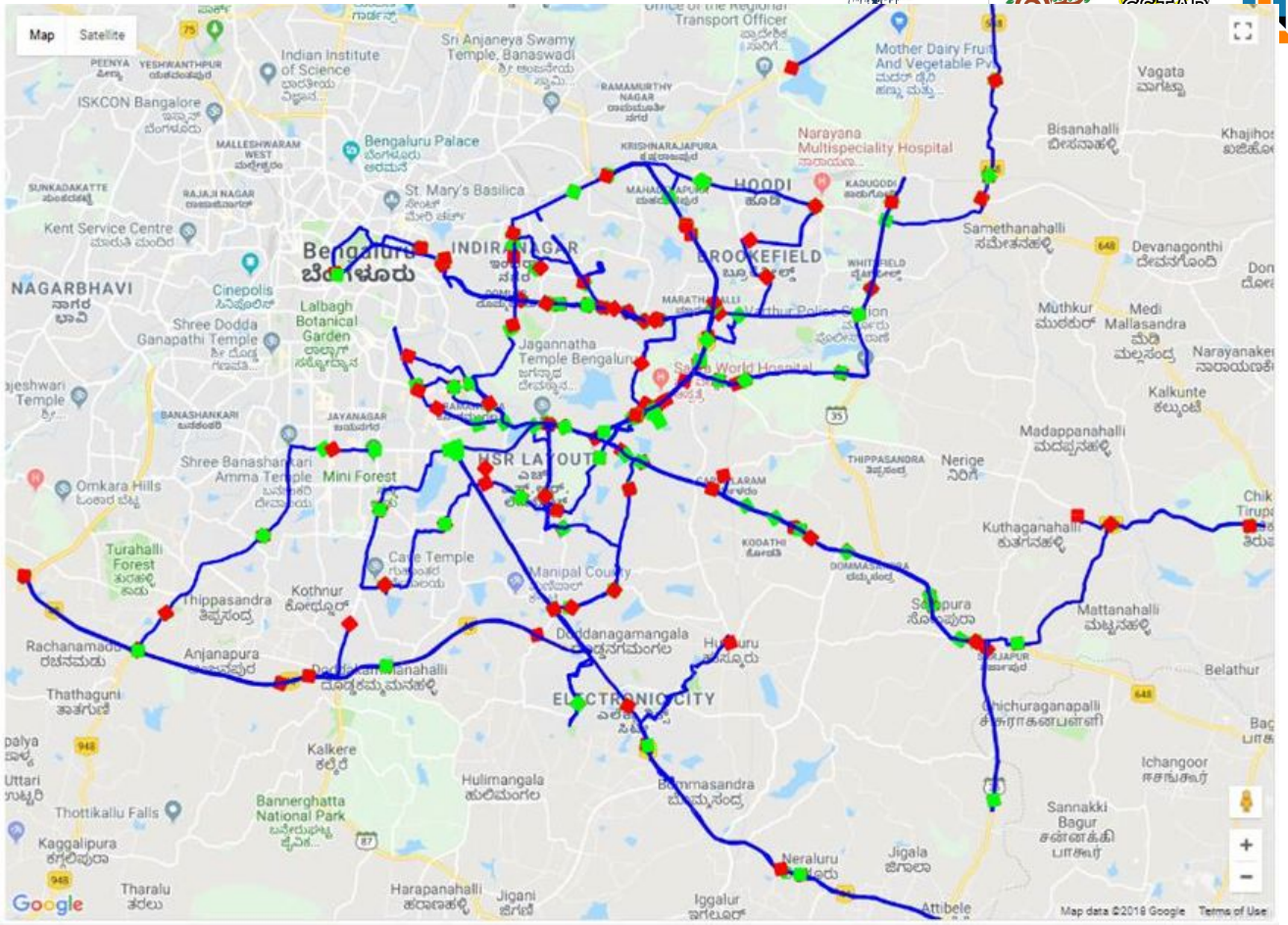
Unstructured Driving Conditions – Data Collection

Input

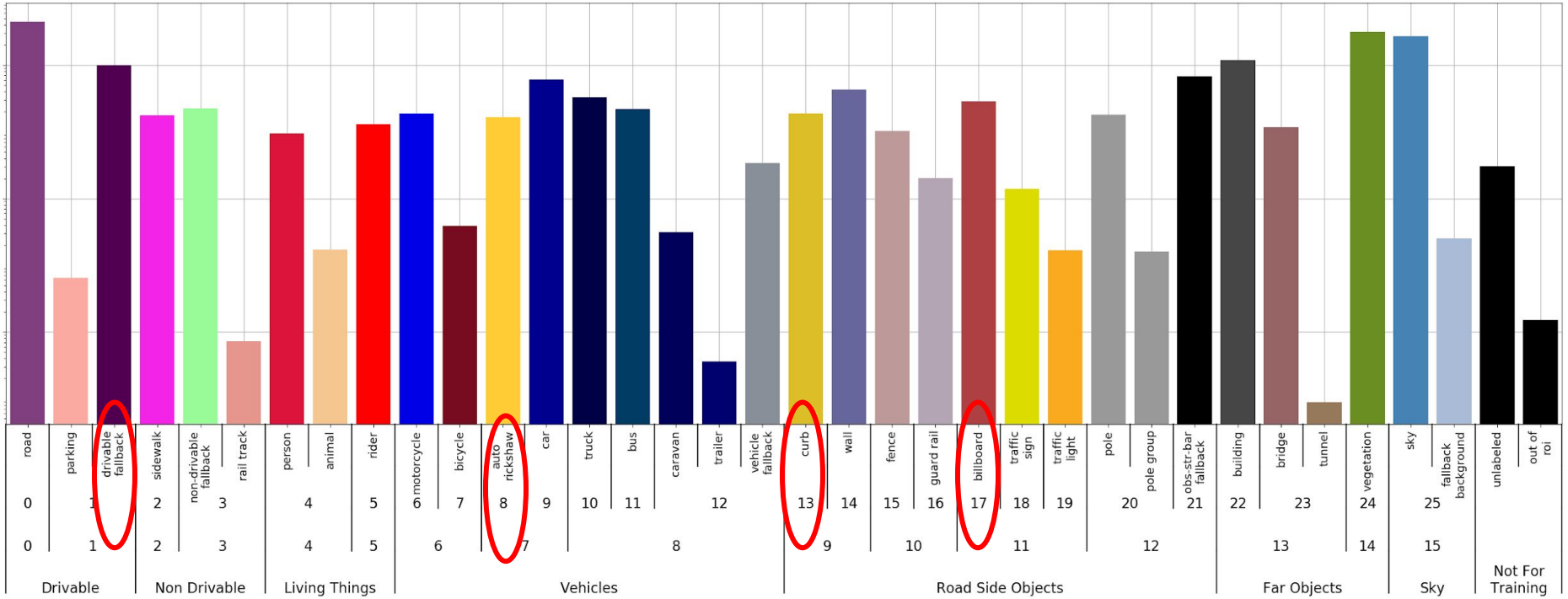
Coarse



Fine



AutoNUE Data Diversity – New Objects on Roads

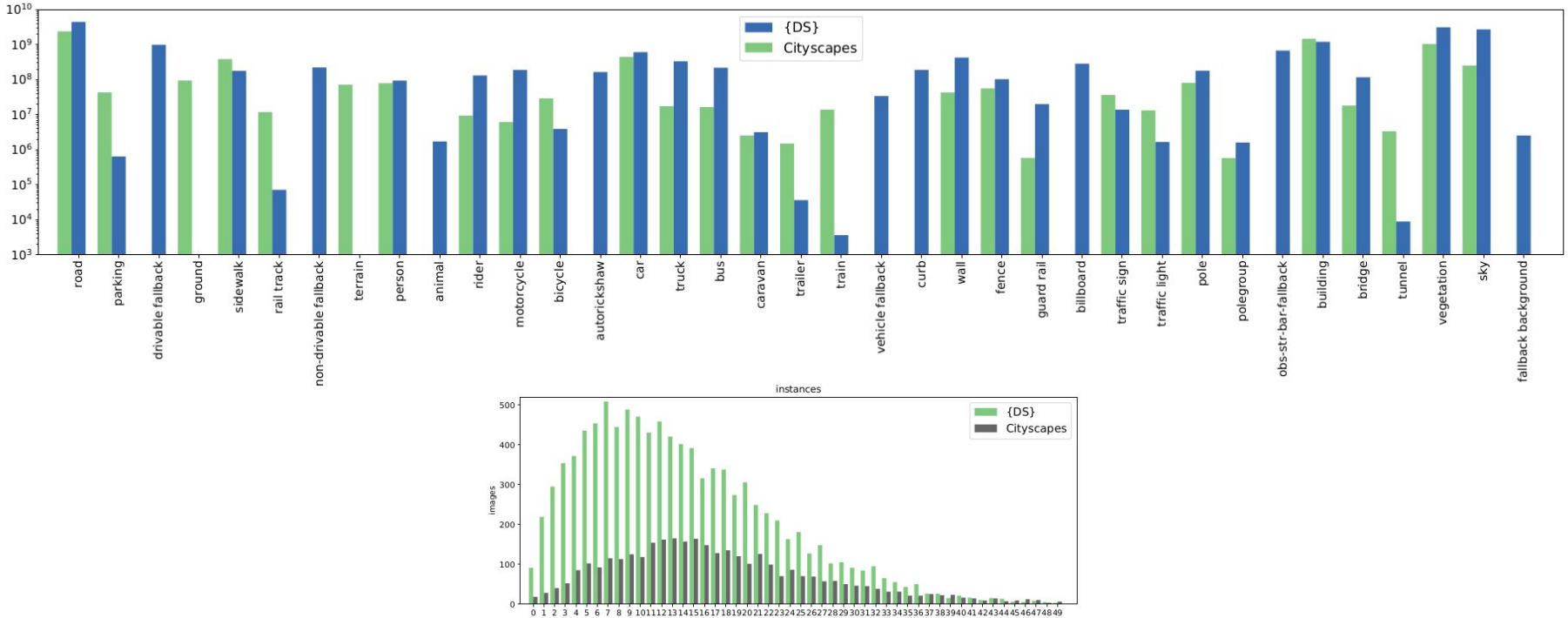


http://cvit.iiit.ac.in/scene-understanding-challenge-2018/overview.php

Label Hierarchy

- 4 Level label Hierarchy
  - L4: 30 labels
  - L3: 26 labels
  - L2: 16 labels
  - L1: 7 labels
- Splits
  - 7000 Train
  - 2000 Test
  - 1000 Validation
  - Procedure
    - Choose randomly subset of 182 sequences
    - Check if for each label pixels are split into 70%,20%,10%
    - If not repeat.

Comparison with Cityscapes



gure 8. Comparison of traffic participants in our dataset wi
tyscapes.

Trained Model

IoUs: Intersection over Union

Method	% mIoU at Levels		
	L1	L2	L3
ERFNet	-	-	55.4
DRN-D-38	85.9	72.6	66.6
*DeeplabV3+ [4]	89.8	78.0	74.0
*PSPNet [27]	89.9	78.0	74.1
*Wider Resnet-38, DeeplabV3 Decoder, Inplace ABN [20], Ensemble of 4	89.7	77.9	74.3

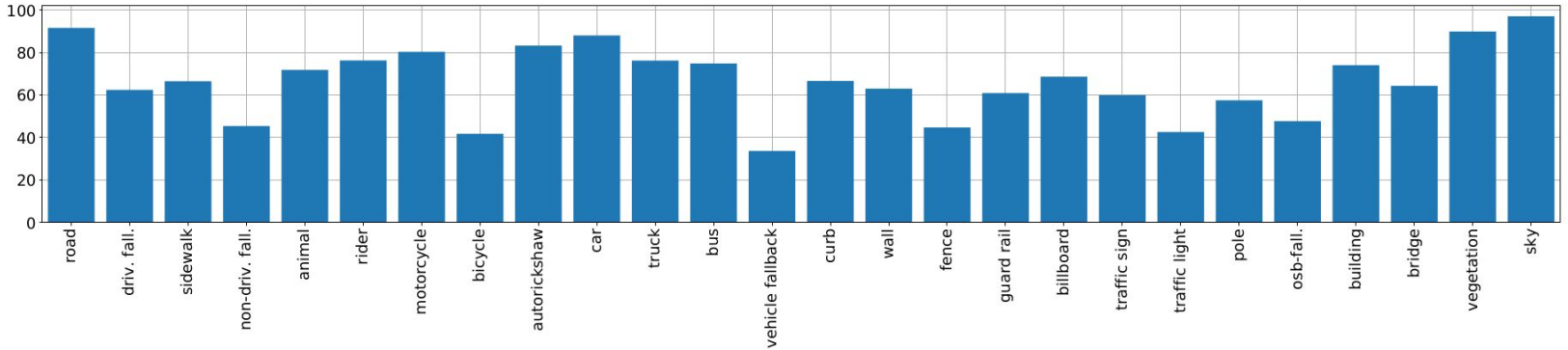


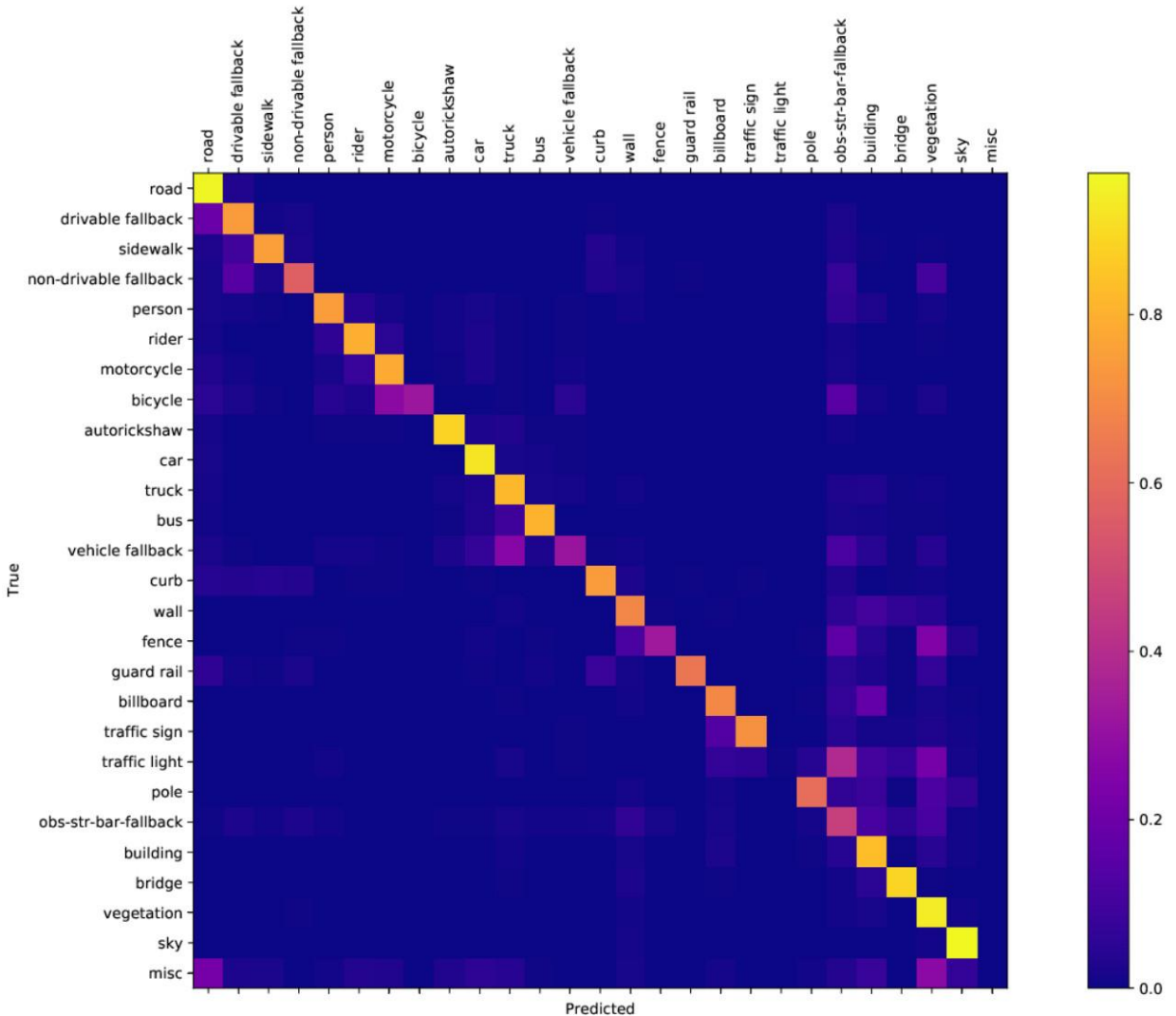
Figure 9. The IoUs for every class for the DRN D 38 model trained on IDDwith mIoU of 66.5%.

Domain Discrepancy

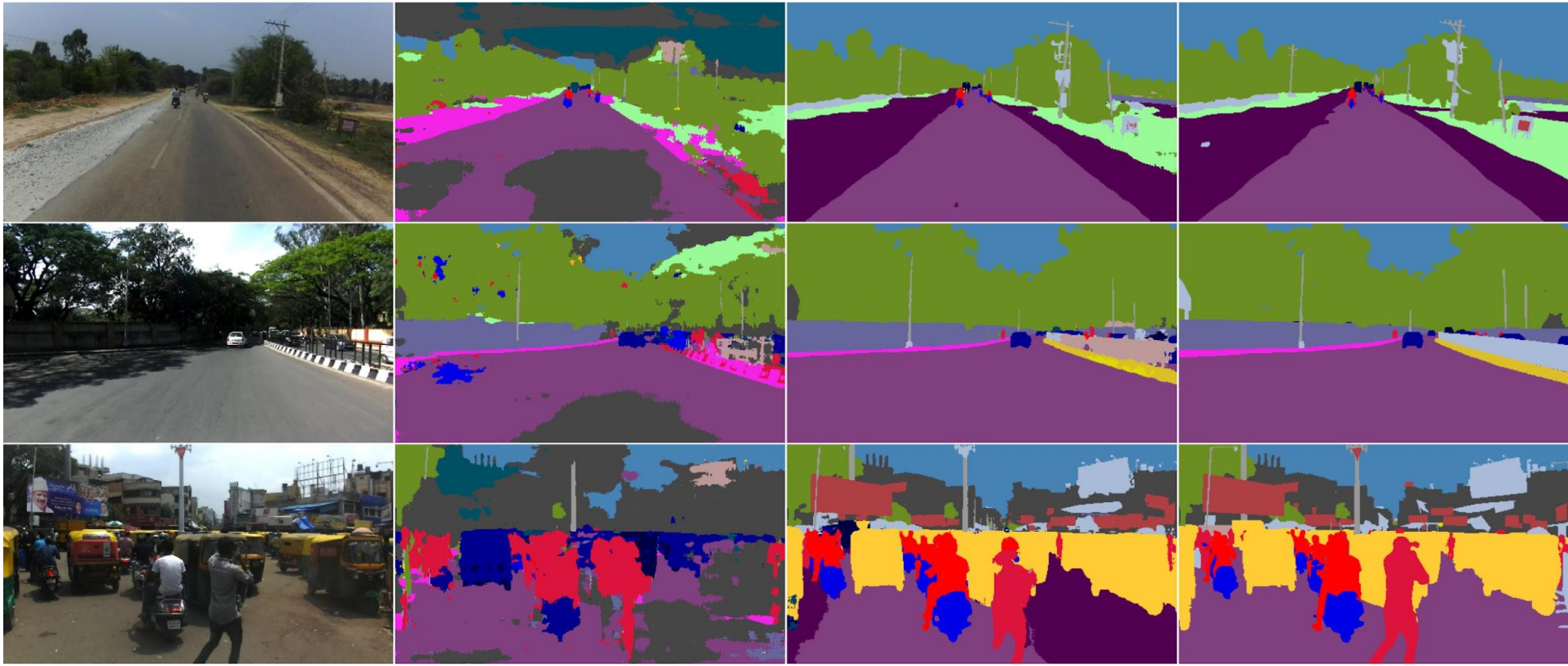
Train	Test	road	sidewalk	person	motorcycle	bicycle	car	truck	bus	wall	fence	traffic sign	traffic light	pole	building	vegetation	sky	mIoU of common labels
CS	DS	72	22	30	47	10	58	30	19	17	13	19	8	23	32	76	68	34
DS	CS	81	26	74	34	55	85	16	17	21	24	25	21	47	77	90	88	49
BD	ID	83	0	38	44	2	52	21	13	0	0	0	0	36	42	83	94	32
ID	BD	84	16	57	34	44	77	14	24	10	33	18	13	41	68	82	87	44
CS	CS	98	84	81	60	76	94	56	78	49	58	77	67	62	92	92	94	76
MV	MV	85	58	73	55	61	90	61	65	45	58	72	67	50	86	90	98	70
ID	ID	92	68	73	80	42	89	79	78	64	45	60	38	58	75	90	97	70
BD	BD	95	62	61	32	22	90	52	57	25	45	52	58	49	85	87	97	60

Table 3. The domain discrepancy between Cityscapes (CS) [5], Mapillary Vistas (MV) [16], Berkeley Deepdrive (BD) [26] Dataset and IDD (ID) using the DRN-D-38 Model [25]. Performance for only the common labels between the four datasets are used. First two rows compares the accuracy of a model trained on one of IDD or Cityscapes and tested on the other dataset. As can be seen, IDD trained model can predict CS and BD labels, better than predictions of trained models of the corresponding datasets on IDD. The bottom four rows gives the performance of models in each of the datasets. IDD dataset is harder than CS dataset and similar in hardness to MV on these 16 labels. BD is harder because i.) it has night scenes ii.) the images are take from a dash board cam, hence has reflections from inside the car as well as distortions like rain drops on the mirror.

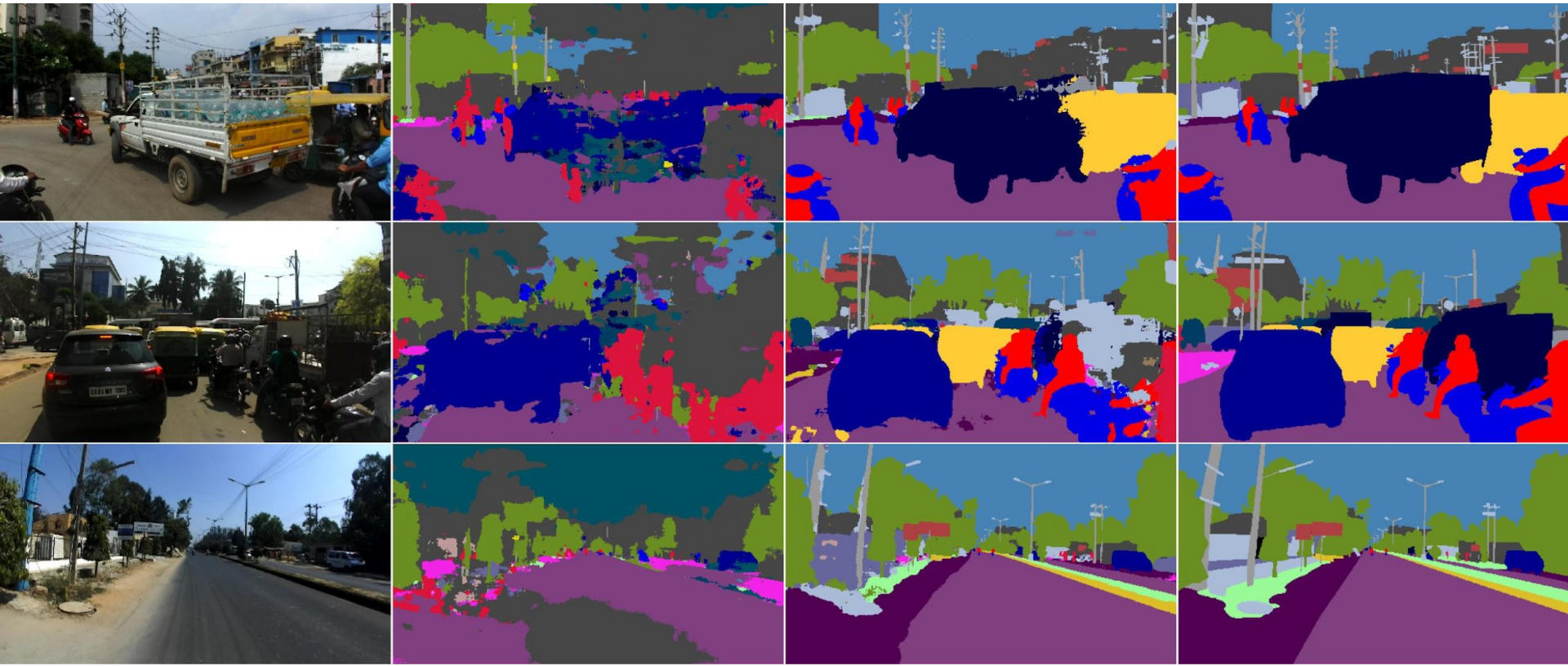
Confusion Matrix



Qualitative Examples



Qualitative Examples

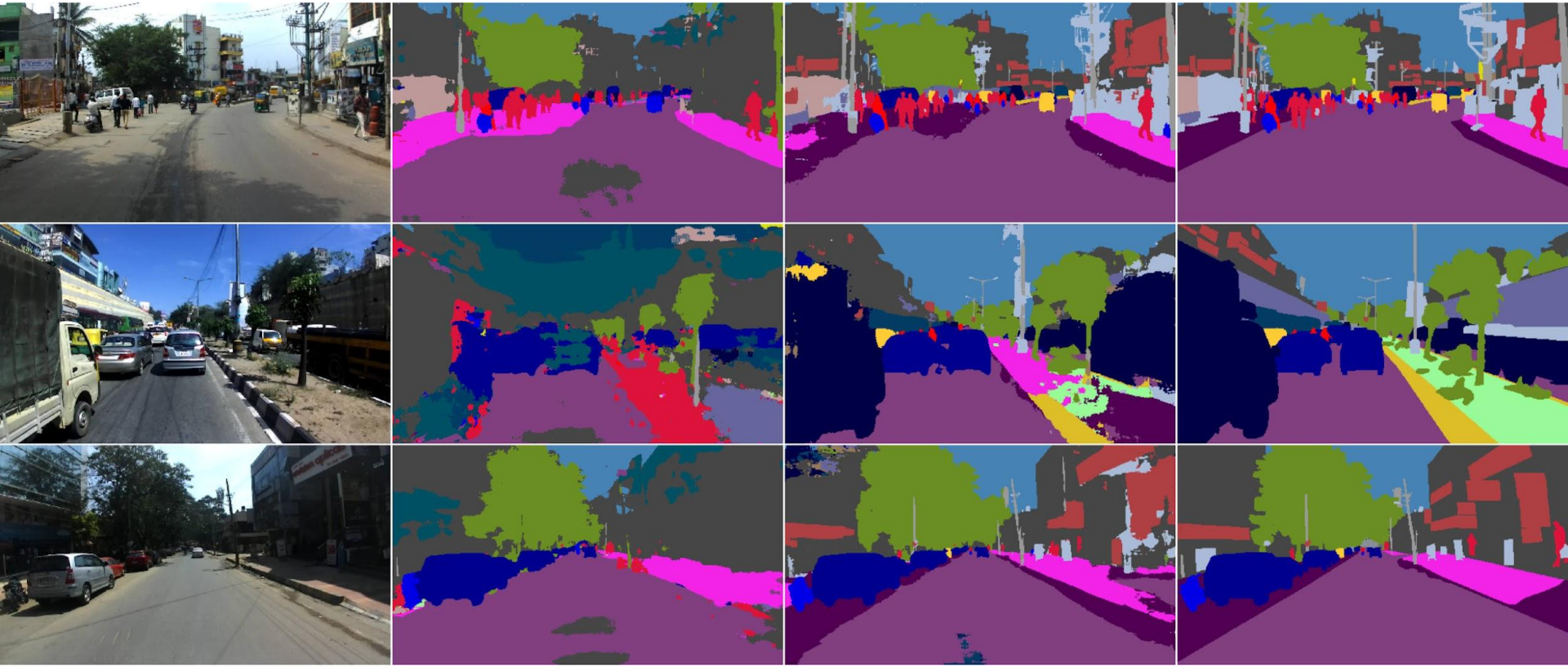


INTERNATIONAL INSTITUTE OF INFORMATION TECHNOLOGY
HYDERABAD

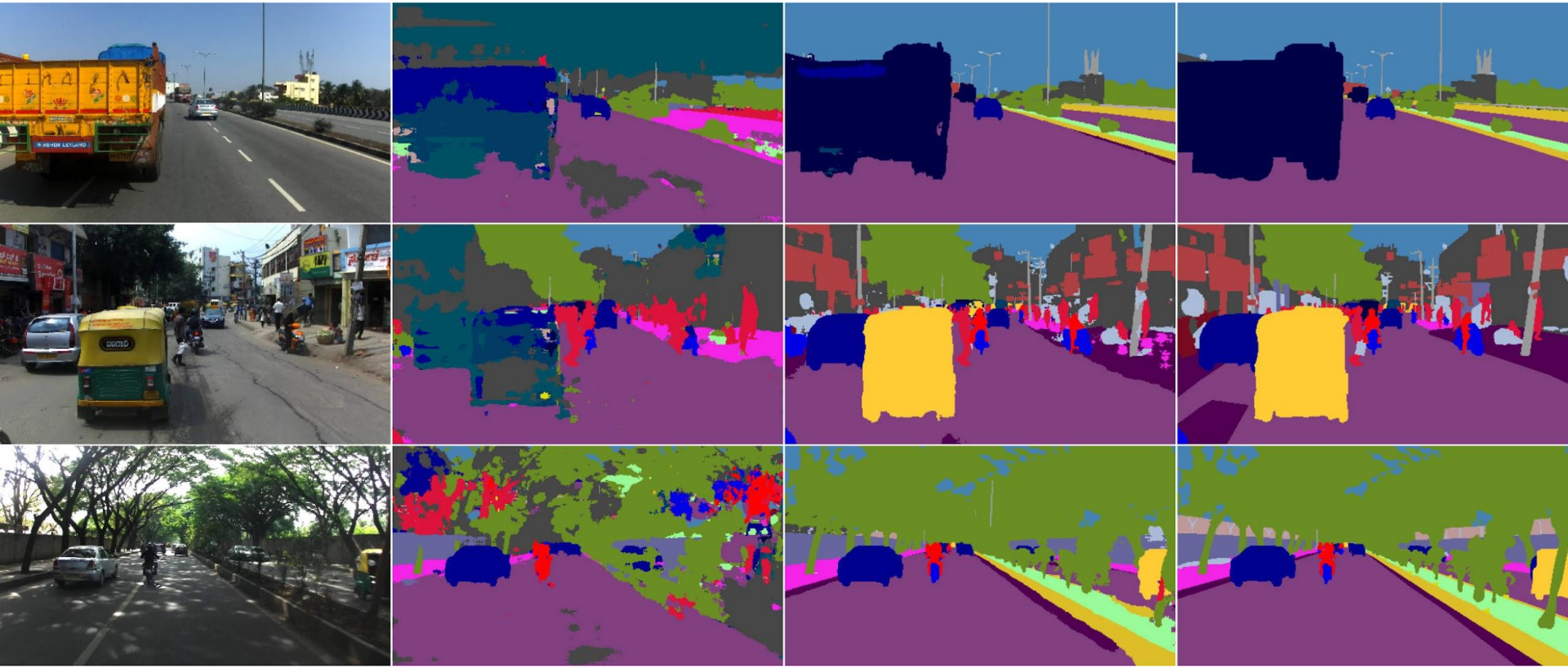
IIIT Hyderabad

CSTAR

Qualitative Examples



Qualitative Examples



AutoNUE Challenge at ICCV 2019

https://cvit.iiit.ac.in/autonue2019/challenge/

Challenge will be open from August 1st.

New larger dataset with challenging conditions.

http://idd.insaan.iiit.ac.in/

http://cvit.iiit.ac.in/scene-understanding-challenge-2018/ (last years challenge).

Some Interesting Links

https://davischallenge.org/

http://wad.ai/

http://robustvision.net/

http://cvit.iiit.ac.in/autonue2018/

Slides borrowed from

http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture11.pdf

http://imatge-upc.github.io/telecombcn-2016-dlcv/slides/D4L2-segmentation.pdf

http://cs231n.stanford.edu/slides/2016/winter1516_lecture13.pdf

http://deeplearning.csail.mit.edu/instance_ross.pdf

http://www.cs.princeton.edu/courses/archive/spring18/cos598B/public/outline/Amodal%20and%20Panoptic%20Segmentation.pdf

That's all Folks!
Thank You.