Convolutional Neural Network

Computer Vision and Artificial Intelligence

Dr Varun Ojha

varun.ojha@ncl.ac.uk

School of Computing

Newcastle University

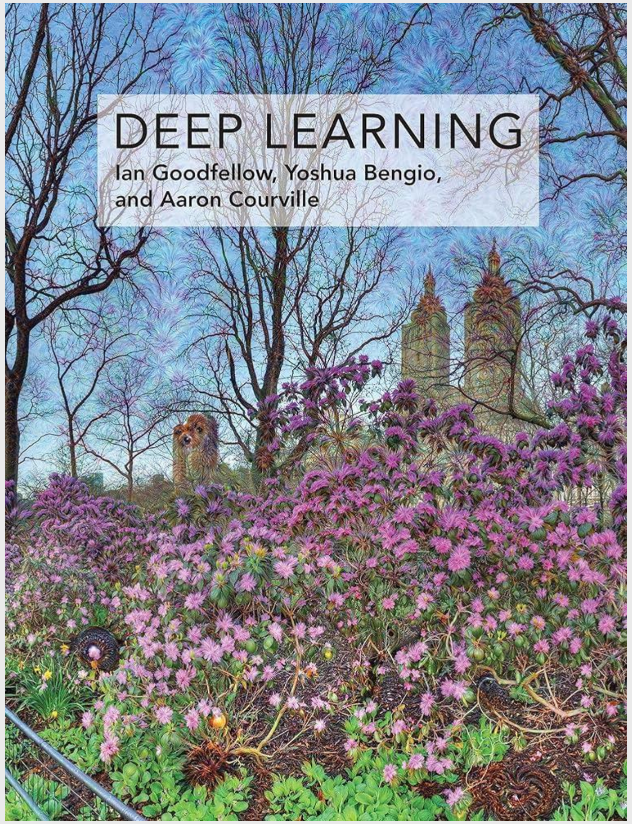
Learning objectives (Convolutional Nets)

By the end of this week, you will be able to:

- Learn the concepts of convolutional neural networks (CNNs or ConvNets)
- Design various convolutional architectures
- Understand computer vision tasks and models:
  - image classification
  - image segmentation
  - object detection
  - Image generation
- Apply and evaluate a ConvNet on image classification task.

Content of this week (CNNs)

- **Part 1: Design of Convolutional Nets**
  - Image Data
  - Components of ConvNets
  - Regularisation in ConvNets / DNNs
  - ConvNet Architectures
- **Part 2: Convolutional Neural Nets Applications and Models**
  - Image Segmentation Models
  - Object Detection Models
  - Generative Models Concept
- **Part 3: Practical Exercise (CNN)**



Goodfellow et al (2017) Deep Learning, MIT Press
https://www.deeplearningbook.org/

Part 1
Design of Convolutional Nets

Data

Image: Gary scale

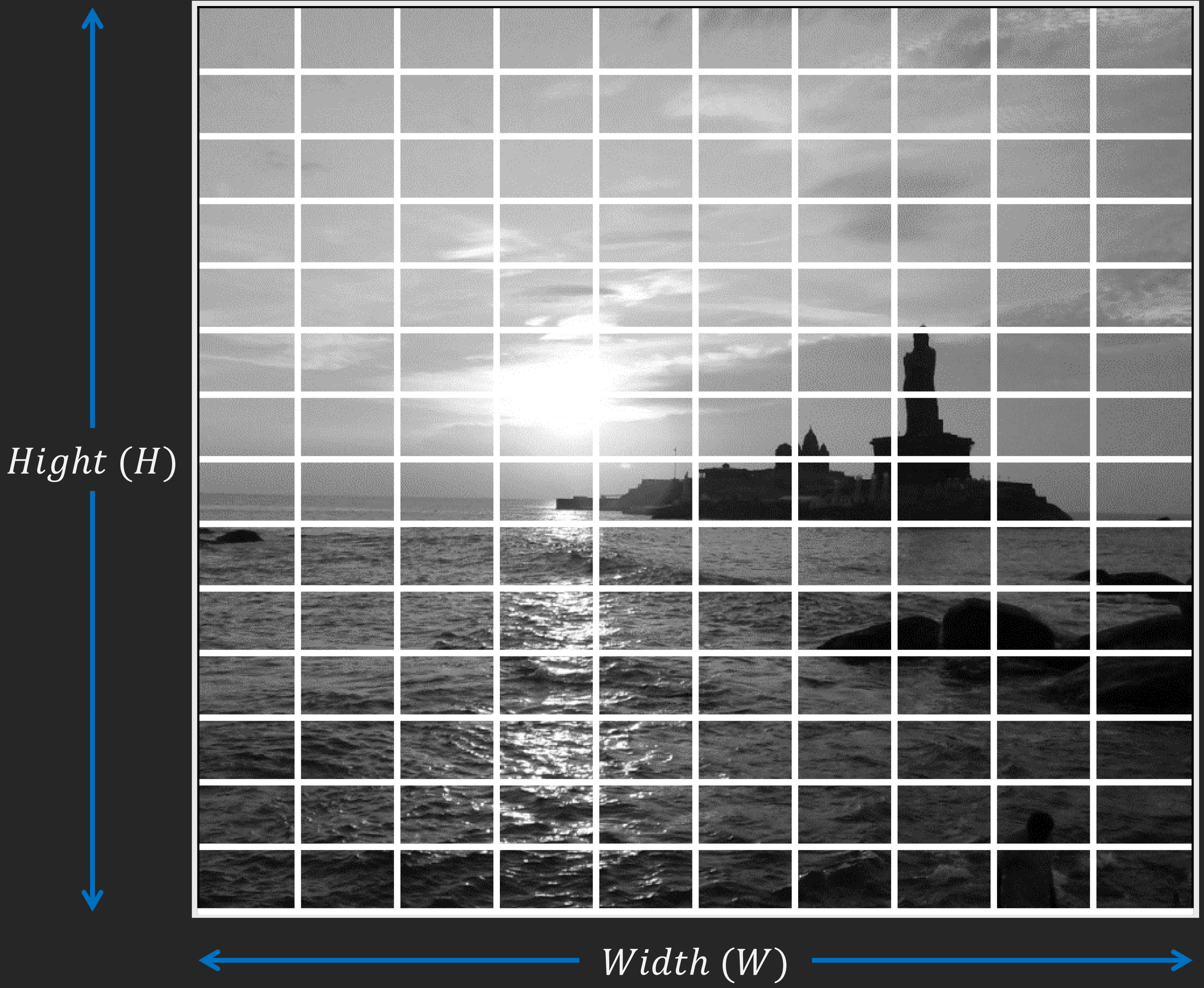
Data: 2D

Image: Gary scale

$$I = \begin{bmatrix} \boldsymbol{p}_{11} & \cdots & \boldsymbol{p}_{1,W} \\ \vdots & \ddots & \vdots \\ \boldsymbol{p}_{H,1} & \cdots & \boldsymbol{p}_{H}, \boldsymbol{p}_{W} \end{bmatrix}$$

For $Hight = 256, Width = 256$

$$I = \begin{bmatrix} p_{11} & \cdots & p_{1,256} \\ \vdots & \ddots & \vdots \\ p_{256,1} & \cdots & p_{256}, p_{256} \end{bmatrix}$$



Data: 2D

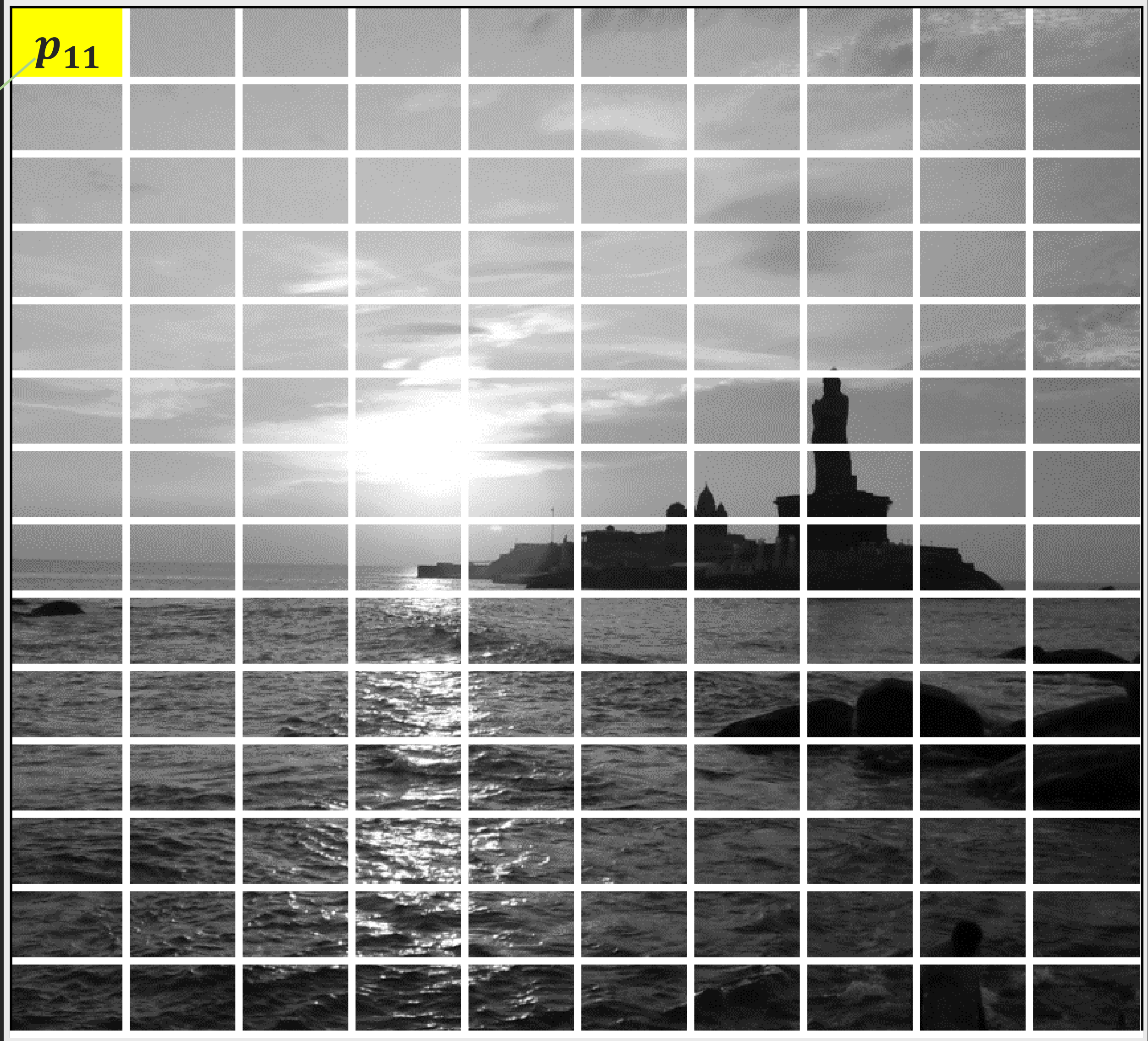
Image: Gary scale

$$p_{ij} \in \{0,1,2.,\ldots,256\}$$

For $Hight = 256, Width = 256$

$$I = \begin{bmatrix} p_{11} & \cdots & p_{1,256} \\ \vdots & \ddots & \vdots \\ p_{256,1} & \cdots & p_{256}, p_{256} \end{bmatrix}$$

$Hight\ (H)$



$Width\ (W)$

Data

Image: Colour

Data: 3D

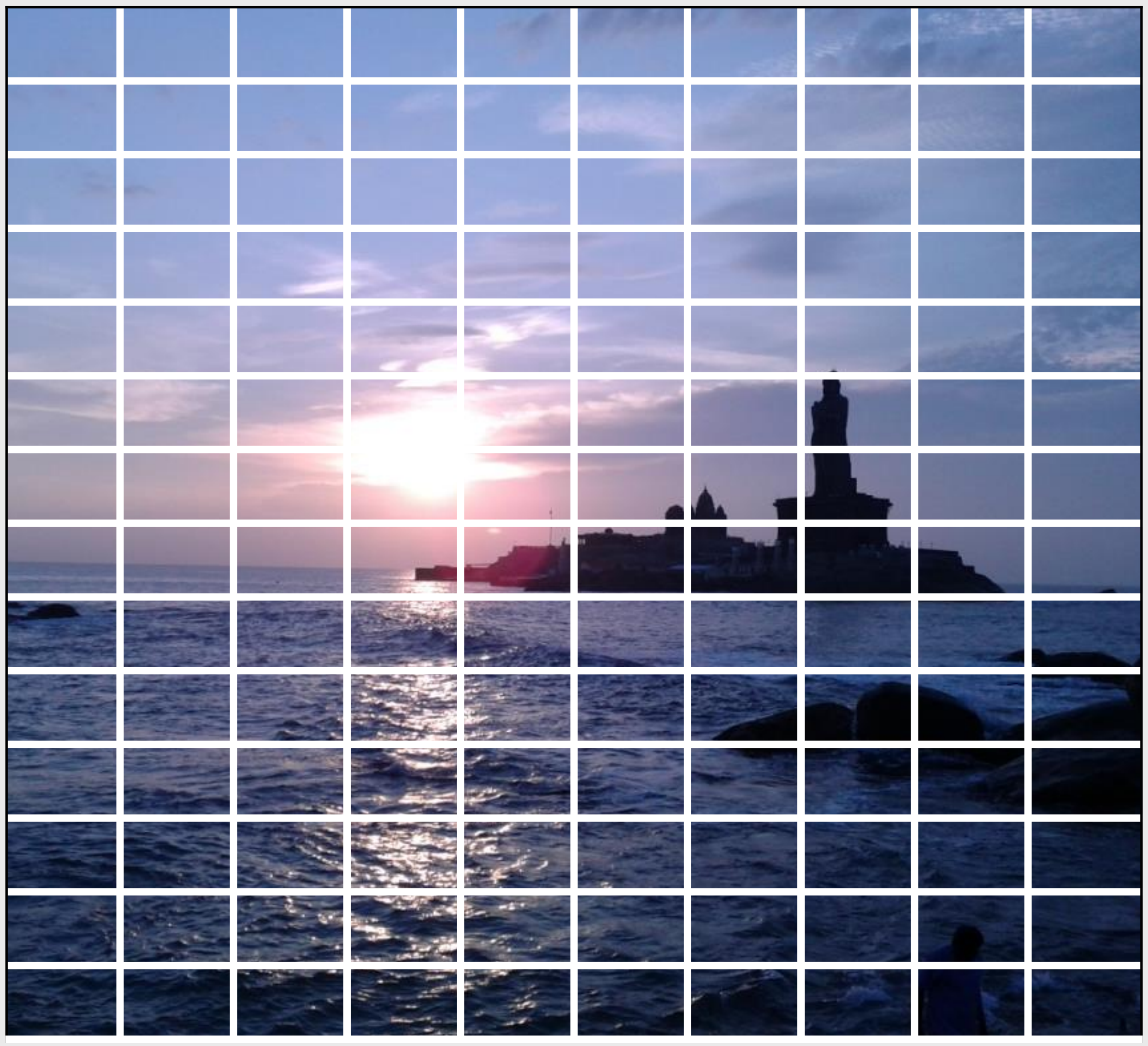
Image: Colour

$$I_{RED} = \begin{bmatrix} p_{11} & \cdots & p_{1,W} \\ \vdots & \ddots & \vdots \\ p_{H,1} & \cdots & p_{H}, p_{W} \end{bmatrix}$$

$$I_{Green} = \begin{bmatrix} p_{11} & \cdots & p_{1,W} \\ \vdots & \ddots & \vdots \\ p_{H,1} & \cdots & p_{H}, p_{W} \end{bmatrix}$$

$$I_{Blue} = \begin{bmatrix} p_{11} & \cdots & p_{1,W} \\ \vdots & \ddots & \vdots \\ p_{H,1} & \cdots & p_{H}, p_{W} \end{bmatrix}$$

$Hight\ (H)$



$Width\ (W)$

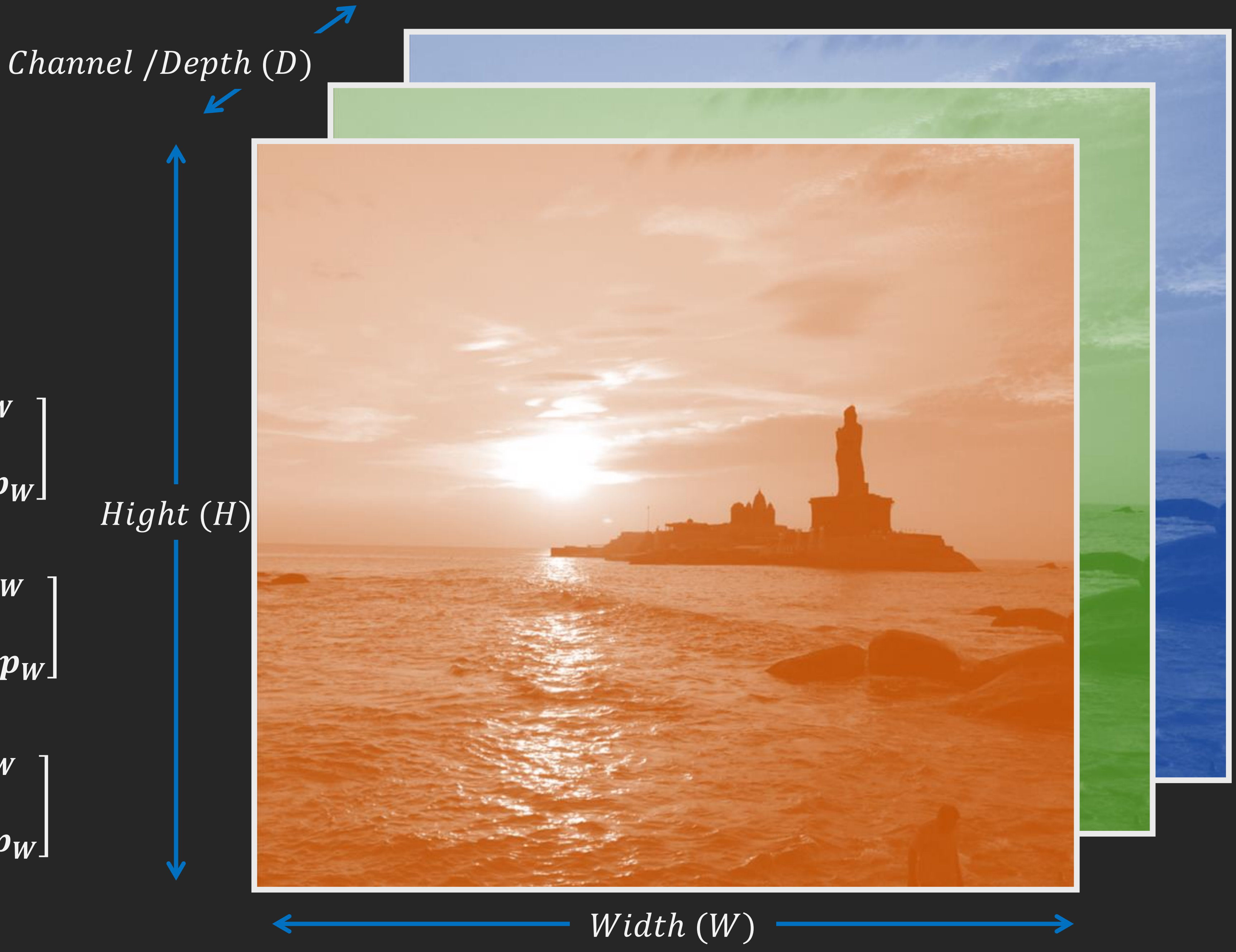
Data

Image: Colour

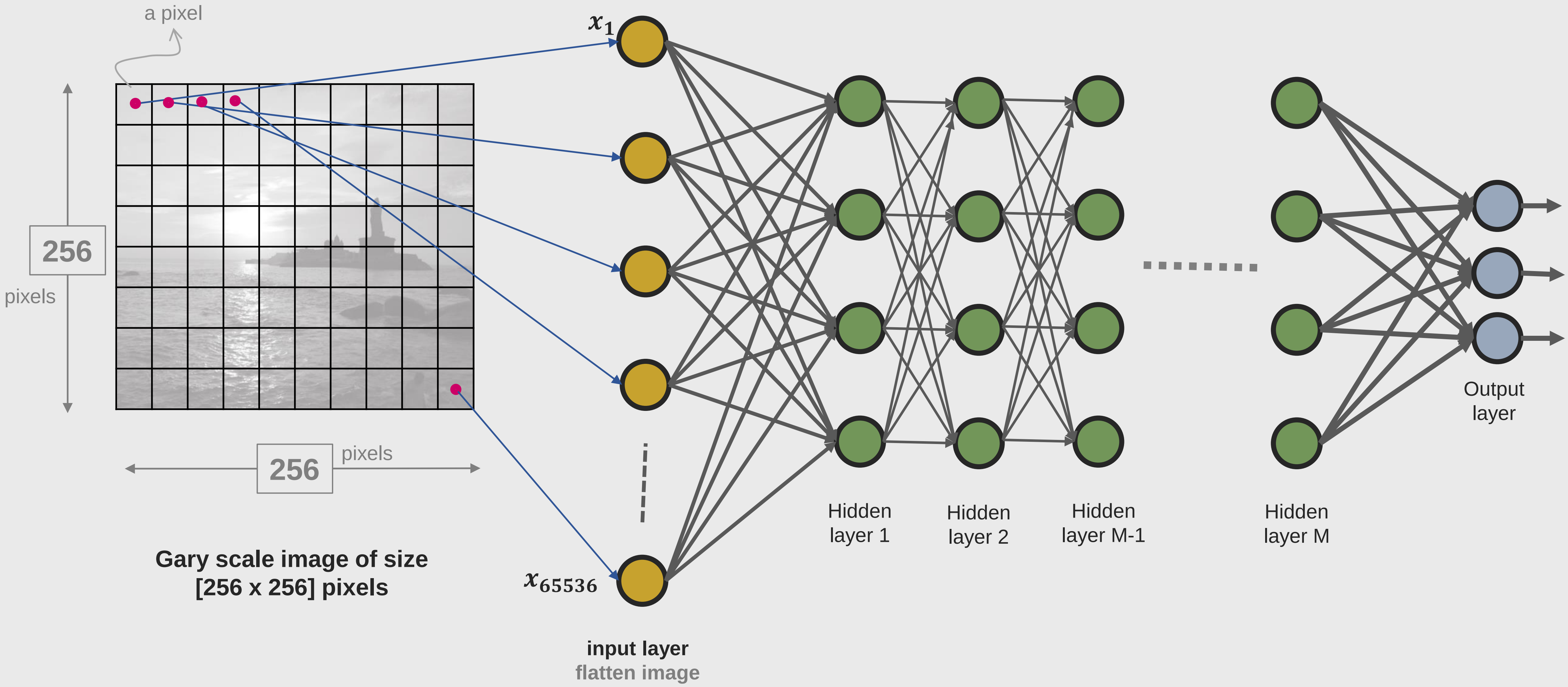
$$I_{RED} = \begin{bmatrix} p_{11} & \cdots & p_{1,W} \\ \vdots & \ddots & \vdots \\ p_{H,1} & \cdots & p_H, p_W \end{bmatrix}$$

$$I_{Green} = \begin{bmatrix} p_{11} & \cdots & p_{1,W} \\ \vdots & \ddots & \vdots \\ p_{H,1} & \cdots & p_H, p_W \end{bmatrix}$$

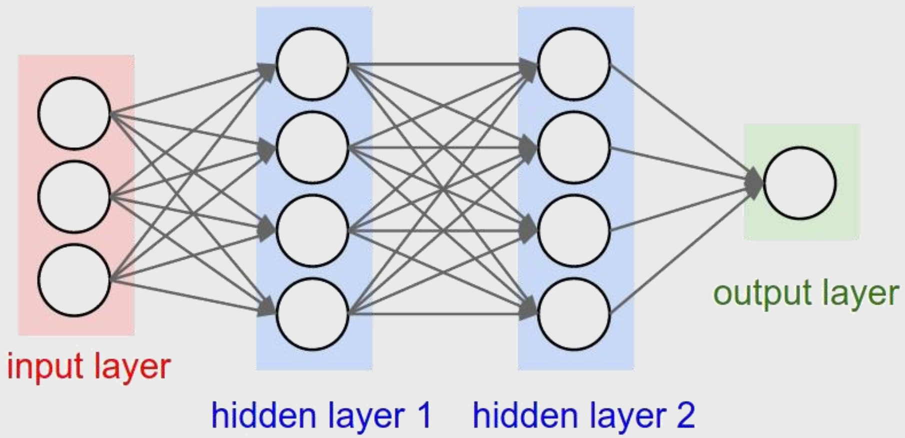
$$I_{Blue} = \begin{bmatrix} p_{11} & \cdots & p_{1,W} \\ \vdots & \ddots & \vdots \\ p_{H,1} & \cdots & p_H, p_W \end{bmatrix}$$

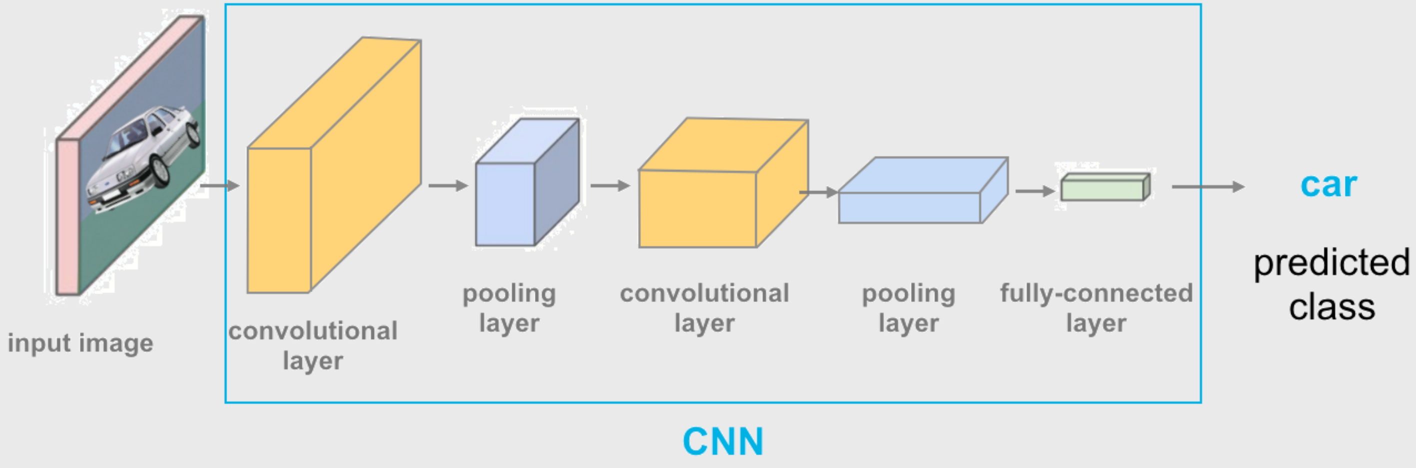


Deep Learning



Convolutional Neural Network (CNN)

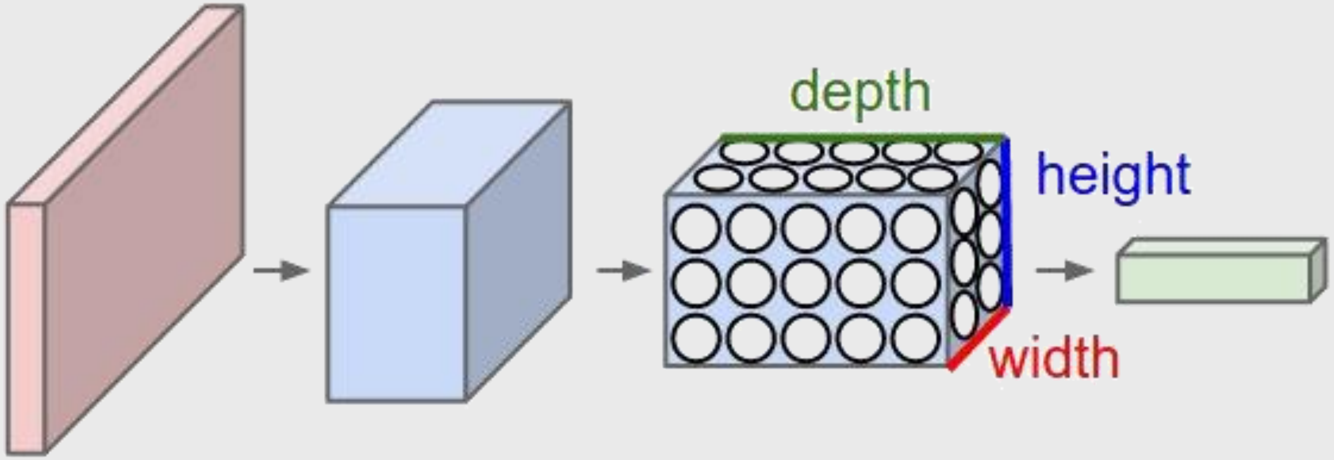




Deep Neural Network (DNN)

Convolutional Neural Network (CNN)

Convolutional Neural Network (ConvNet)



A **ConvNet** arranges its neurons in three dimensions (**width, height, depth**).

Every layer of a **ConvNet** transforms the 3D input volume to a 3D output volume of neuron activations.

In this example, the red input layer holds the image, so its width and height would be the dimensions of the image, and the depth would be 3 (Red, Green, Blue channels)

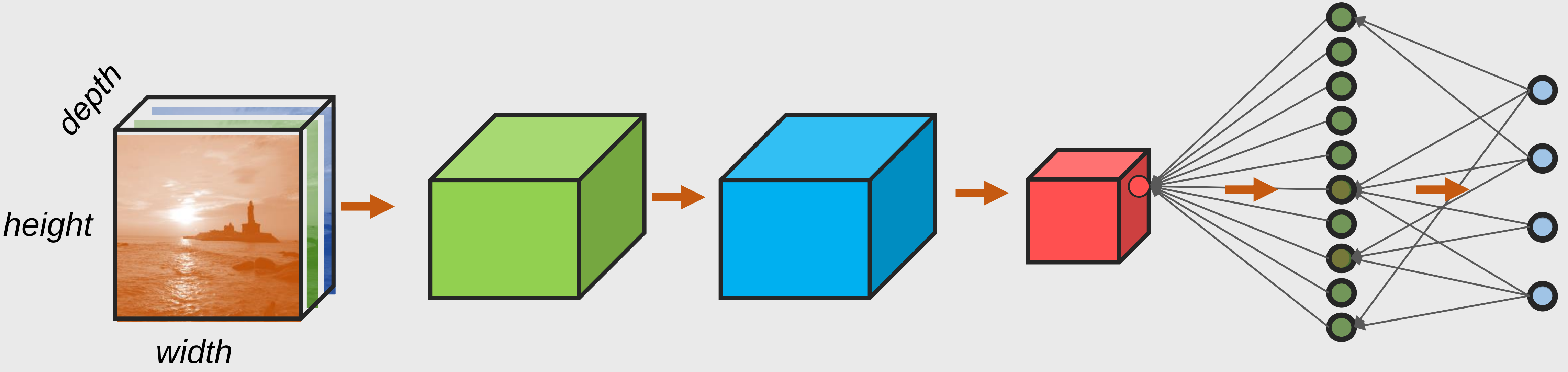
Ch 9 Goodfellow, Deep Learning, MIT Press

A very good source: http://cs231n.github.io/convolutional-networks/

ConvNet/ CNN

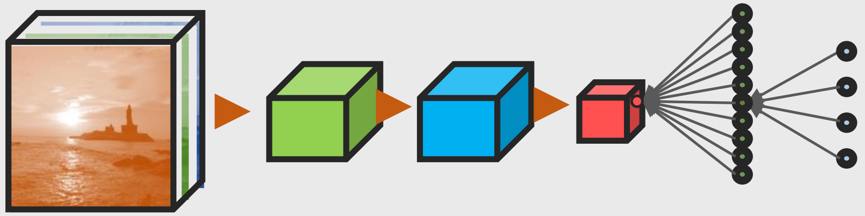
Architecture: A Simple ConvNet / CNN

[**INPUT** - CONV - RELU - POOL - FC]



[Input - Convolution - RELU - POOLING - Fully Connected - Output]

ConvNet/ CNN

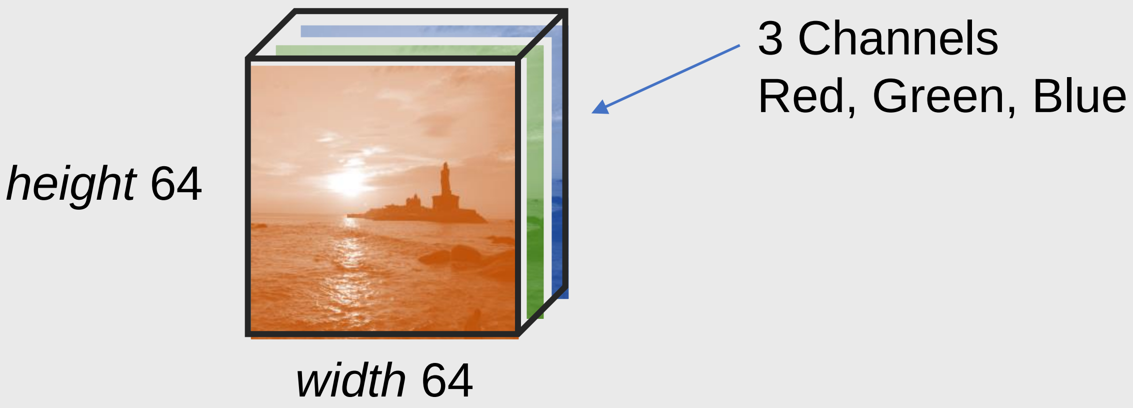


Architecture: A Simple ConvNet / CNN

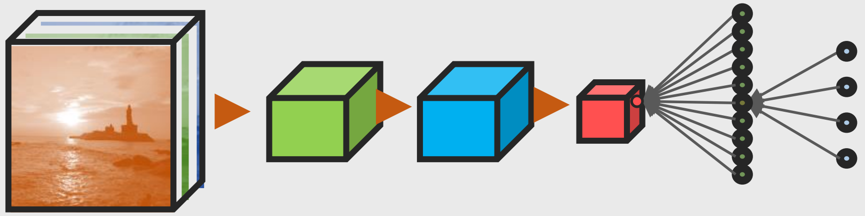
[INPUT - CONV - RELU - POOL - FC]

INPUT [64x64x3] holds the raw pixel values of the image.

Image *width* 64, *height* 64, and with *three* colour channels R,G,B.



A very good source: http://cs231n.github.io/convolutional-networks/



ConvNet/ CNN

Architecture: A Simple ConvNet / CNN

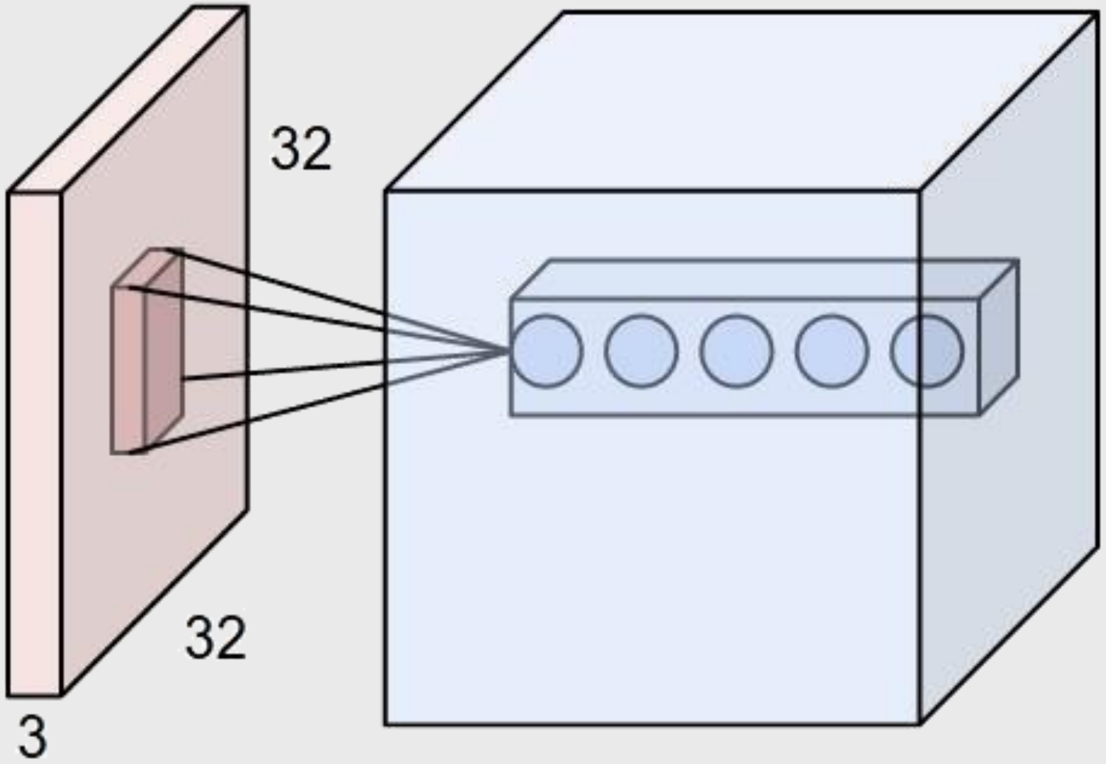
[INPUT - CONV - RELU - POOL - FC]

CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume.

E.g. The Convolution of INPUT [64x64x3] may result in volume [32x32x12] if we decided to use 12 filters

A very good source: http://cs231n.github.io/convolutional-networks/

ConvNet
Convolution Layer



An input volume in red (e.g. a 32x32x3), and an example volume of neurons in the first Convolutional layer.

A very good source: http://cs231n.github.io/convolutional-networks/

ConvNet
Convolution Layer

- CONV layer’s parameters consist of a set of **learnable filters**.

- Every filter is small spatially (along width and height) but extends through the full depth of the input volume.

- A typical filter on a first layer of a ConvNet might have size **5x5x3** (i.e. 5 pixels width and height, and 3 because images have depth 3, the colour channels)

A very good source: http://cs231n.github.io/convolutional-networks/

ConvNet

Convolution Layer

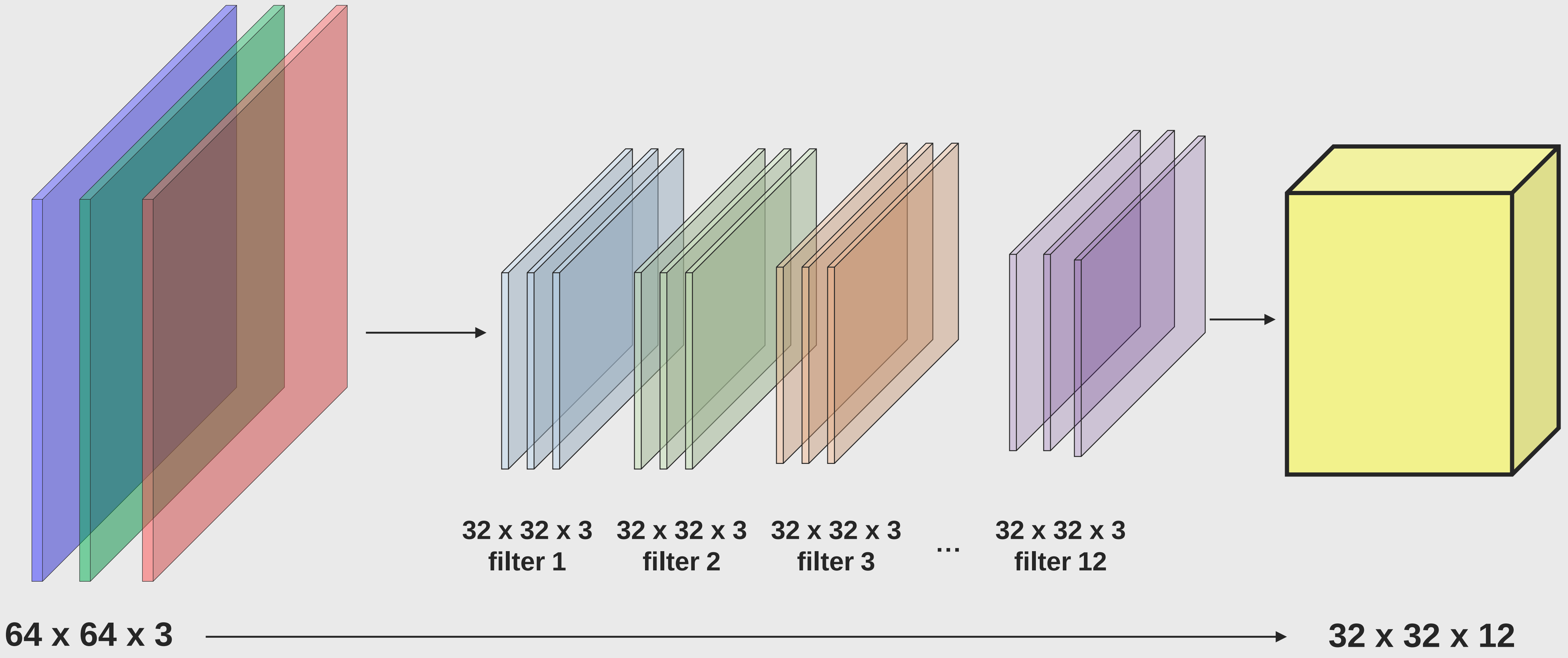
- Forward pass: we slide (**convolve**) each filter across the width and height of the input volume and compute dot products between the entries of the filter and the input at any position.

- When we slide the **filter** over the width and height of the input volume, we will produce a **2-dimensional activation map** that gives the responses of that filter at every spatial position

- We can have a set of filters (e.g., 12)

A very good source: http://cs231n.github.io/convolutional-networks/

ConvNet
Convolution Layer



ConvNet

Convolution Layer

Input volume of size $W_1 \times H_1 \times D_1$

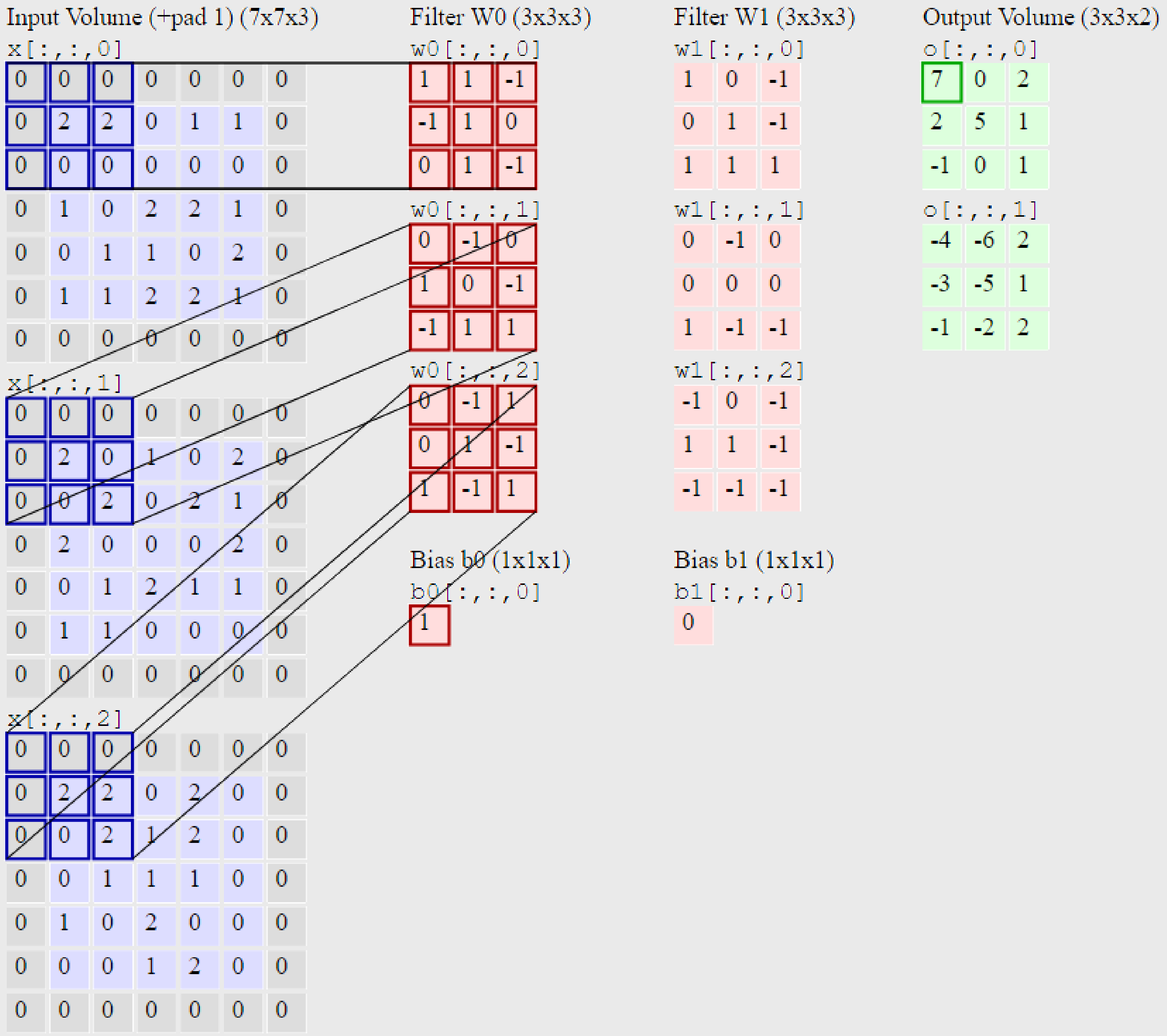
Requires four hyperparameters:

- Number of filters **K**,
- their spatial extent **F**,
- the stride **S**,
- the amount of zero padding **P**.

Output volume of size $W_2 \times H_2 \times D_2$

where:

- $\mathbf{W_2} = (W_1 - F + 2P)/S + 1$
- $\mathbf{H_2} = (H_1 - F + 2P)/S + 1$
- (i.e. width and height are computed equally by symmetry)
- $\mathbf{D_2} = K$

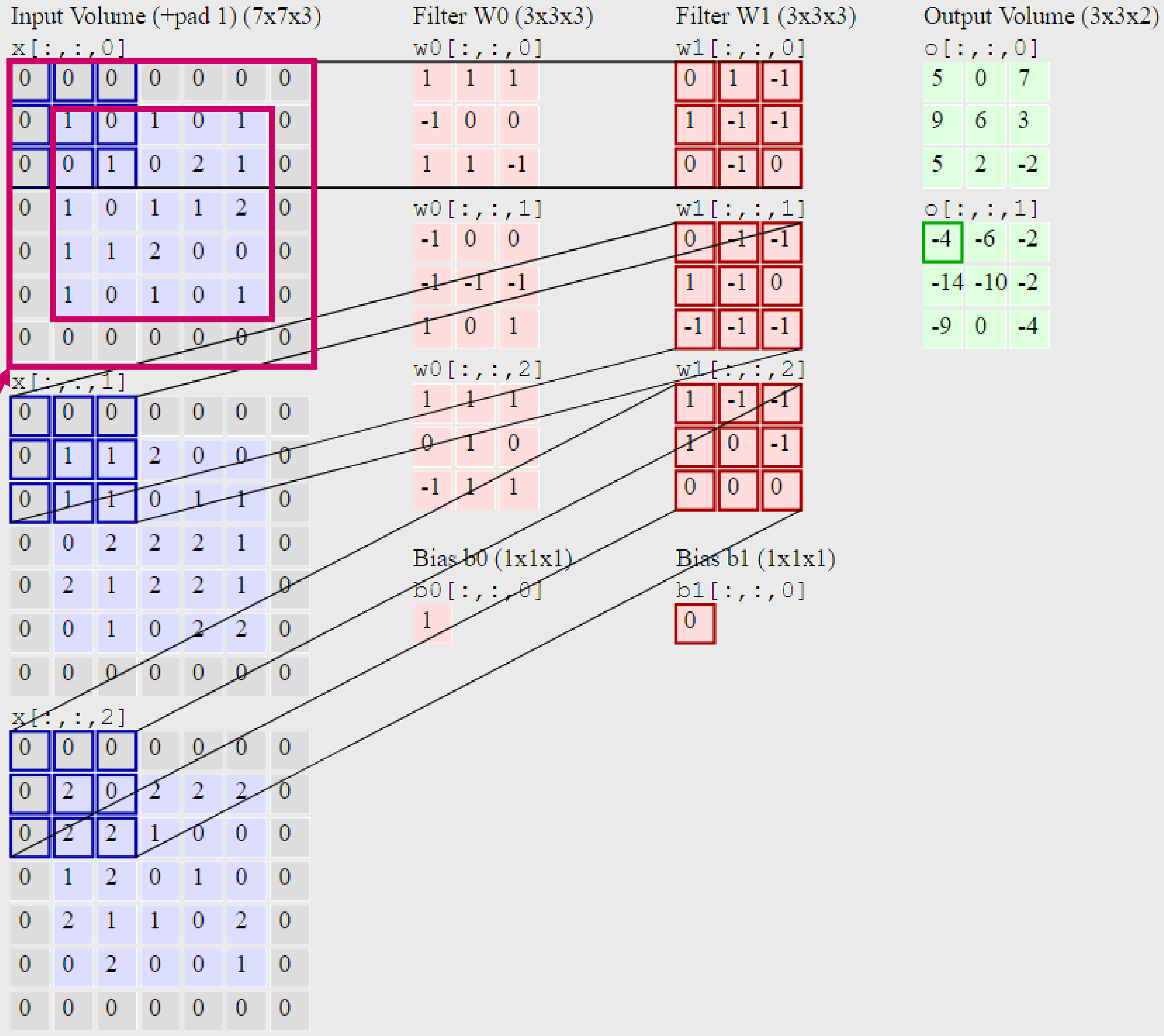


Source: http://cs231n.github.io/convolutional-networks/

ConvNet
Convolution Layer

Input volume of size 5×5×3
Requires four hyperparameters:
- Number of filters **K = 2**,
- their spatial extent **F = 3**,
- the stride **S = 2**,
- the amount of zero padding **P = 1**.

Output volume of size $W_2 \times H_2 \times D_2$
where:
- $W_2 = (5 - 3 + 2*1)/2 + 1$
- $H_2 = (5 - 3 + 2*1)/2 + 1$
- (i.e. width and height are computed equally by symmetry)
- $D_2 = 2$



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

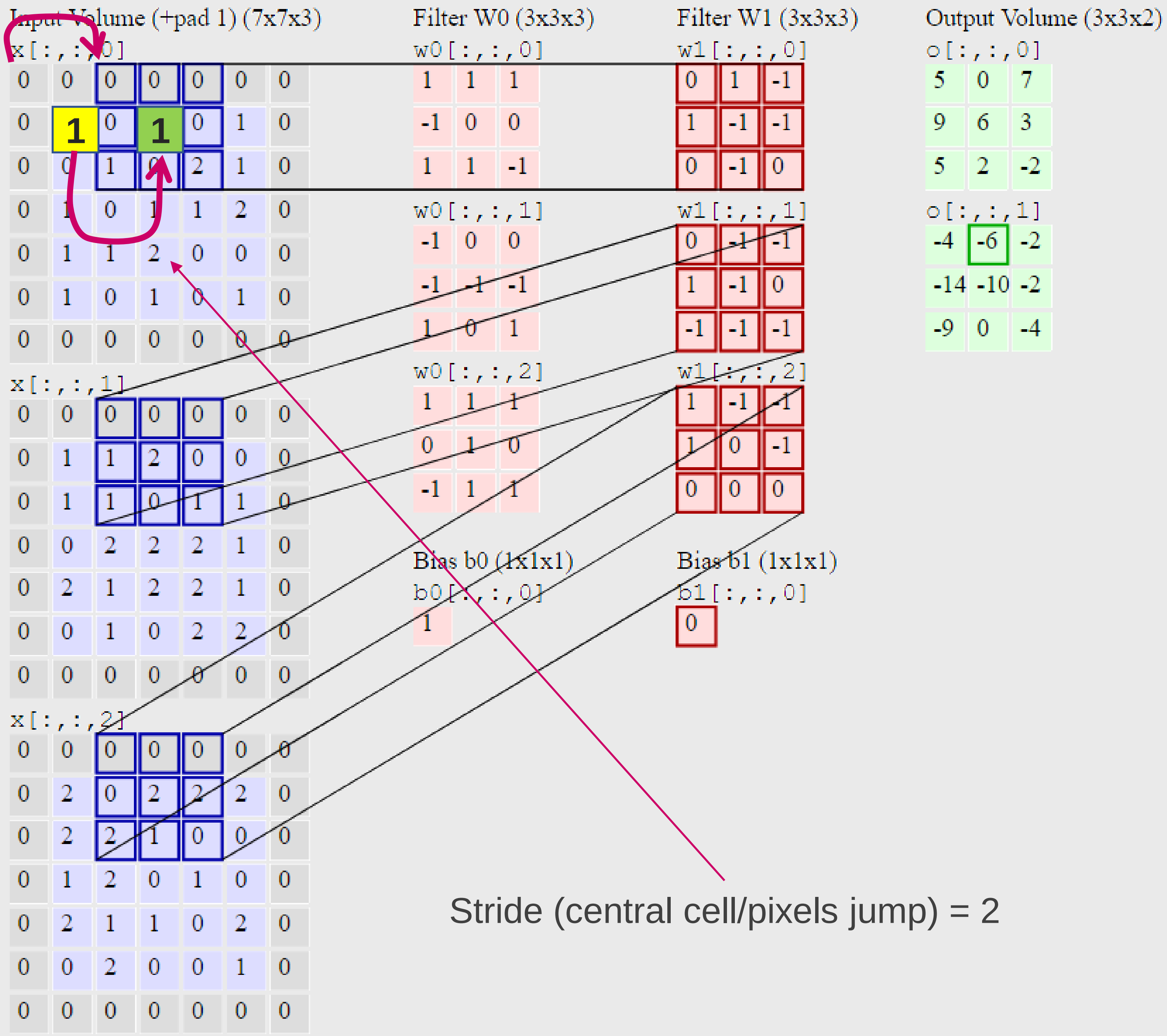
ConvNet
Convolution Layer

Input volume of size 5×5×3
Requires four hyperparameters:

- Number of filters **K = 2**,
- their spatial extent **F = 3**,
- the stride **S = 2**, (moved 2 pixels)
- the amount of zero padding **P = 1**.

Output volume of size $W_2 \times H_2 \times D_2$
where:

- W_2=3
- H_2 =3
- (i.e. width and height are computed equally by symmetry)
- D_2 = 2

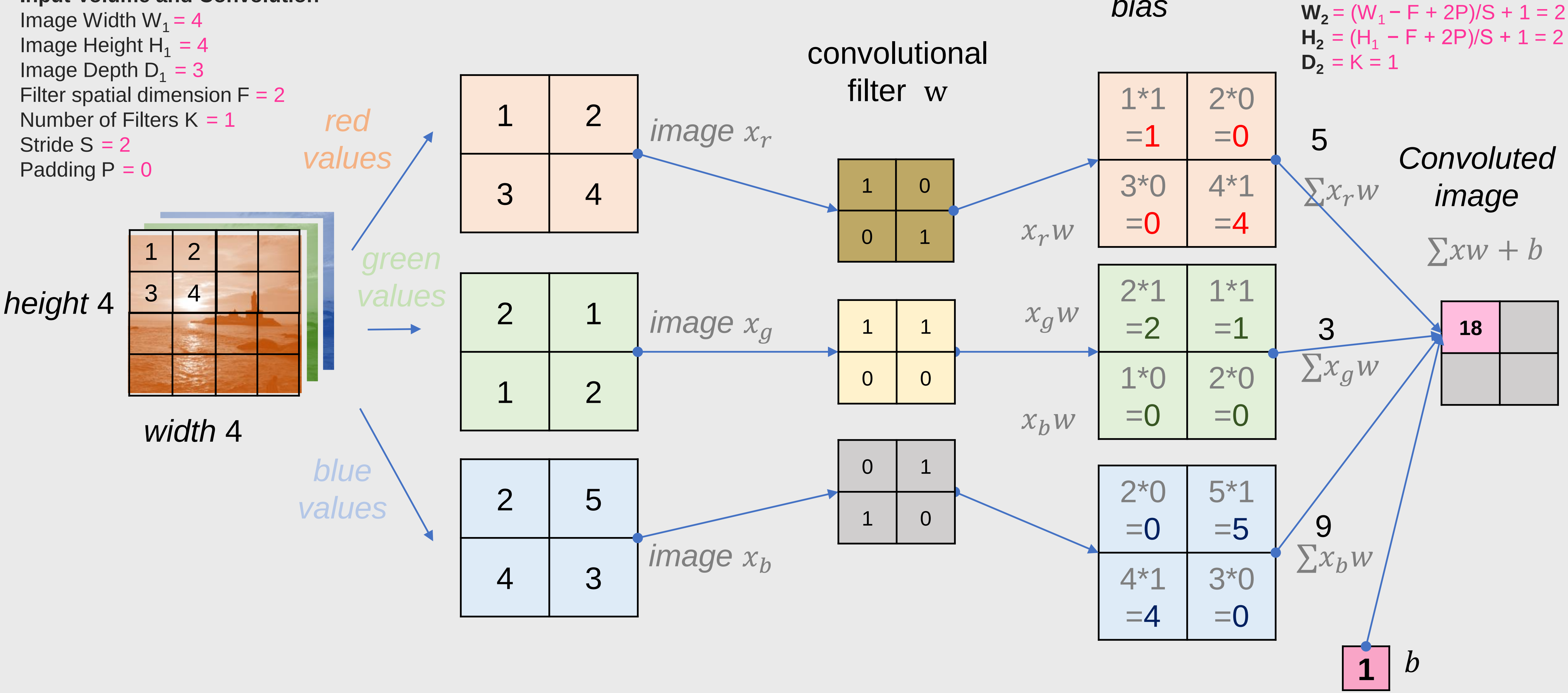


Source: http://cs231n.github.io/convolutional-networks/

Simple Convolution Example

Input Volume and Convolution
Image Width W_1 = 4
Image Height H_1 = 4
Image Depth D_1 = 3
Filter spatial dimension F = 2
Number of Filters K = 1
Stride S = 2
Padding P = 0

Output Volume
$W_2 = (W_1 - F + 2P)/S + 1 = 2$
$H_2 = (H_1 - F + 2P)/S + 1 = 2$
$D_2 = K = 1$



Note that each 2D map performs element-wise multiplication instead a dot product as may be advised in most text. **However**, if you flatten the tensors (i.e., make them vectors), you can do a dot product. Note that is only 1 filter but we may have many filters.

Effect of Learned Convolutional Filter on Images

This is most likely scenario as we do not know exactly what will be the filter weights after training. Hence CNN is a Blackbox

These weights/values of 3x3 kernels/filters are learned during training
(filter values and outputs are mere representative)

Identity

0	0	0
0	**1**	0
0	0	0

Sharpen

0	**-1**	0
-1	**7**	**-1**
0	**-1**	0

Blur

1/5	1/5	1/5
1/5	1/5	1/5
1/5	1/5	1/5

Laplacian

0	**1**	0
1	**-5**	**1**
0	**1**	0

Gaussian

1/8	2/8	1/8
2/8	4/8	2/8
1/8	2/8	2/8

Original

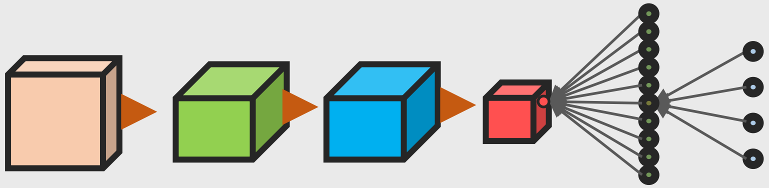
Identity

Sharpen

Blur

Laplacian

Gaussian

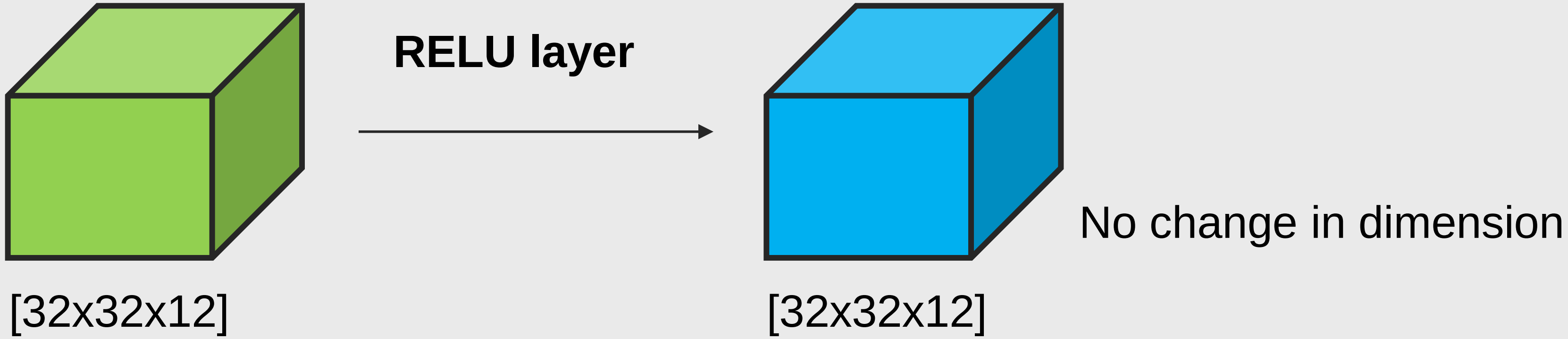


ConvNet/ CNN

Architecture: A Simple ConvNet / CNN

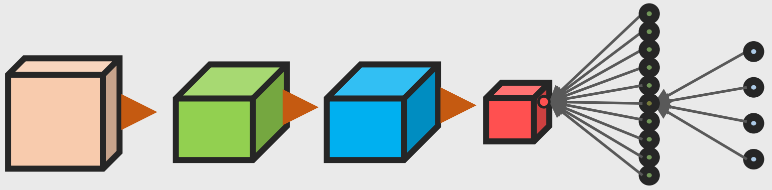
[INPUT - CONV - **RELU** - POOL - FC]

RELU layer will apply an elementwise activation function, such as the $\boldsymbol{max(0,x)}$ thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).



A very good source: http://cs231n.github.io/convolutional-networks/

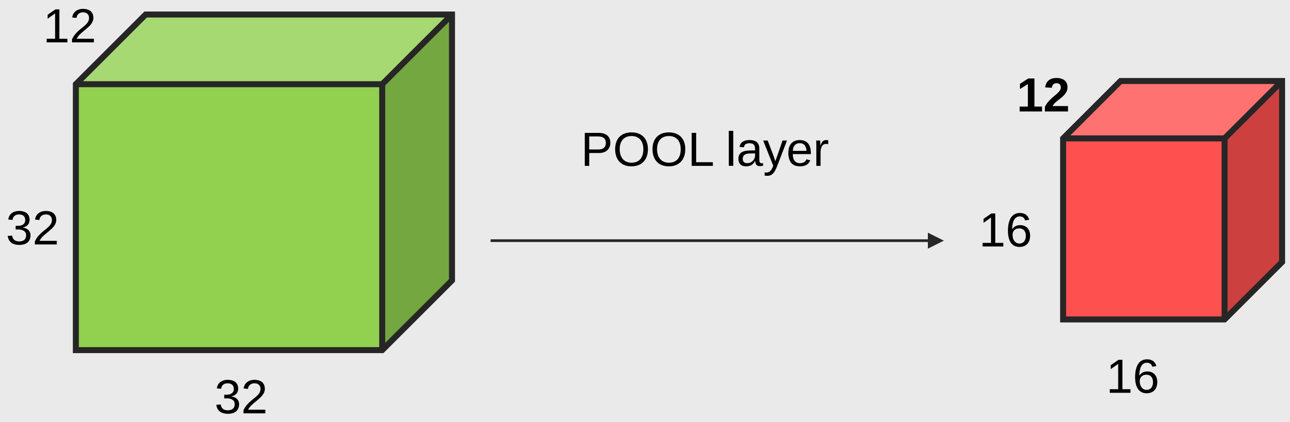
ConvNet/ CNN



Architecture: A Simple ConvNet / CNN

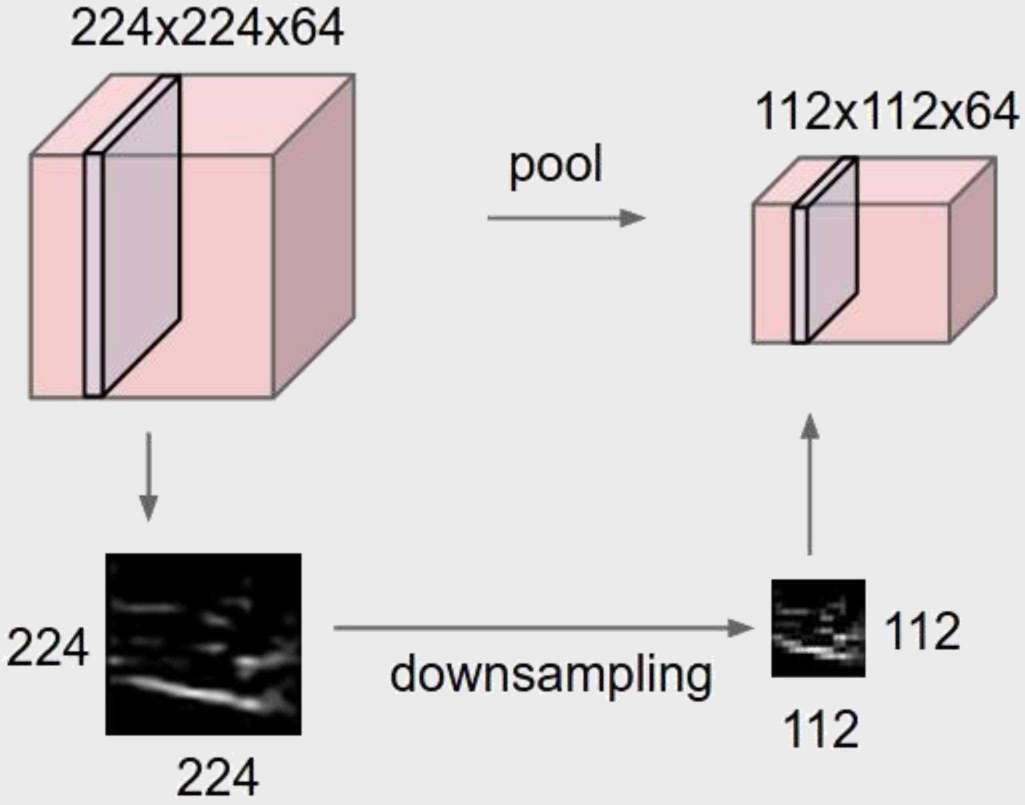
[INPUT - CONV - RELU - POOL - FC]

POOL layer will perform a **down sampling** operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12]



A very good source: http://cs231n.github.io/convolutional-networks/

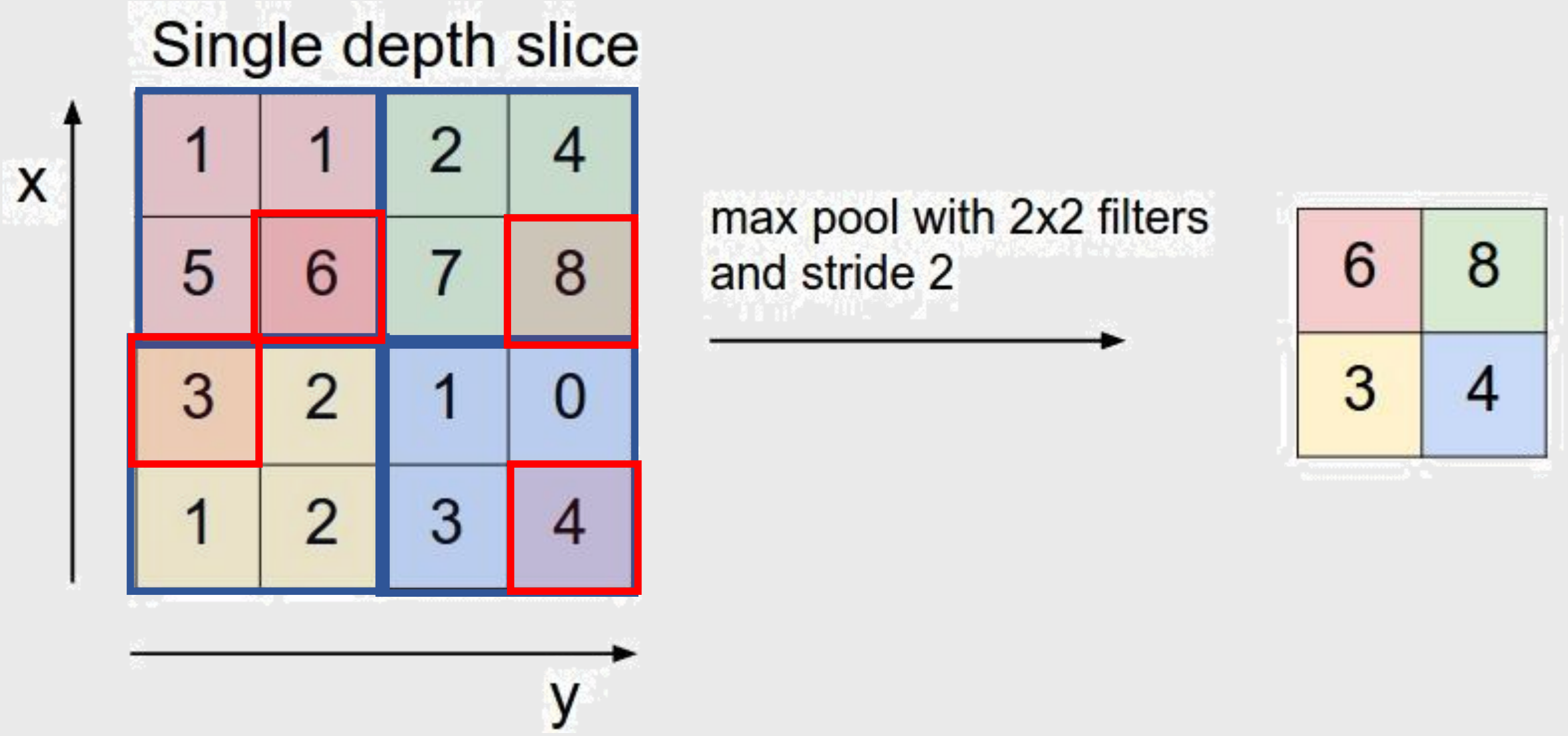
ConvNet
Pooling Layer



Pooling layer **down samples** the volume spatially, independently in each depth slice of the input volume

A very good source: http://cs231n.github.io/convolutional-networks/

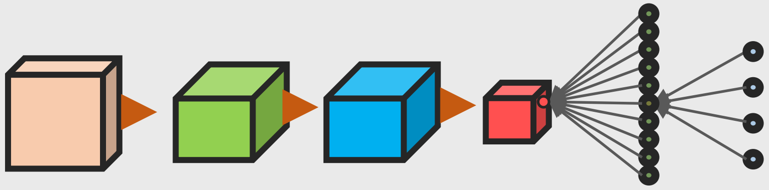
ConvNet
Pooling Layer



Max Pooling layer

A very good source: http://cs231n.github.io/convolutional-networks/

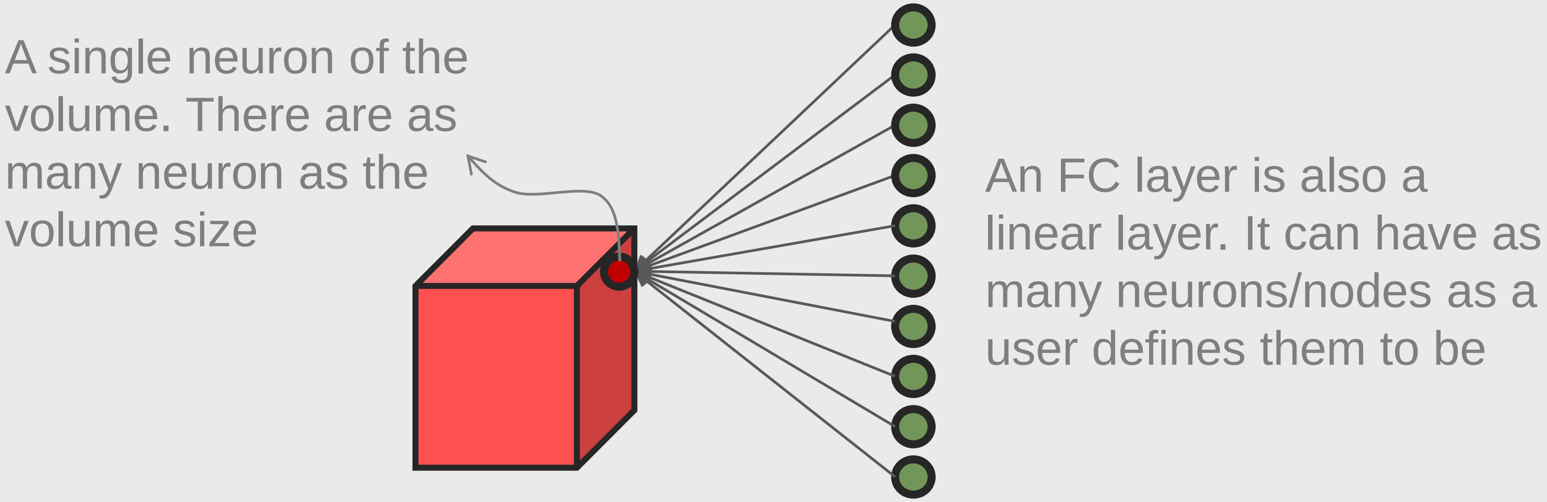
ConvNet/ CNN



Architecture: A Simple ConvNet / CNN

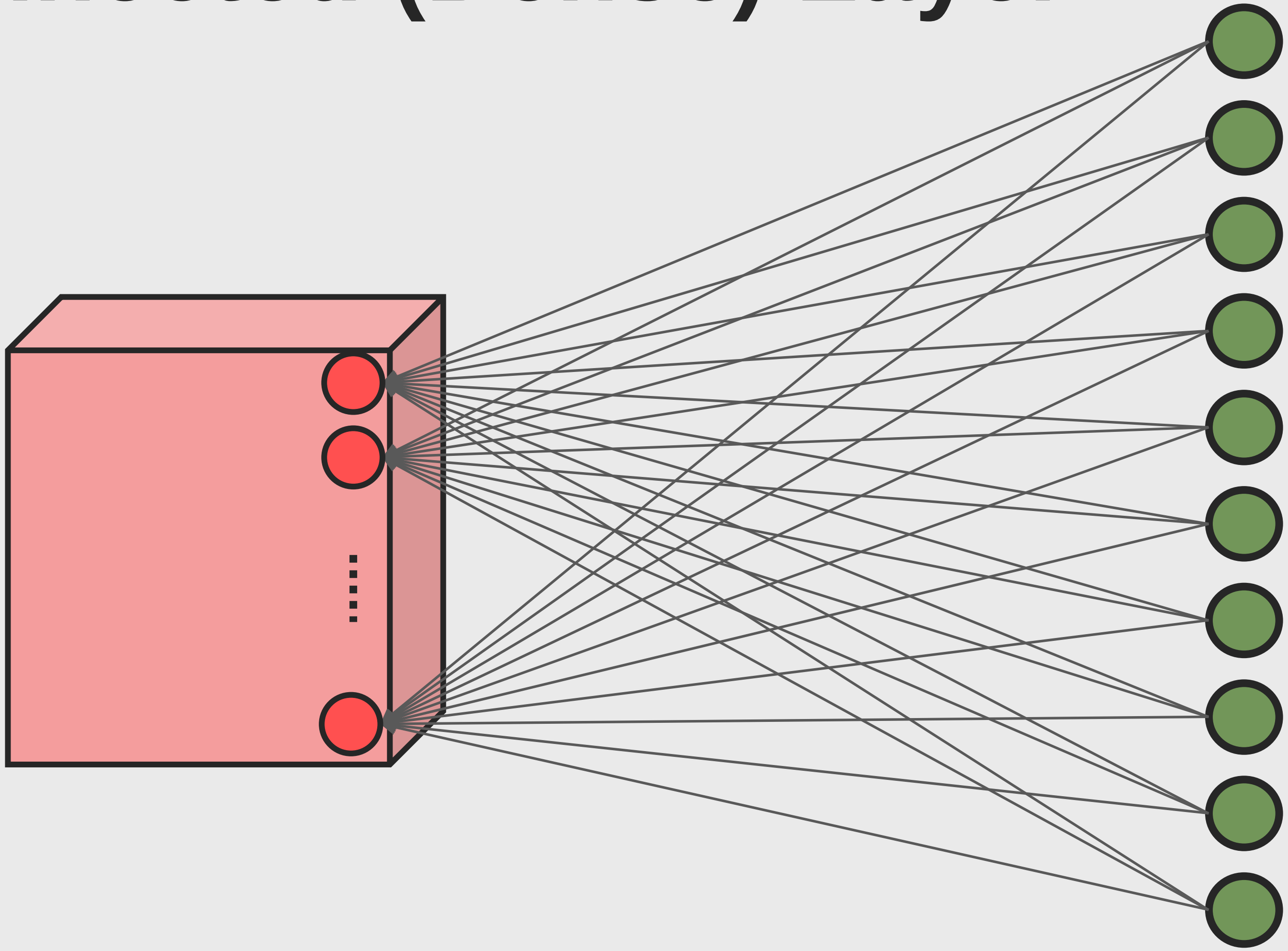
[INPUT - CONV - RELU - POOL - **FC**]

FC (i.e. fully-connected) layer may compute the class scores, resulting in volume of size [1x1x10], where each of the 10 neurone correspond to a class score, such as among the 10 categories.



ConvNet

Fully Connected (Dense) Layer



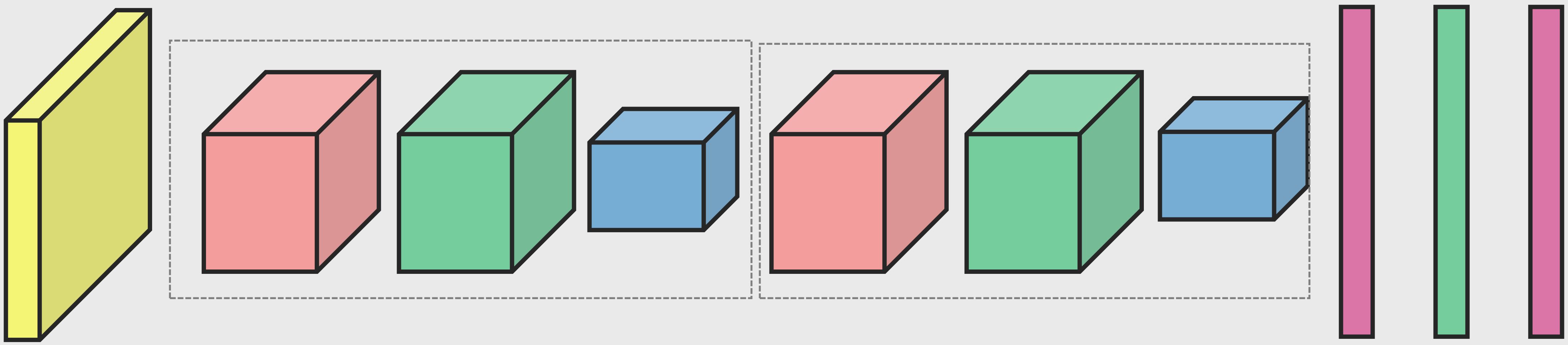
Classes (e.g. 10 for MNIST dataset)

This typically outputs SoftMax

ConvNet

ConvNet Architecture

INPUT $\longrightarrow$ [CONV $\longrightarrow$ RELU $->$ POOL] $*$ 2 $\longrightarrow$ FC $\longrightarrow$ RELU $\longrightarrow$ FC]



A very good source: http://cs231n.github.io/convolutional-networks/

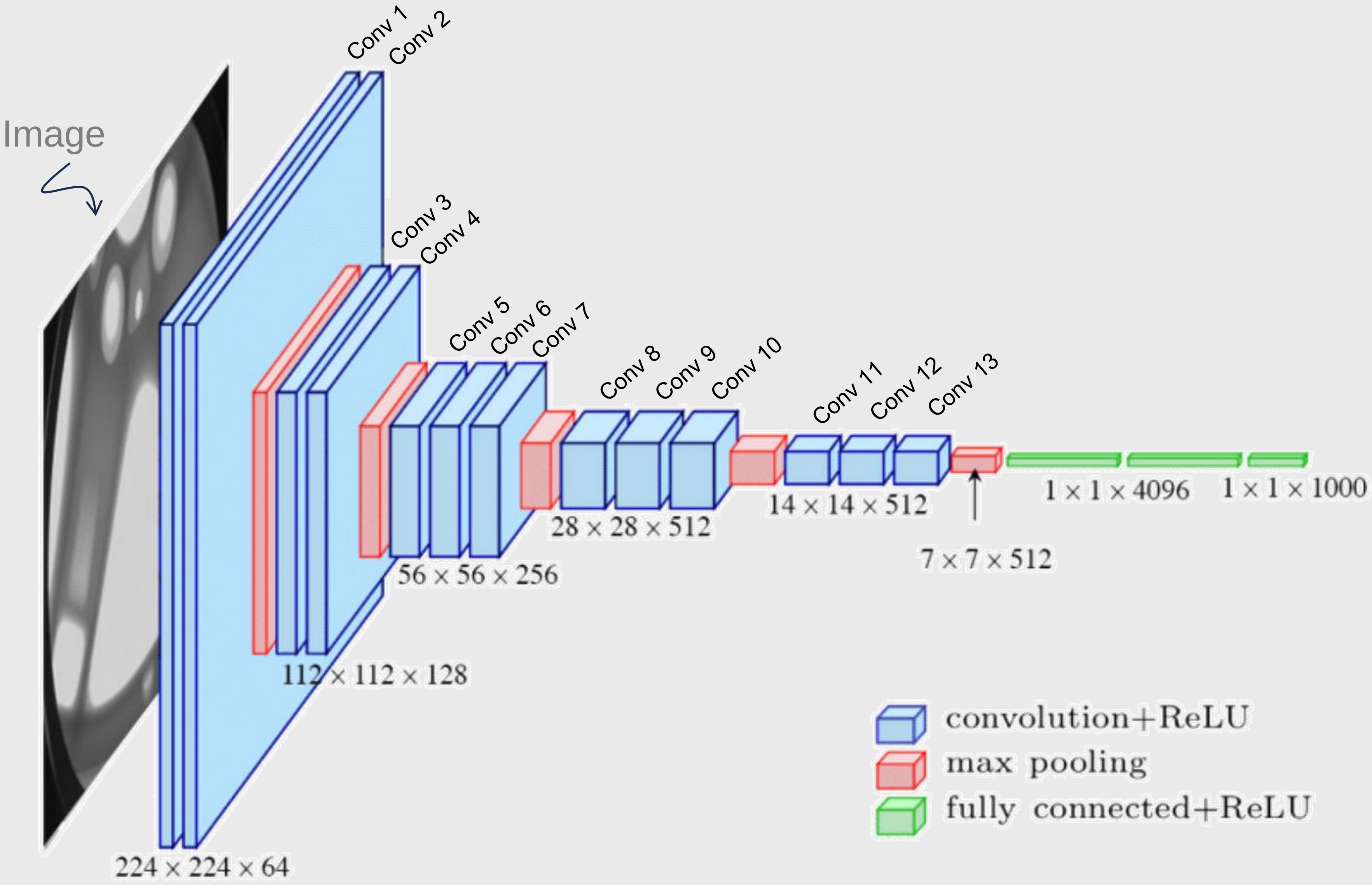
ConvNet: Image Classification Example

Live demo http://cs231n.stanford.edu/

A very good source:
http://cs231n.github.io/convolutional-networks/

VGG Net

- Visual Geometry Group (VGG) Network (VGG Net)
- VGG 16 - an example architecture: 13 Convolution layers 3 fc layer.



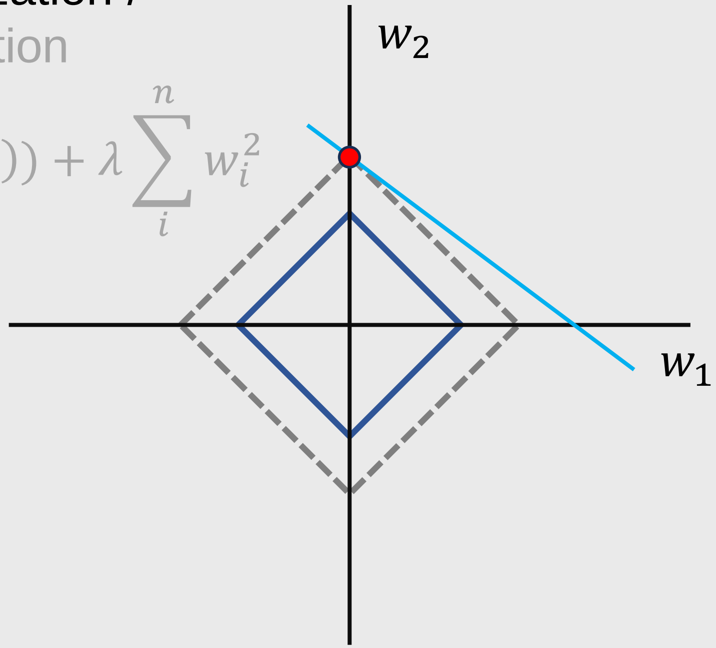
Simonyan et al. (2014) Very deep convolutional networks for large-scale image recognition. *ICLR 2015*

Regularization: Avoid Overfitting

L_1-norm regularization /

Lasso regularization

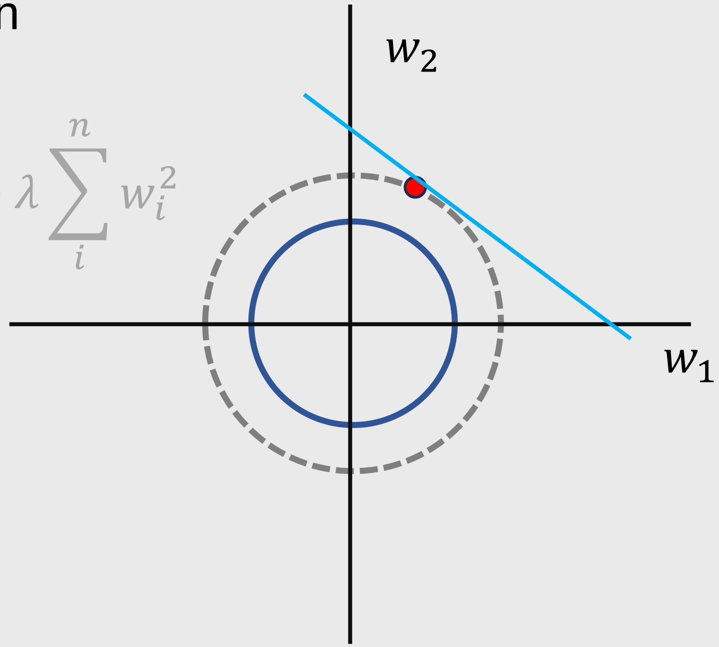
$$E = \frac{1}{n}\sum_{i}^{n}(y - f(\boldsymbol{x})) + \lambda \sum_{i}^{n} w_i^2$$



L_2-norm regularization

Ridge regularization

$$E = \frac{1}{n}\sum_{i}^{n}(y - f(\boldsymbol{x})) + \lambda \sum_{i}^{n} w_i^2$$



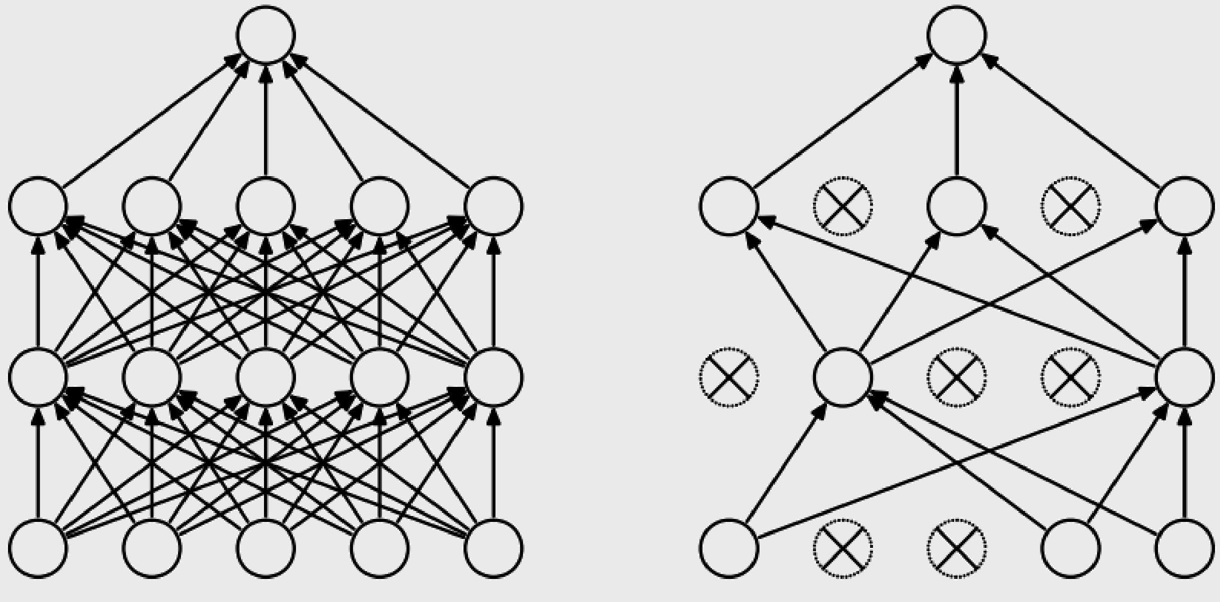
- Both Lasso and Ridge regularization help DNNs and CNNs avoid overfitting.
- As weights can get zero in Lasso regularization introduce sparsity in the networks and help feature selection because some weights goes to zero. Thus, eliminating effect of some input features, while in ridge regression weights can only get close to zeros and not exactly (see images).
- L2 penalizes large errors much more heavily than small errors (compared to L1), thus on optimization of network, it is safe to assume that all the errors are roughly of the same order of magnitude

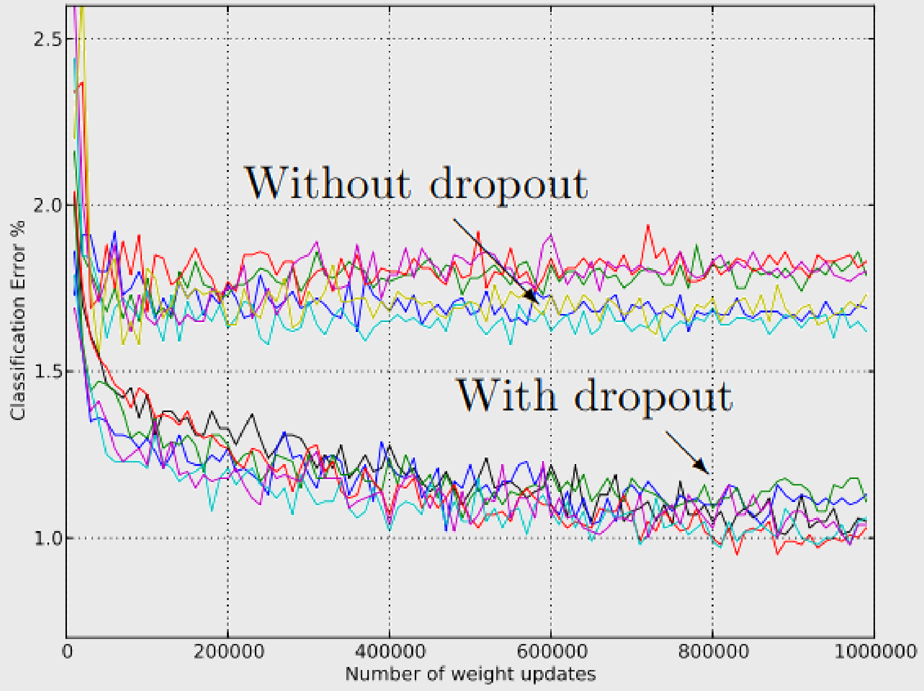
Regularization: Dropouts Layer

Regularisation of DNNs and CNNs

!!!
Slows down training in Convolutional Nets

- It drop nodes with some probability
- It regularise Deep Neural Nets and Convolutional Neural Nets


(a) Standard Neural Net (b) After applying dropout.



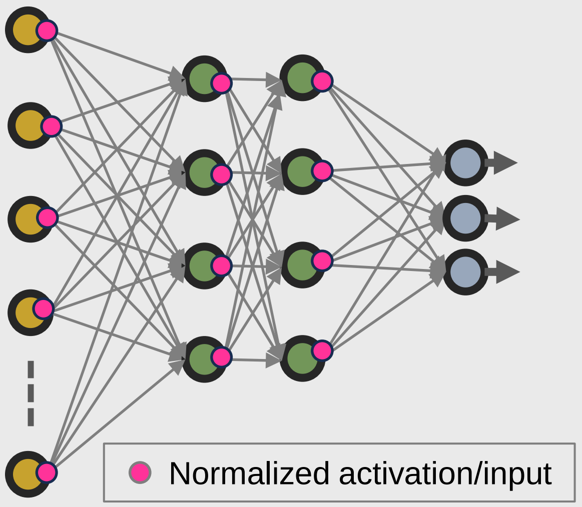
Sec 7.12, Goodfellow, Deep Learning, MIT Press

Srivastava et al (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting, JMLR

Batch Normalization Layer

Regularisation of Convolutional Neural Networks

- It eliminates the need of dropout
- Accelerates ConvNet training
- It reduces sensitivity to network weight initialisation

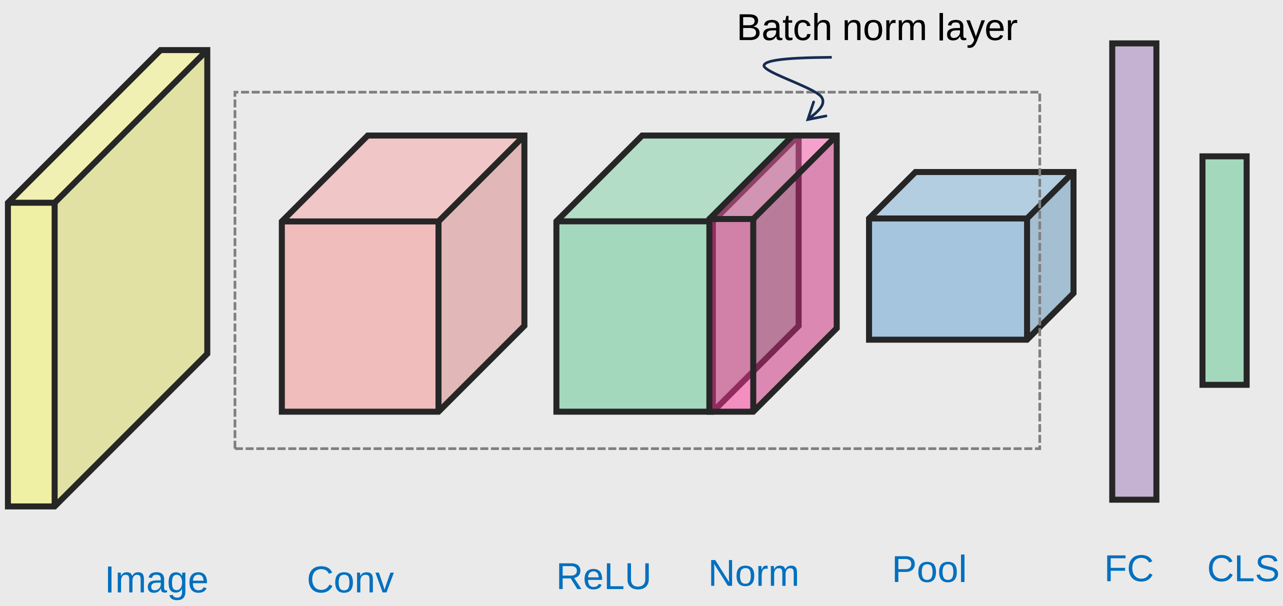


$$\mu_{\mathcal{B}} = \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}}^2 = \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$$

$$\hat{x}_i = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$$

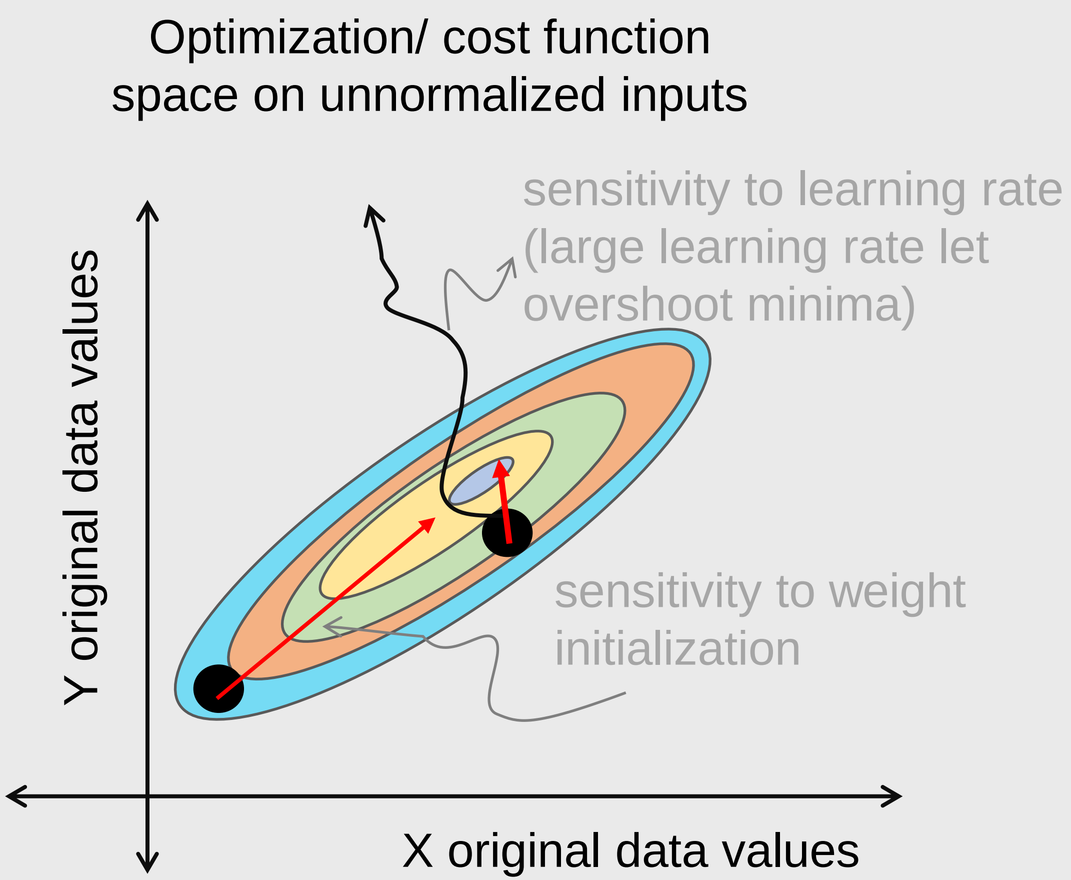
$$y_i = \gamma \hat{x}_i + \beta = \mathrm{BN}_{\gamma,\beta}(x_i)$$

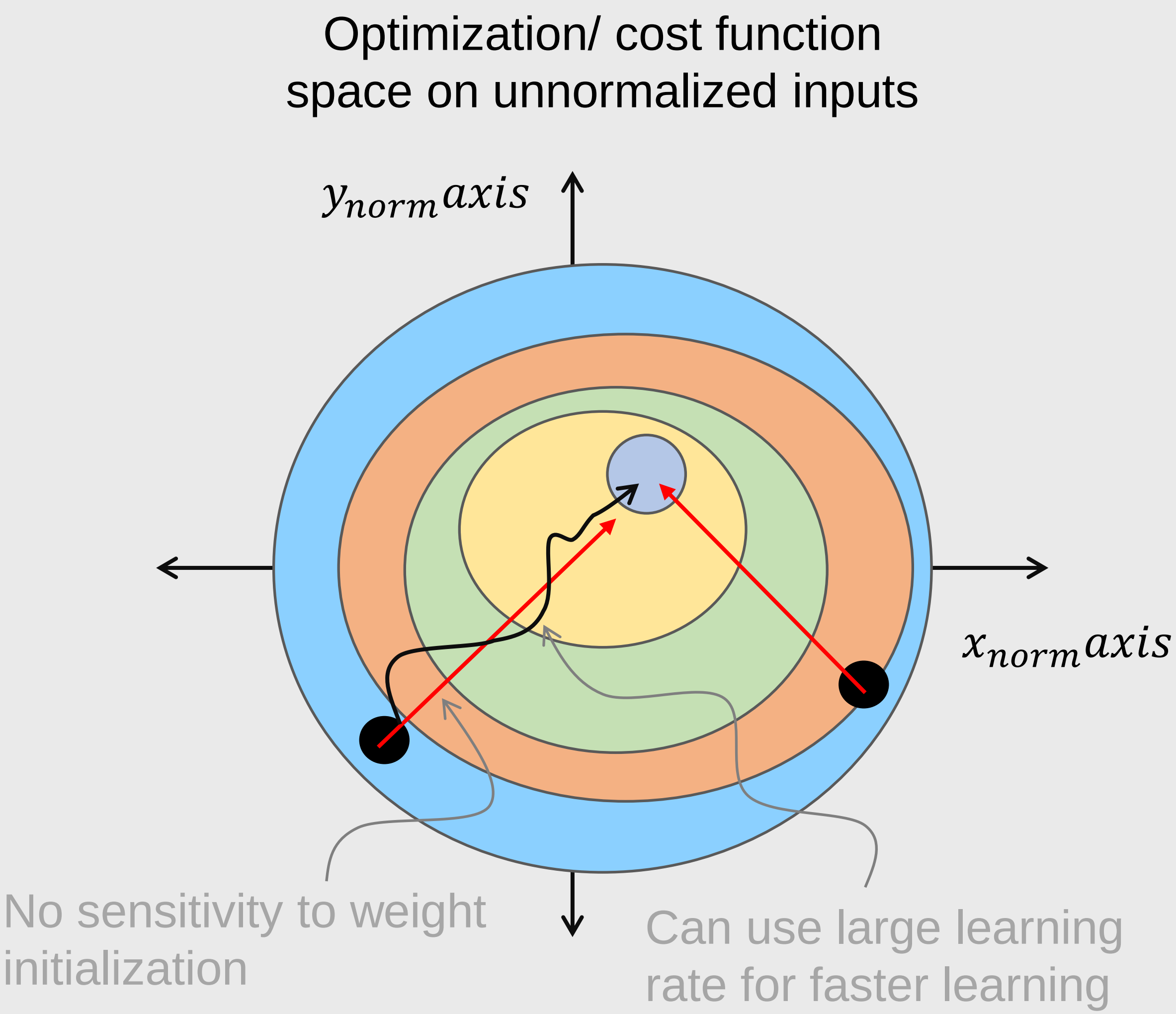


Ioffe and Szegedy (2017) Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift ICML.

Batch Normalization Layer

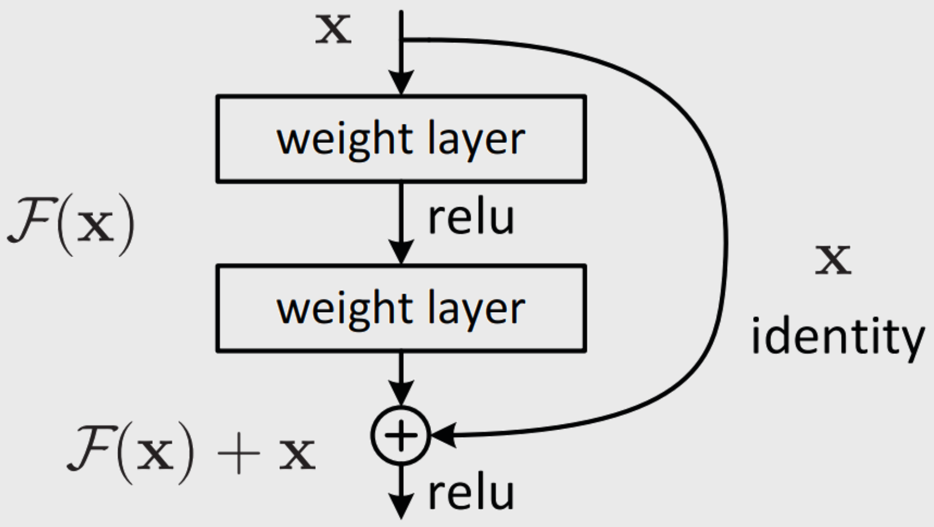
Regularisation of Convolutional Neural Networks



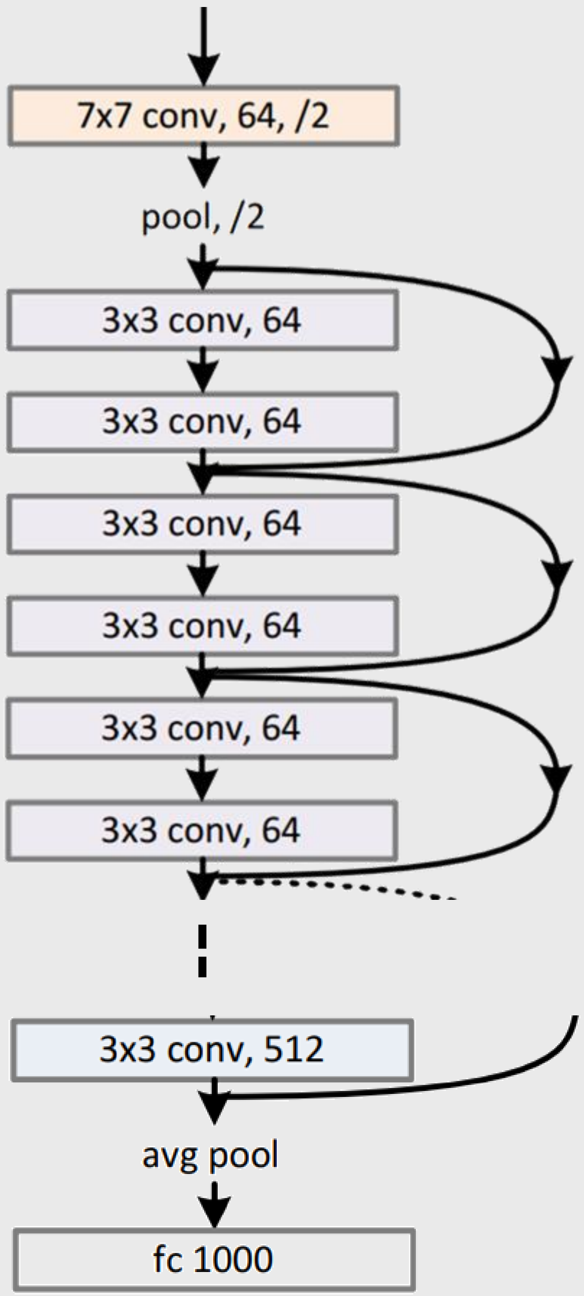


Residual Network (ResNet)

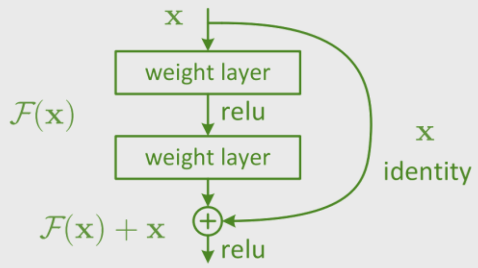
ResNet Block

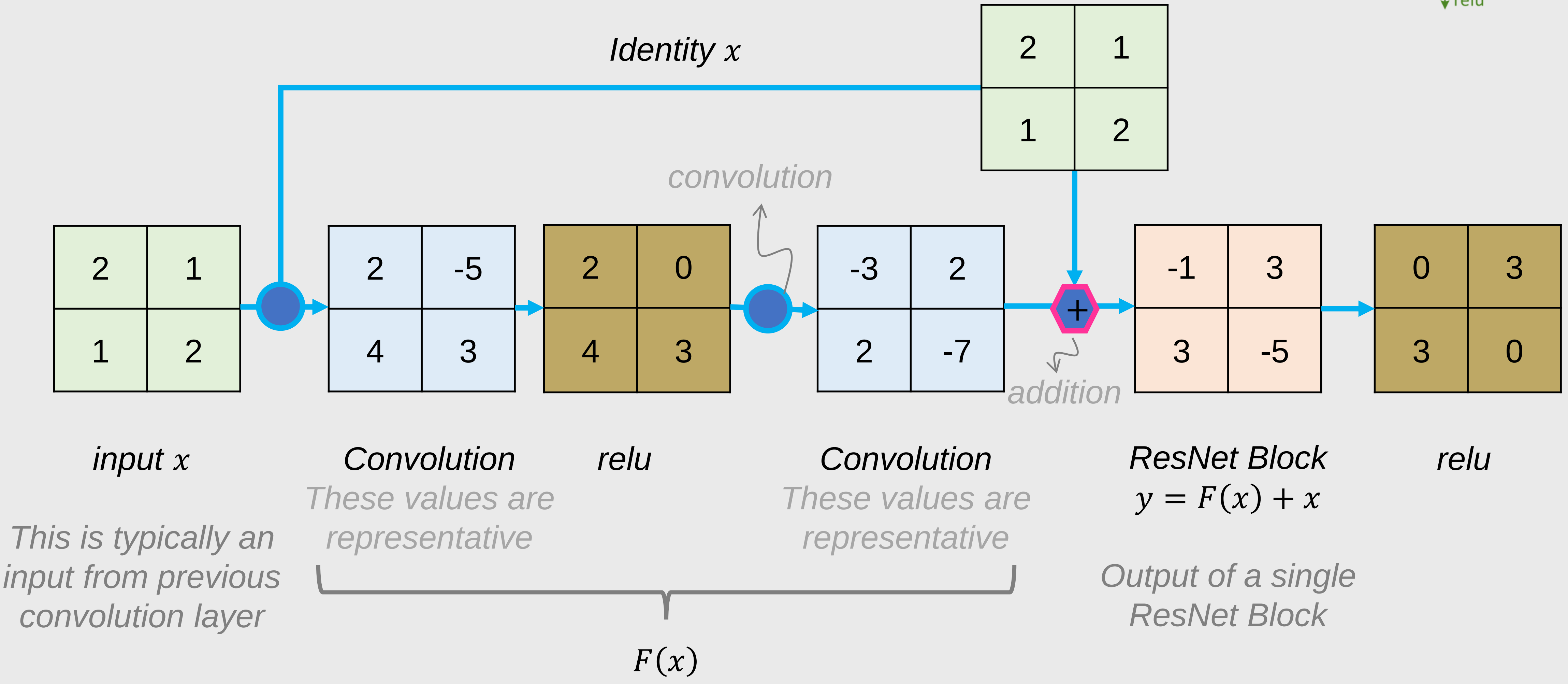


He et al (2016) Deep Residual Learning for Image Recognition, CVPR

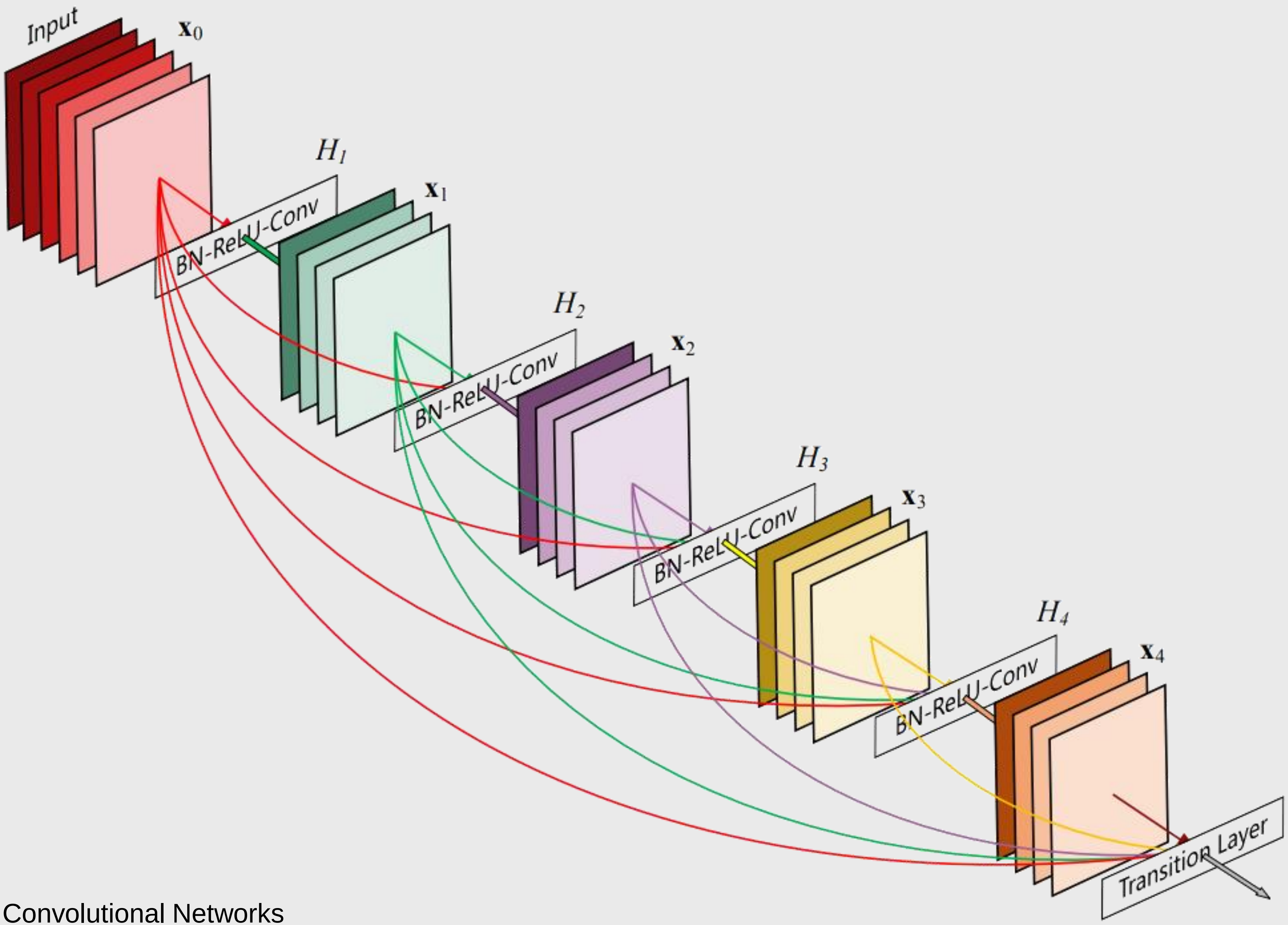


Simple Example of ResNet Block





Dense Net



Huang et al (2017) Densely Connected Convolutional Networks

Part 2
Convolutional Neural Network Applications

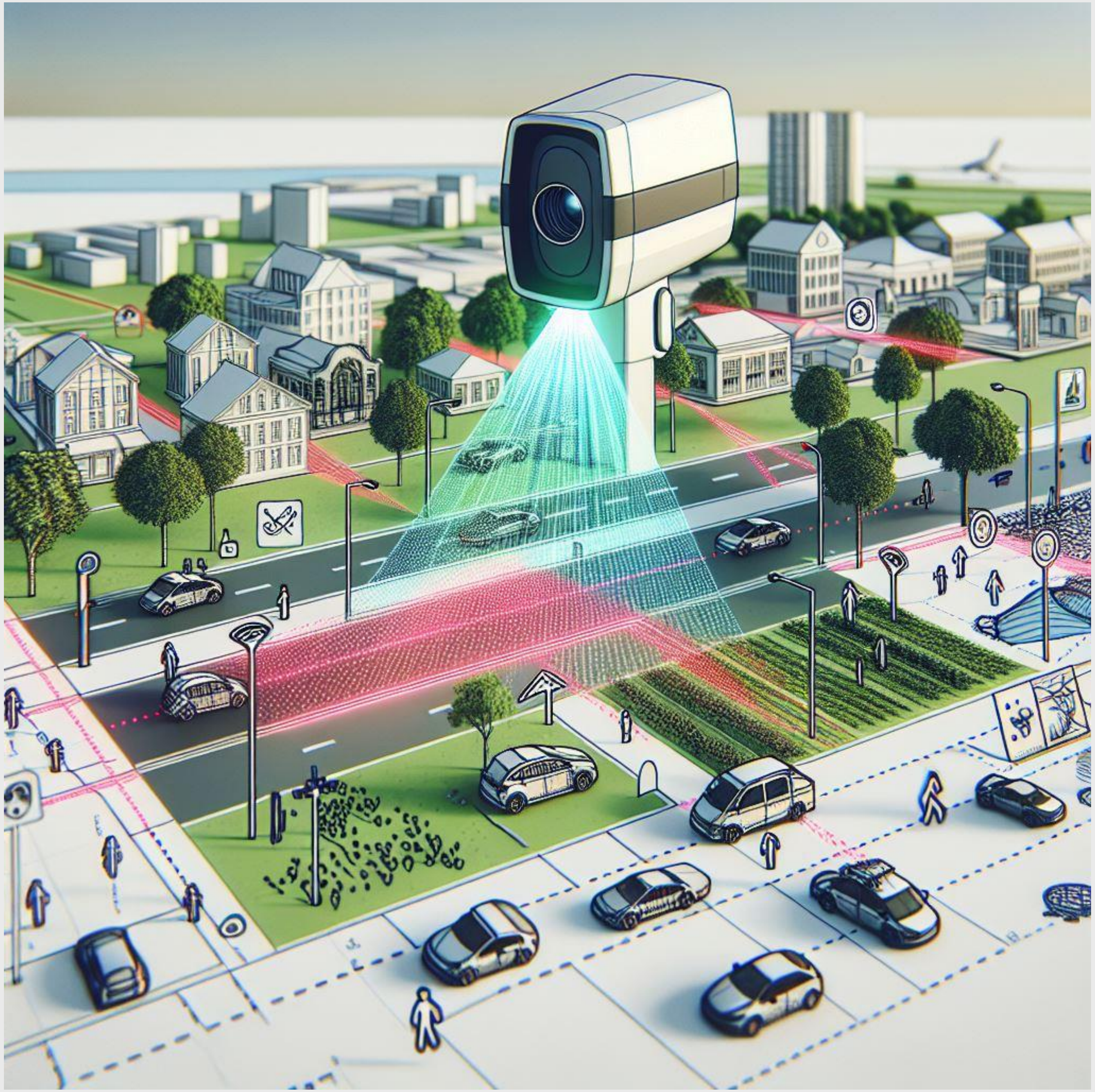


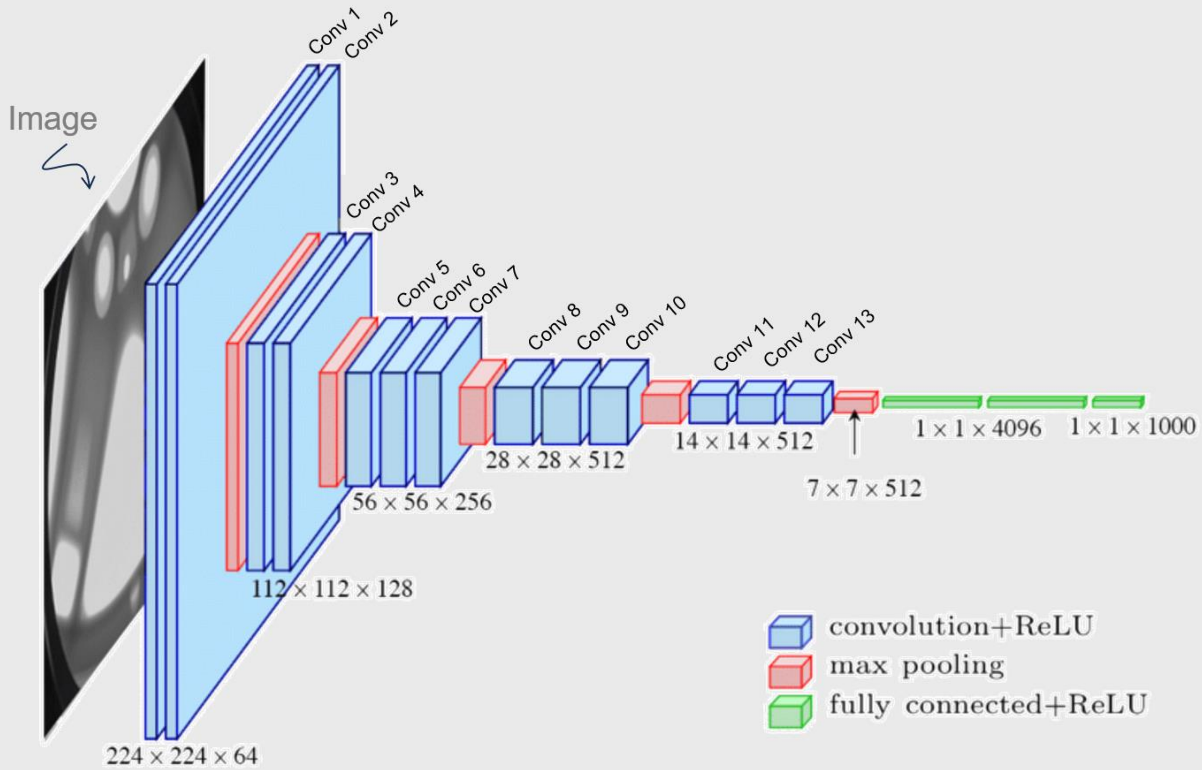
Image Classification Example

Live demo http://cs231n.stanford.edu/

A very good source:
http://cs231n.github.io/convolutional-networks/

Image Classification Models

- Residual Networks (**ResNet**)
- Visual Geometry Group (VGG) Network (**VGG Net**)
- **DenseNet**
- **InceptionNet**
- Other pre-trained Image Classification Nets on pytorch library: https://pytorch.org/vision/main/models.html



VGG Net 16

Image Segmentation (Concept)

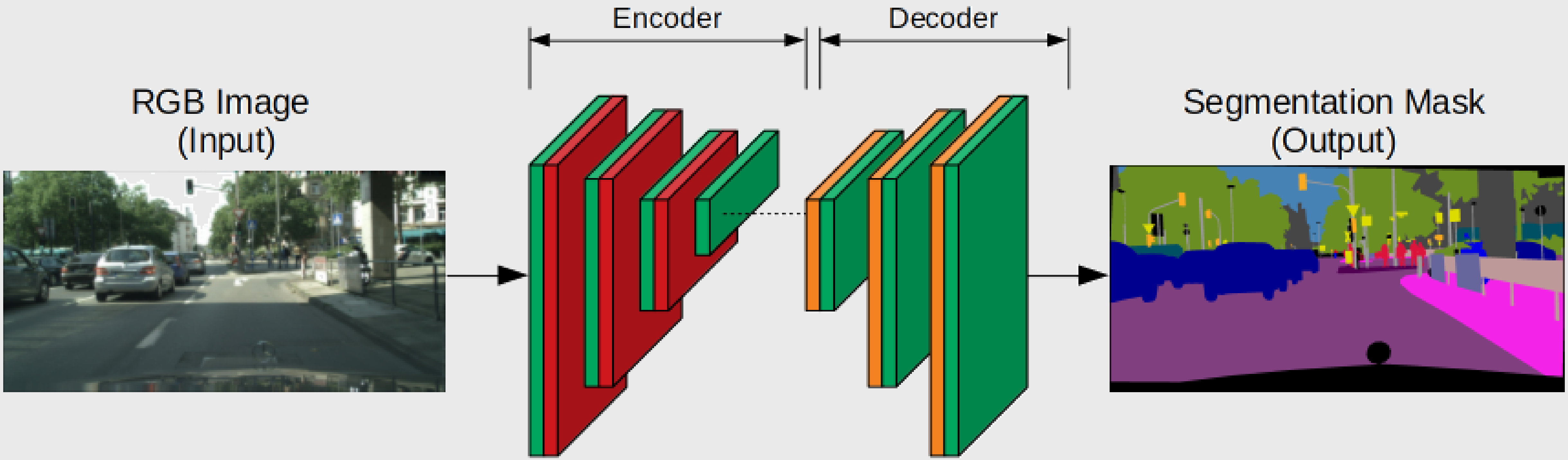
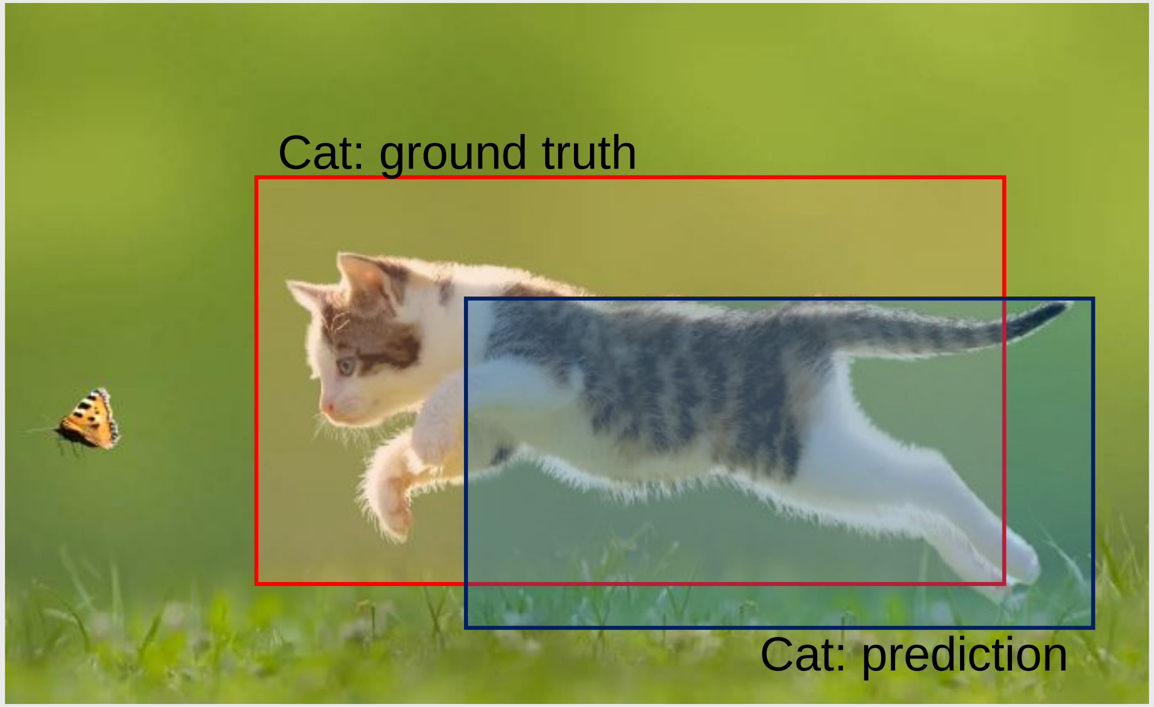
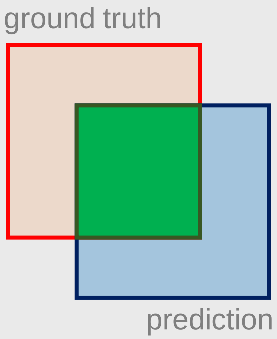


Image source: https://towardsdatascience.com/espnetv2-for-semantic-segmentation-9e80f155d522

Intersection over Union (IoU)

IoU measure the performance of image segmentation and object detection algorithms performance





$$\text{IoU} = \frac{\text{Area of Intersection}}{\text{Area of Union}}$$

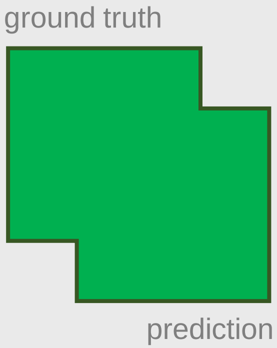
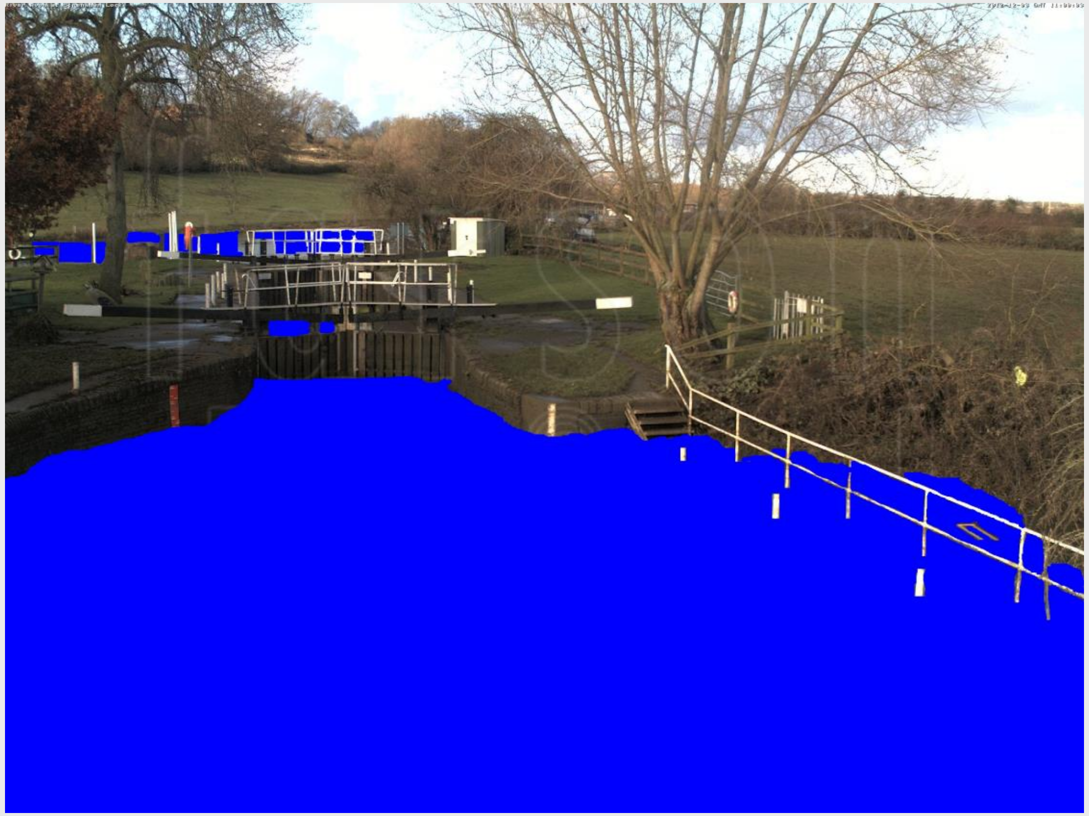


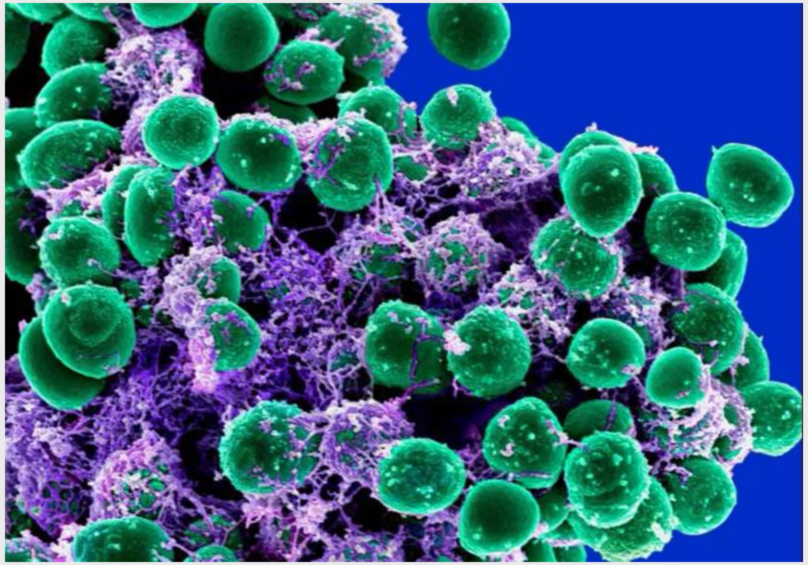
Image Segmentation used for water level estimation



Pixel-wise water segmentation of RGB images for river water-level monitoring or flood monitoring

Vandaele, R., Dance, S. L., & Ojha, V. (2020). Automated water segmentation and river level detection on camera images using transfer learning. GCPR

Image Segmentation (Concept)

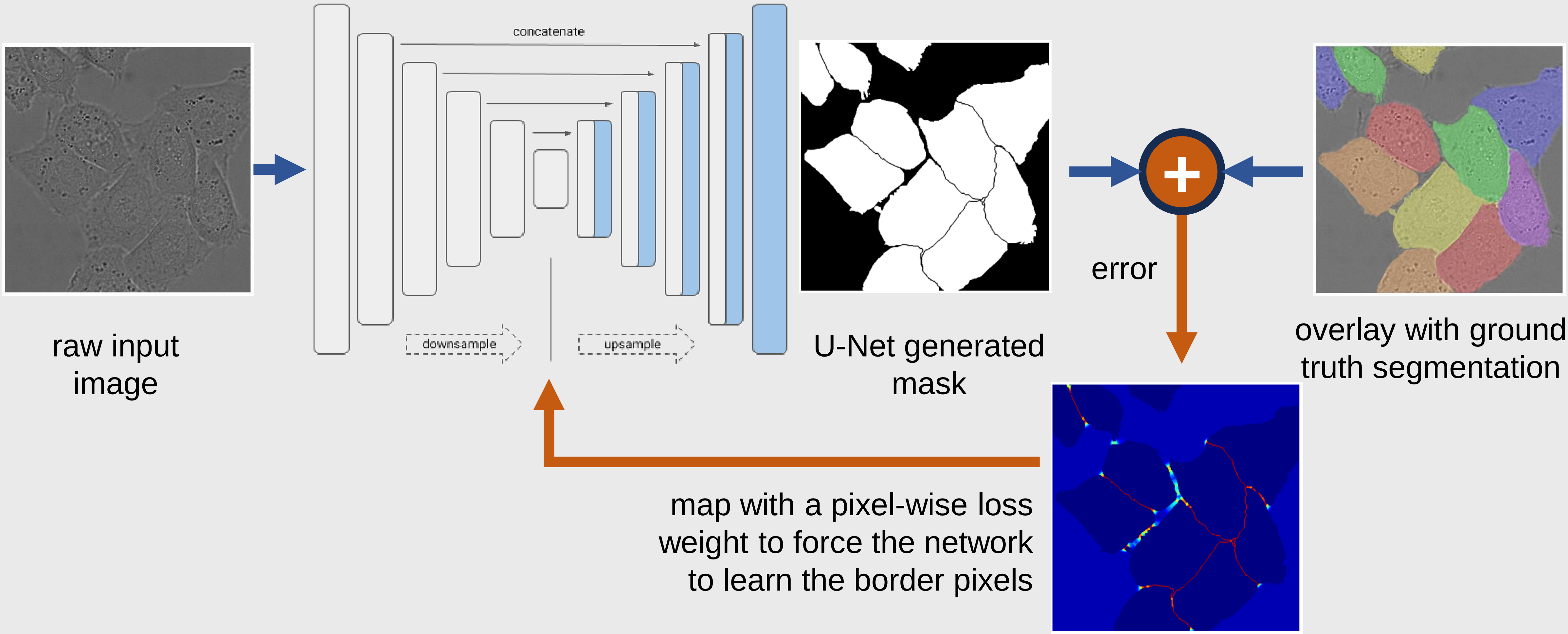


Segment Anything Model from Meta (2023)

https://segment-anything.com/

UNet

U-Net is a typical CNN architecture for semantic segmentation.
It has a contracting path (down sampling) and an expansive path (up sampling).

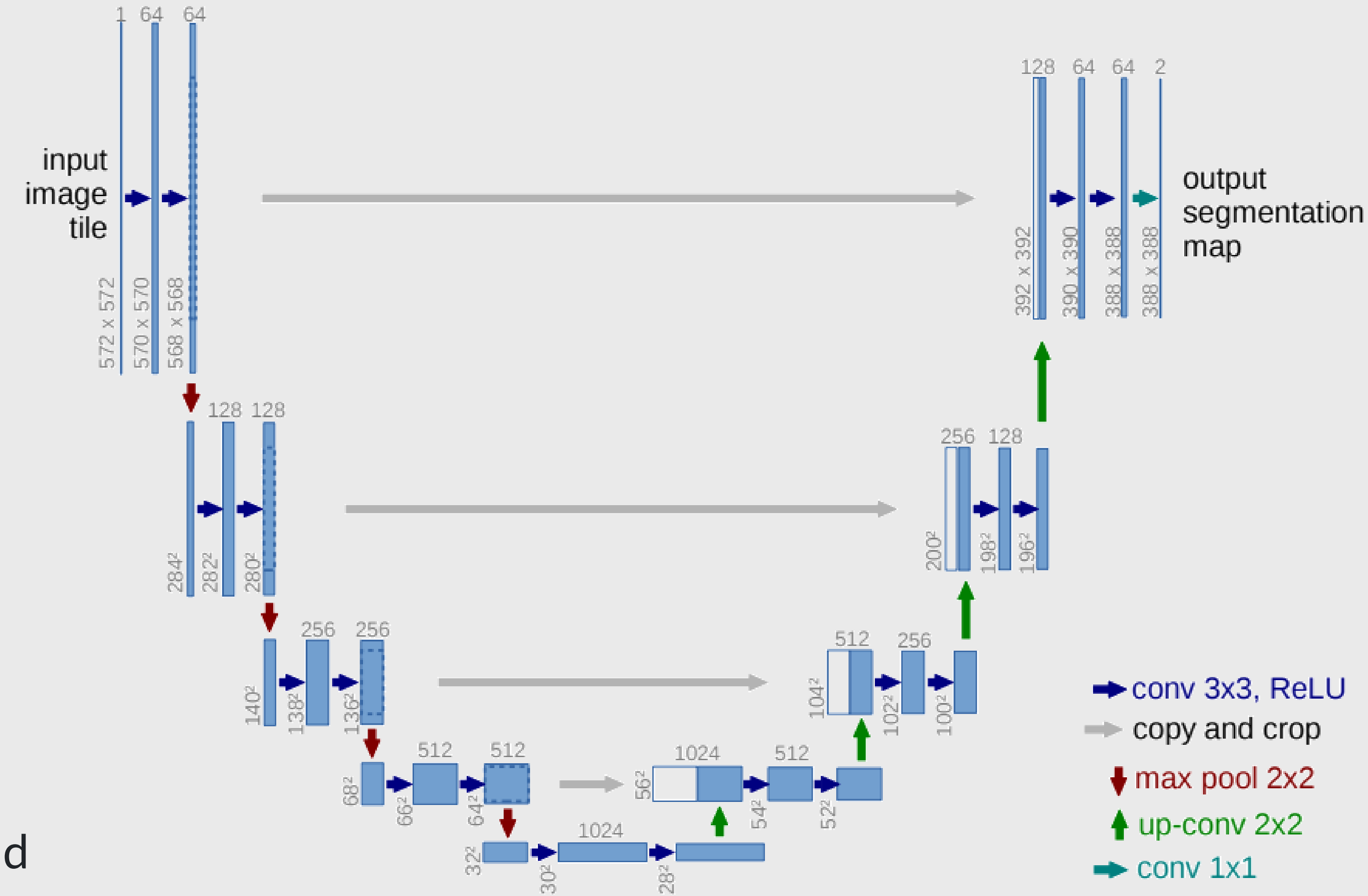


Ronneberger et al (2015). U-net: Convolutional networks for biomedical image segmentation. *MICCAI*

U-Net

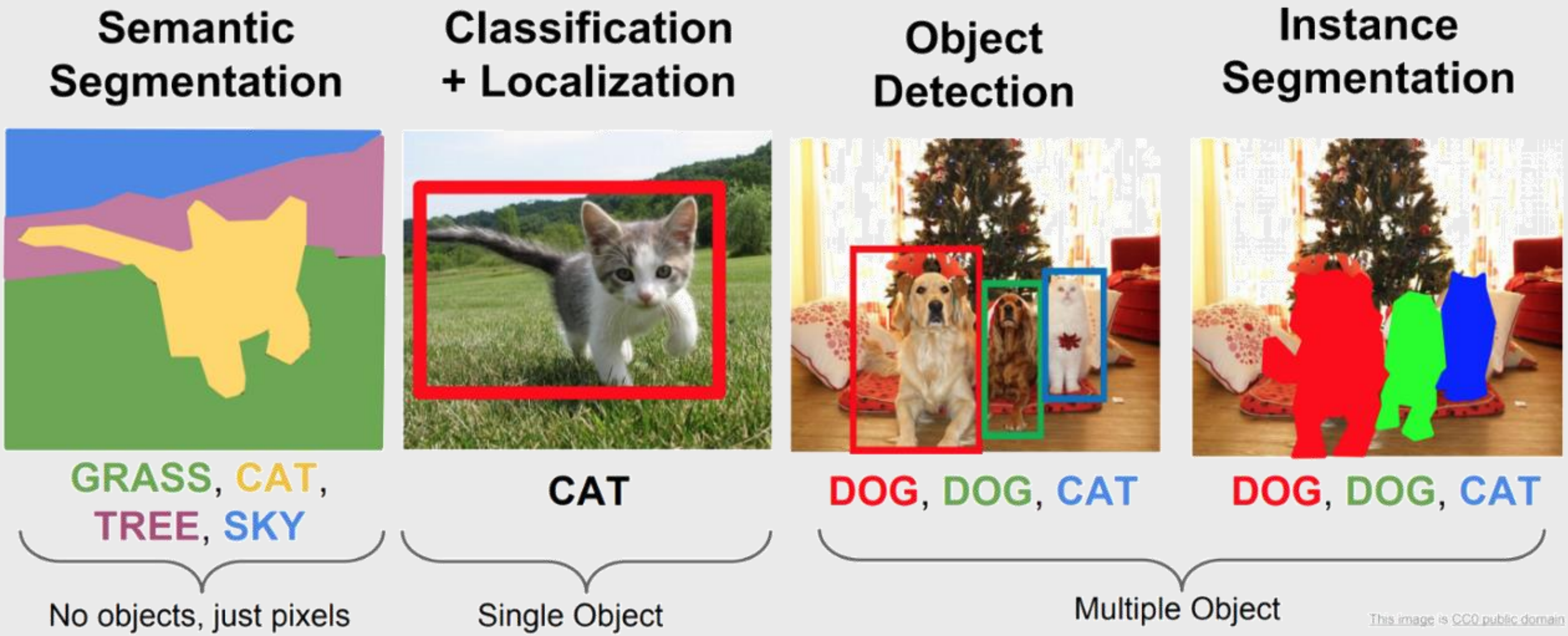
- **Contracting path** is a repeated application of **two 3x3 convolutions** and **a ReLU** and **a 2x2 Max Pooling** operation with stride 2

- **Expansive path** is an up sampling of the feature map followed by **a 2x2 Convolution** ("up-convolution") that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and **two 3x3 convolutions**, each followed by **a ReLU**.



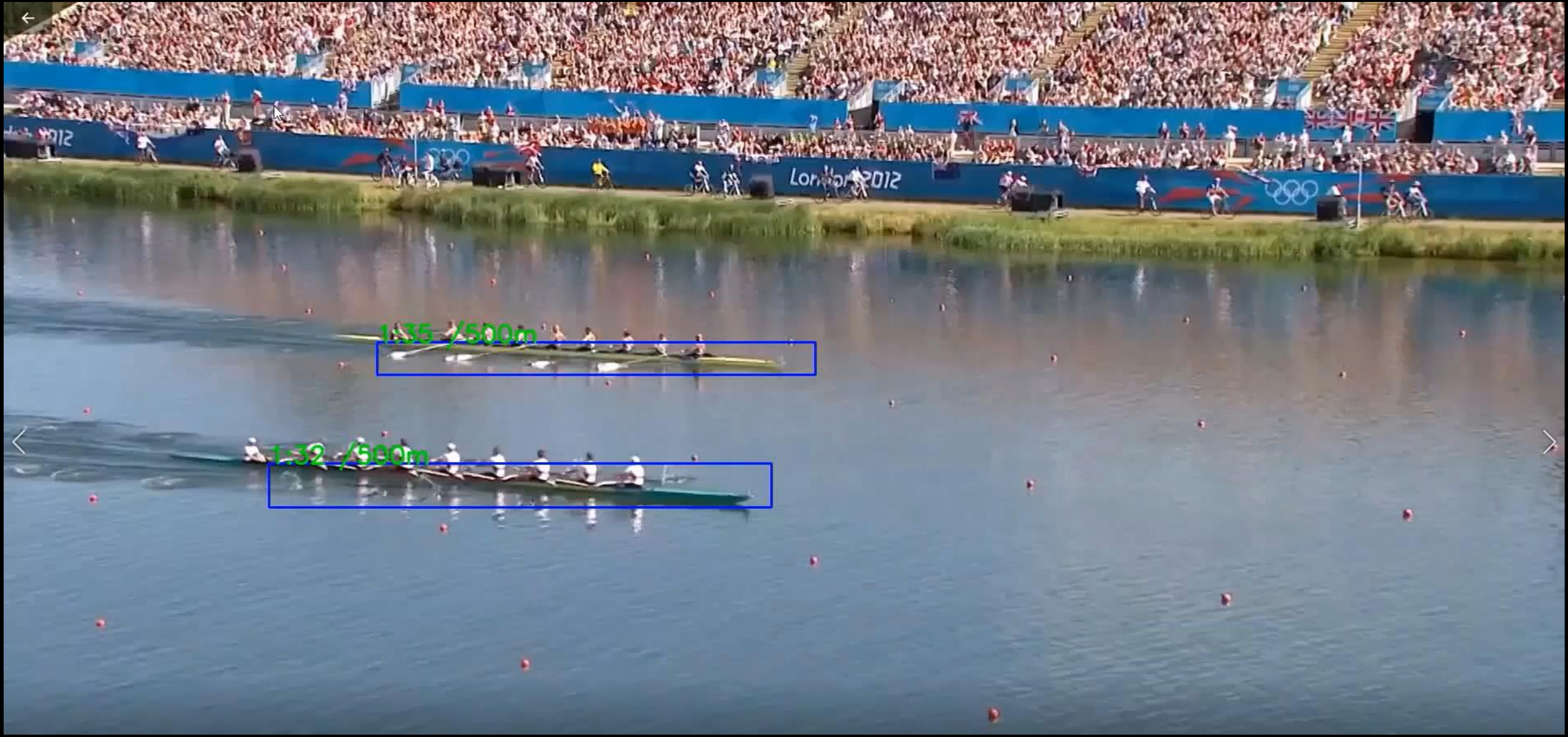
Ronneberger et al (2015). U-net: Convolutional networks for biomedical image segmentation. *MICCAI*

Object Detection

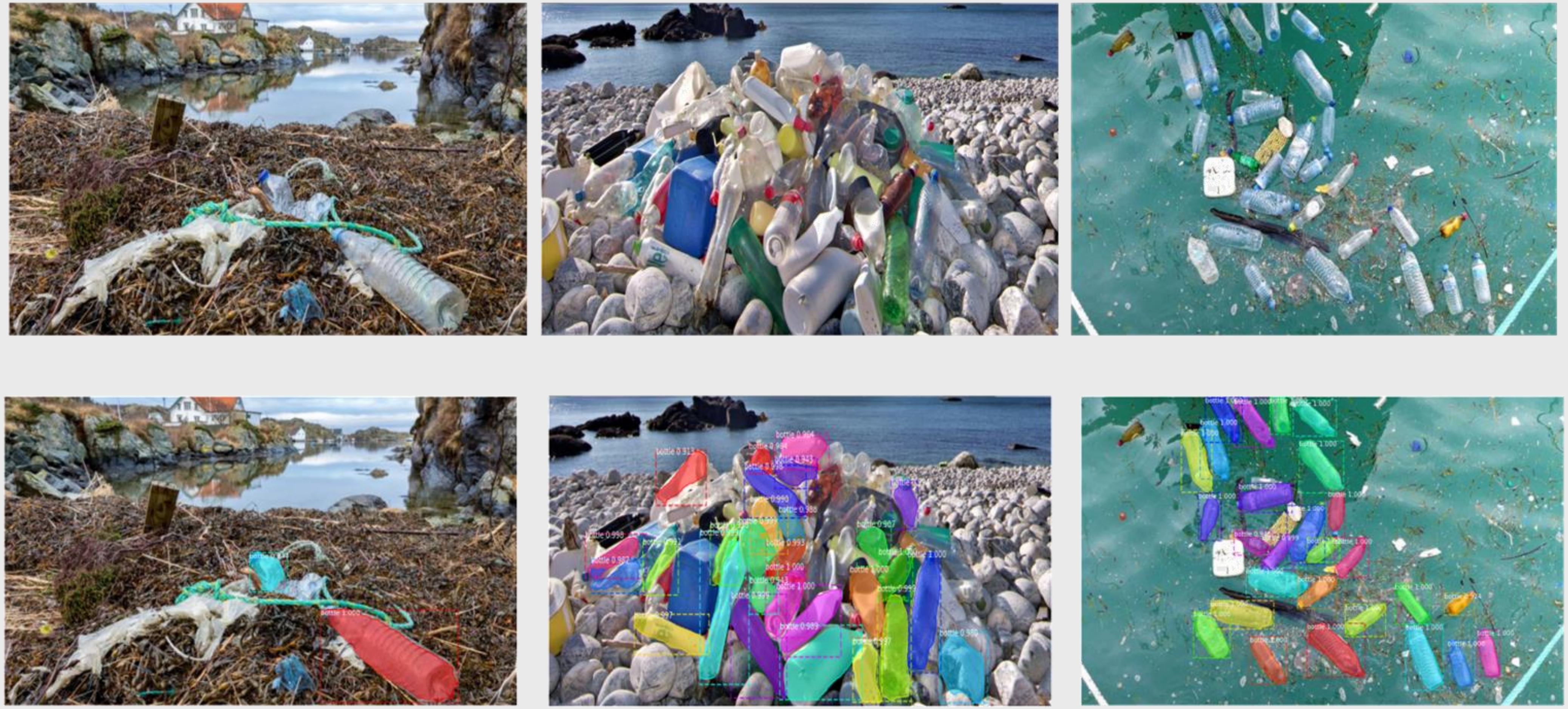


Application: Boat (Object) Detection and boat speed measurement

One of my Undergraduate student’s project



Application: Instance Segmentation and Object Detection: Plastic Pollution Detection



Jaikumar P et al. (2020) ISDA

Object Detection

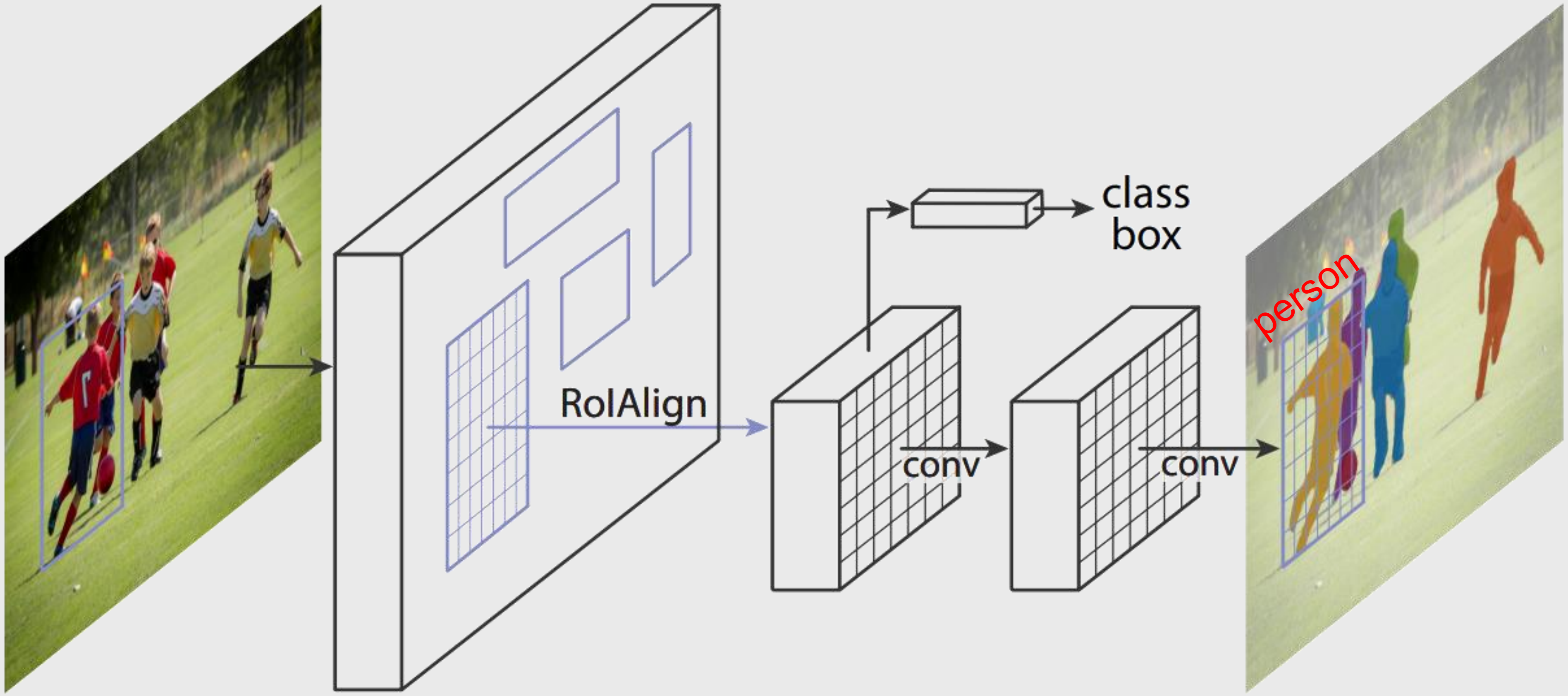
(plastic pollution detection - One of my student’s project)

Input Video

Output Video

Published work: Jaikumar P et al. (2020) ISDA, https://centaur.reading.ac.uk/98569/

Mask RCNN



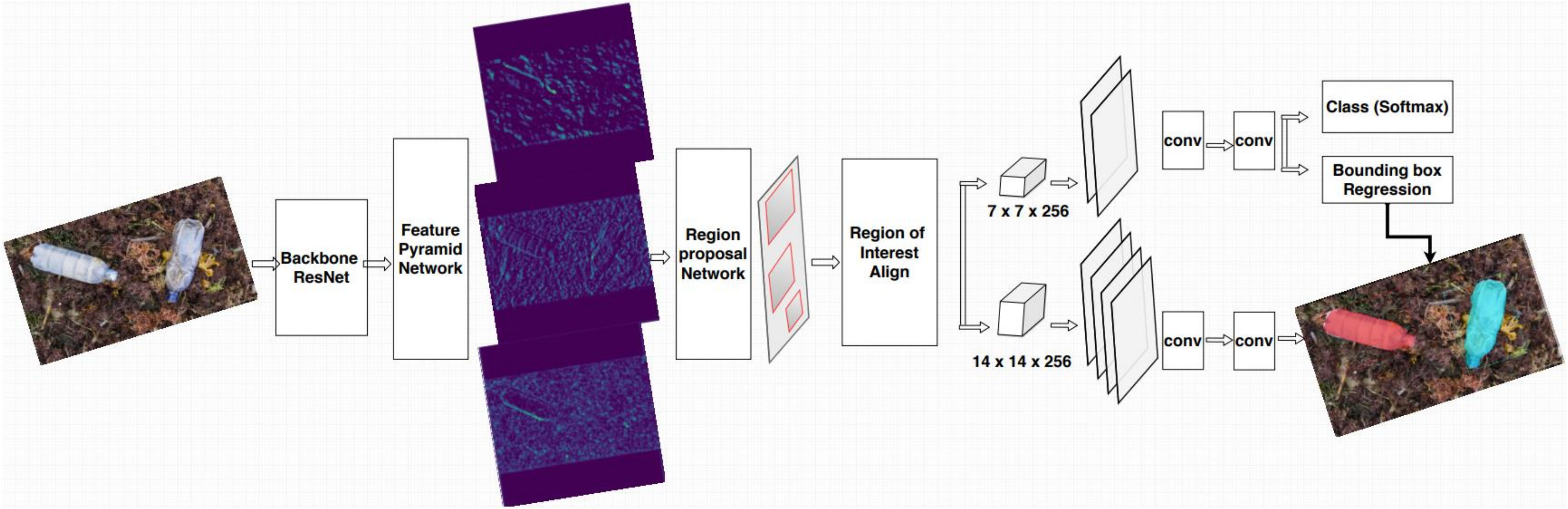
Input Image

Output

- A bounding box
- A class label (e.g., person)
- A Segmentation mask

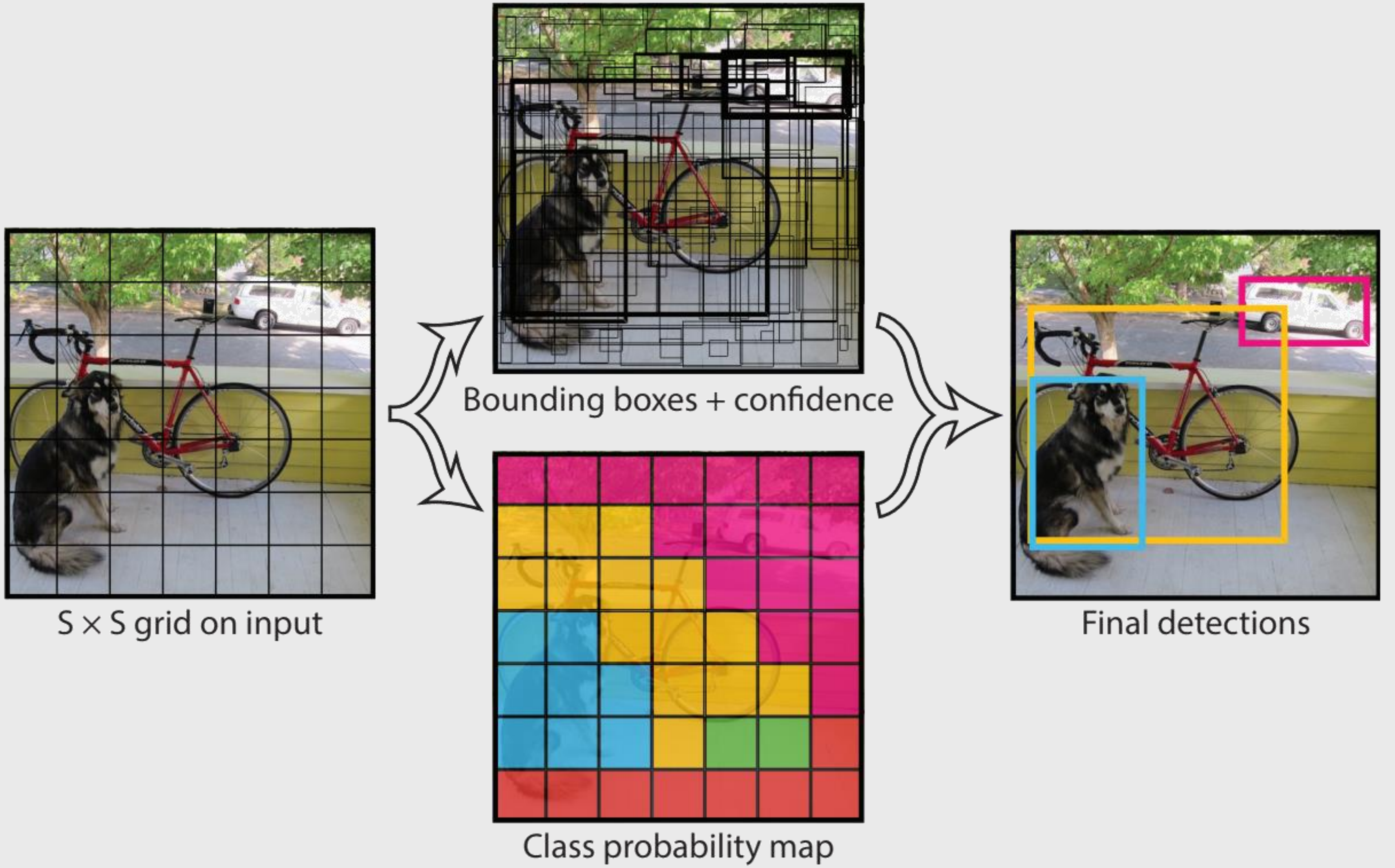
He et al. (2017) Mask-RCNN, ICCV

Mask-RCNN



Jaikumar P et al. (2020) ISDA

You **L**ook **O**nly **O**nce (YOLO) Model

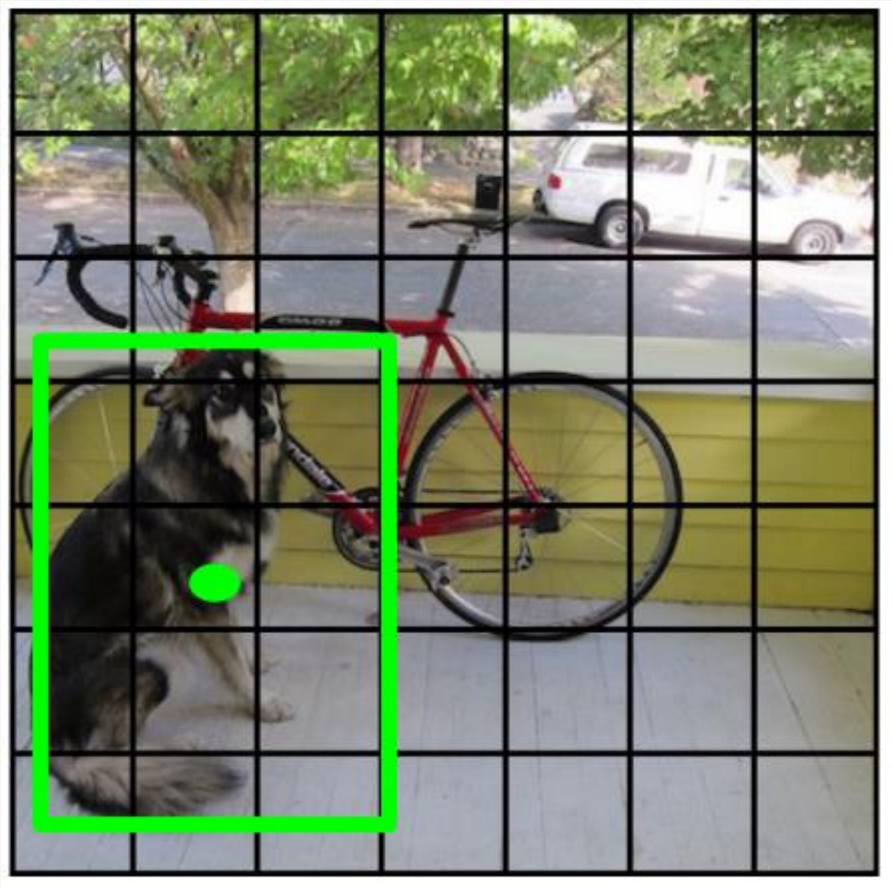


Object detection in YOLO models are as a regression problem. It divides the image into an S × S grid and for each grid cell it predicts B bounding boxes, confidence for those boxes, and C class probabilities.

Redmon, J. (2016). You only look once: Unified, real-time object detection. CVPR

You **L**ook **O**nly **O**nce (YOLO) Model

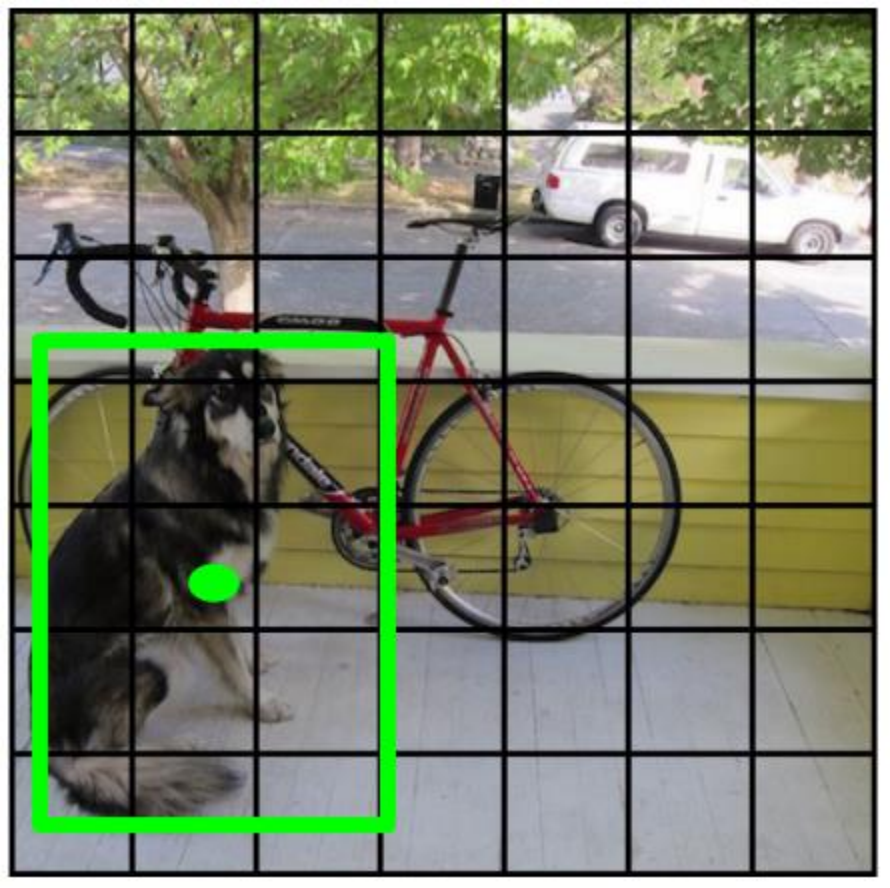
Original input



Ground Truth

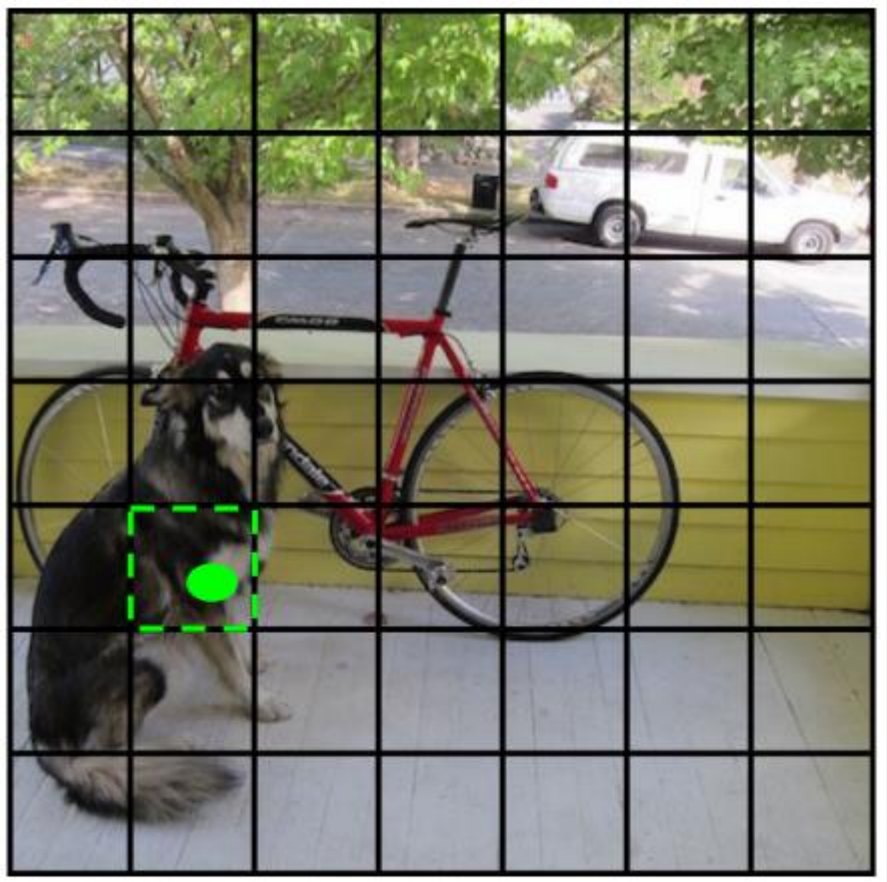
Redmon, J. (2016). You only look once: Unified, real-time object detection. CVPR

You **L**ook **O**nly **O**nce (YOLO) Model



Ground Truth

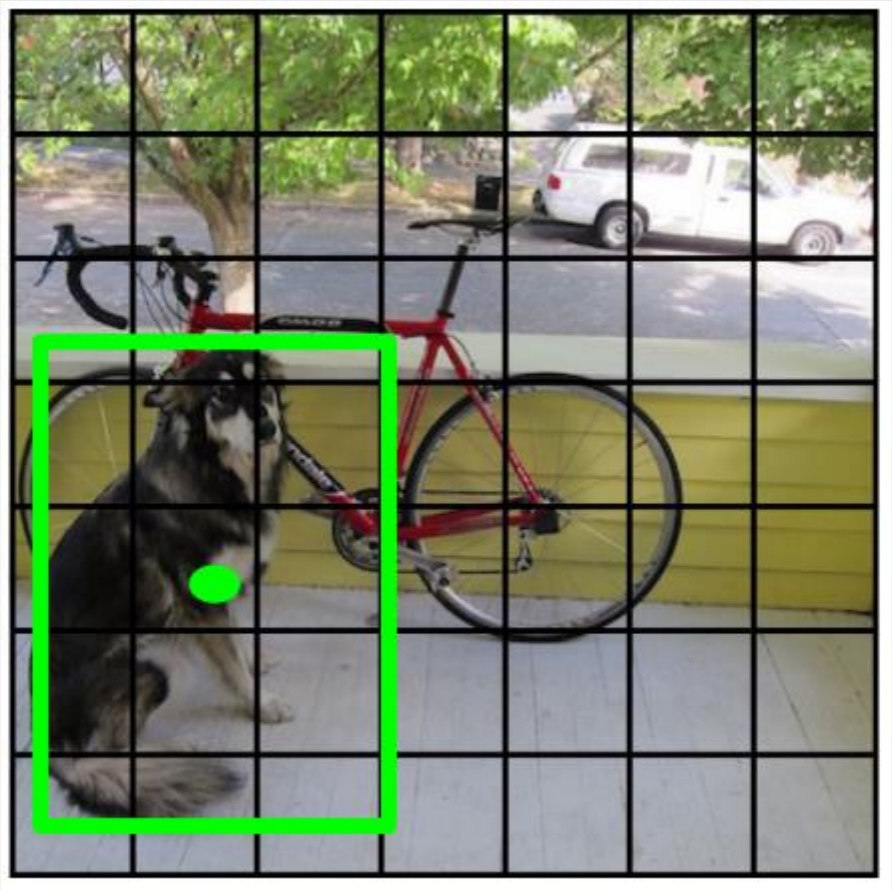
Match predicted cell with ground through



Cell Prediction

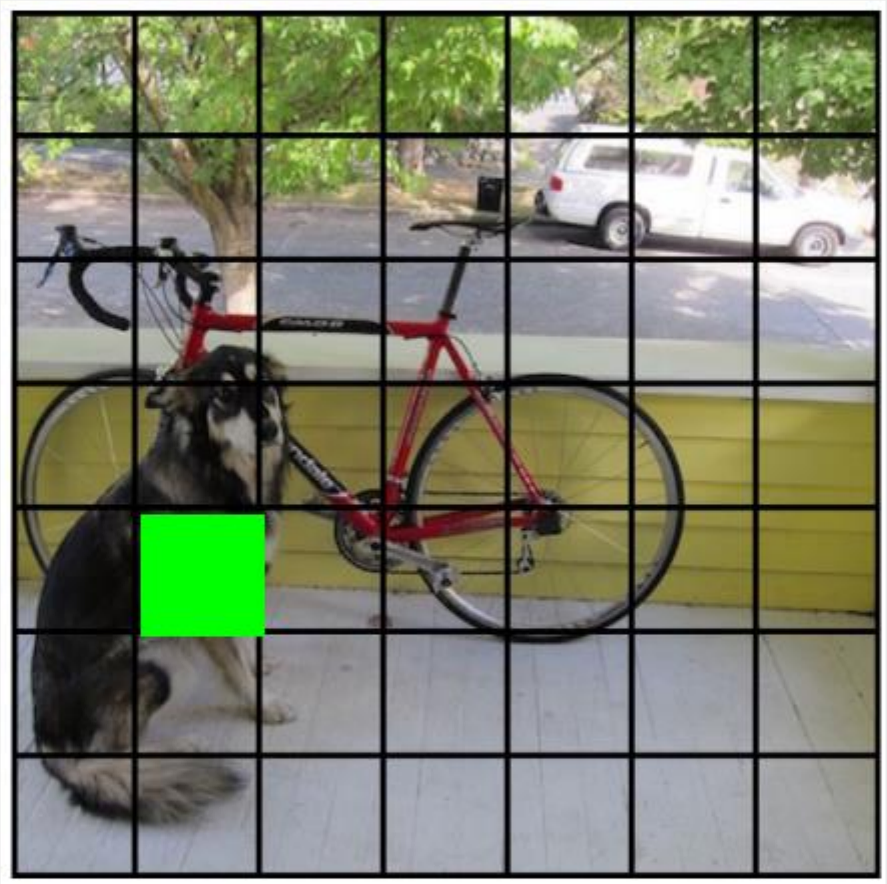
Redmon, J. (2016). You only look once: Unified, real-time object detection. CVPR

You Look Only Once (YOLO) Model



Ground Truth

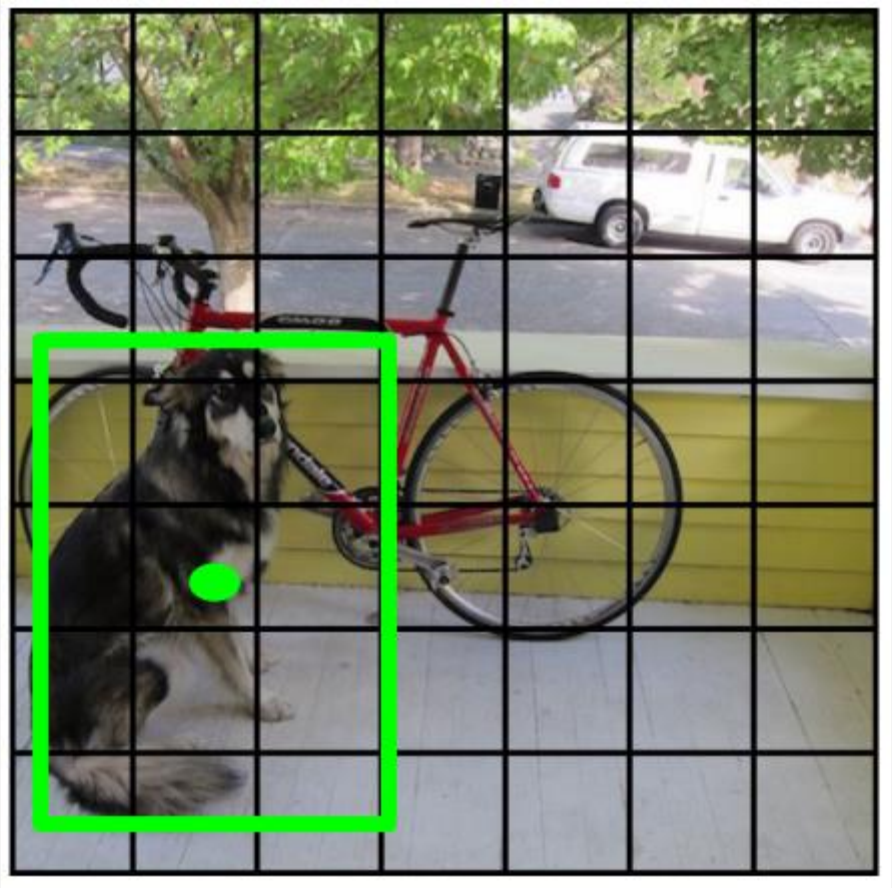
Check the class prediction (class probability, i.e., SoftMax)



Class Prediction

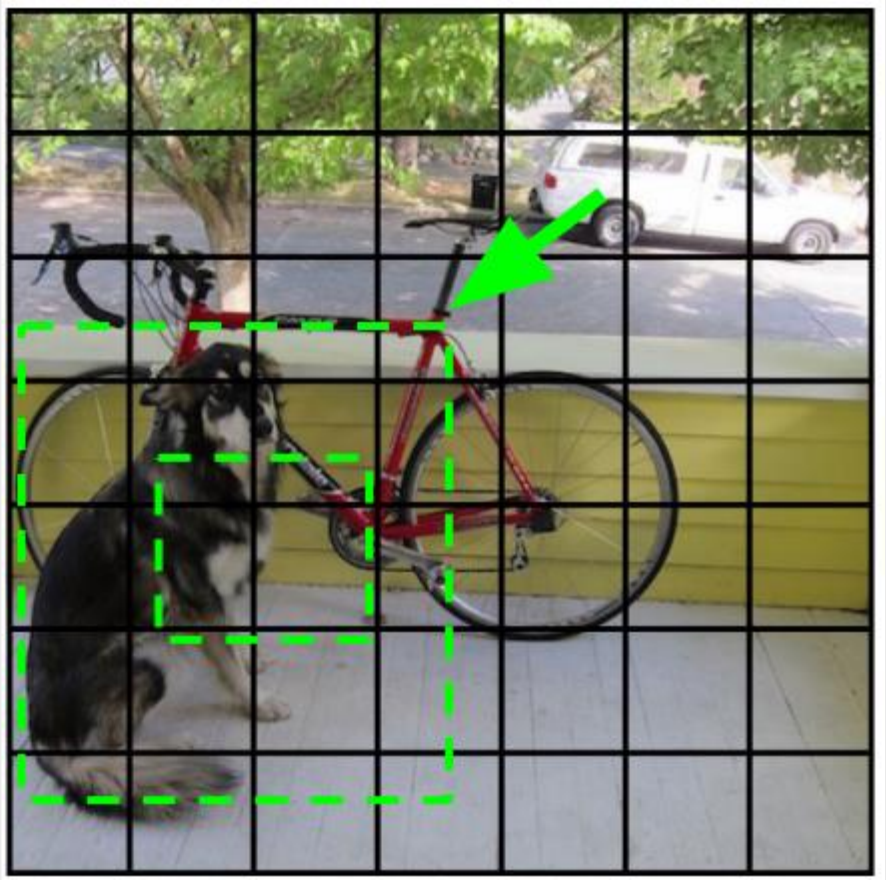
Redmon, J. (2016). You only look once: Unified, real-time object detection. CVPR

You Look Only Once (YOLO) Model



Ground Truth

Check the box prediction and its confidence (i.e., IoU)

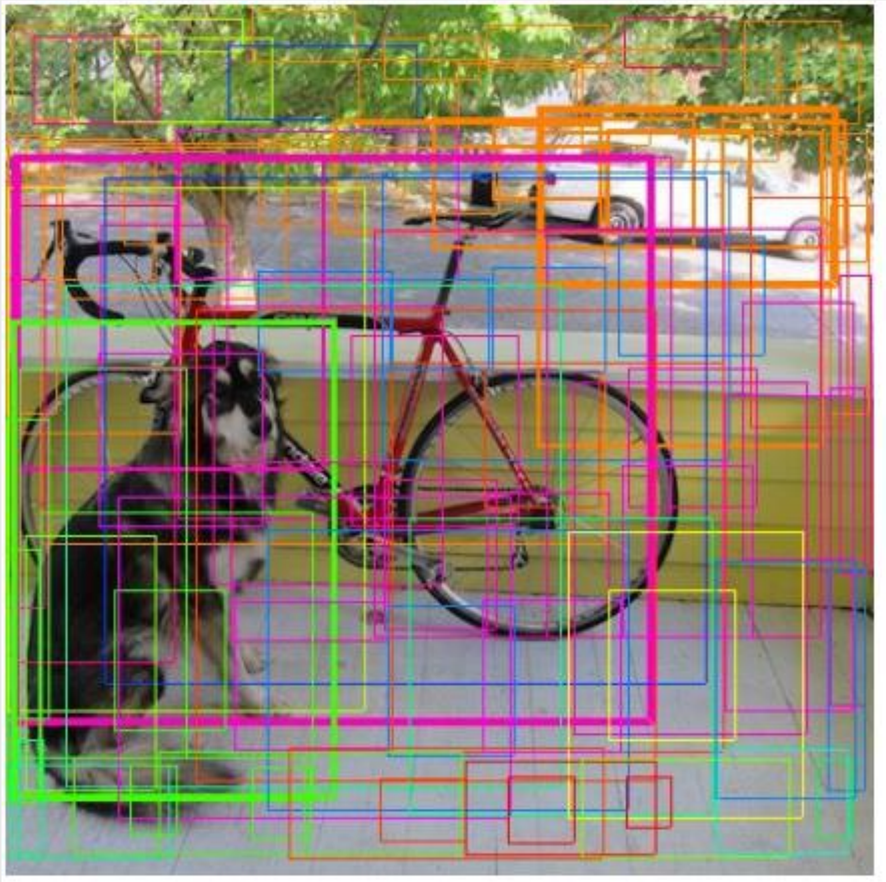


Box Prediction

Redmon, J. (2016). You only look once: Unified, real-time object detection. CVPR

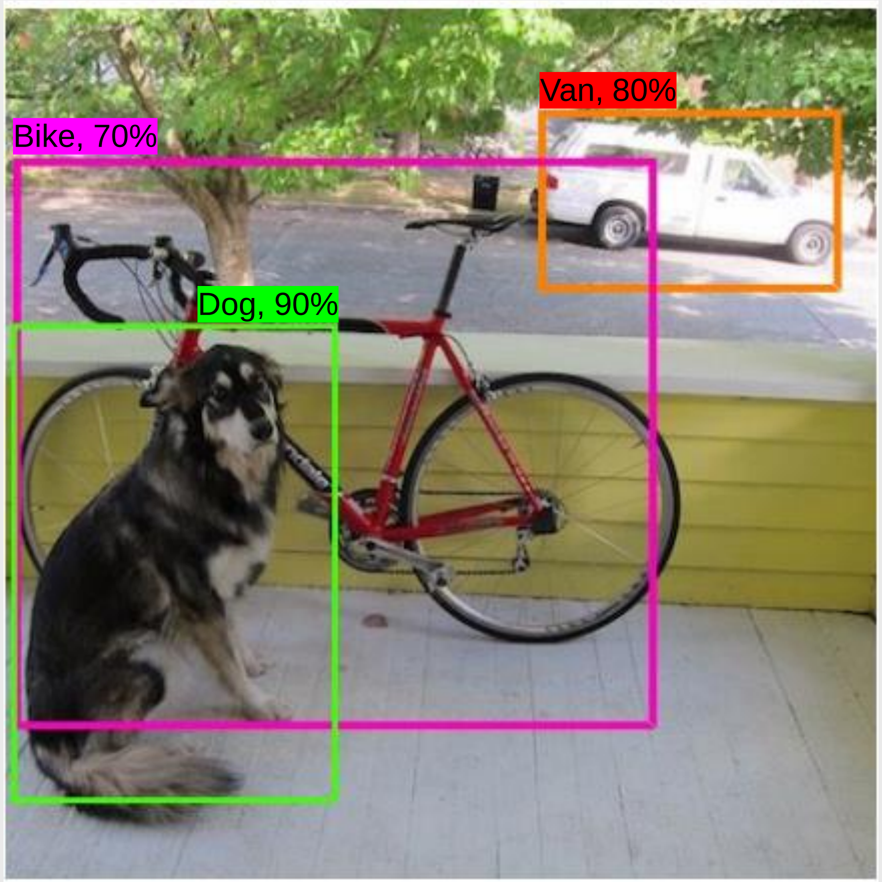
You Look Only Once (YOLO) Model

Non-maximal suppression (NMS) $Box = argmax(C(P_1), C(P_2), \ldots, C(P_n))$



Box Prediction with confidence scores

Perform NMS, i.e., keep only boxes with maximal confidence score



Final Detection

Redmon, J. (2016). You only look once: Unified, real-time object detection. CVPR

Training Loss of YOLO models

$$L_{YOLO} = L_{\text{clsss}} + L_{\text{loclization}}$$

$$L_{\text{clsss}} = \sum_{i=0}^{S^2} \mathbb{I}_i^{\text{object}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c)\right)^2$$

$$\mathbb{I}_i^{\text{object}} = \begin{cases} 1 & \text{if object in box } i \\ 0 & \text{otherwise} \end{cases}$$

$$L_{\text{loclization}} = L_{\text{confidance}} + L_{\text{coordinate}}$$

Training Loss of YOLO models

$$L_{YOLO} = L_{\text{clsss}} + L_{\text{loclization}}$$

$$L_{\text{loclization}} = L_{\text{confidance}} + L_{\text{coordinate}}$$

$$L_{\text{coordinate}} = \lambda_{\text{coordinate}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{I}_{ij}^{\text{object}} \, l$$

$$l = \left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 + (x_i - x_i)^2 + (y_i - y_i)^2$$

$$L_{\text{confidance}} = \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[\mathbb{I}_{ij}^{\text{object}} (c_i - \hat{c}_i)^2 \right] + \lambda_{\text{noObject}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \left[\mathbb{I}_{ij}^{\text{object}} (c_i - \hat{c}_i)^2 \right]$$

Where w, h, x, y, s are grid size, width, height, x-axis, and y-axis position of the box B, and the grid size

Generative Models

Which one is Real,
and which one is Fake?

1

2

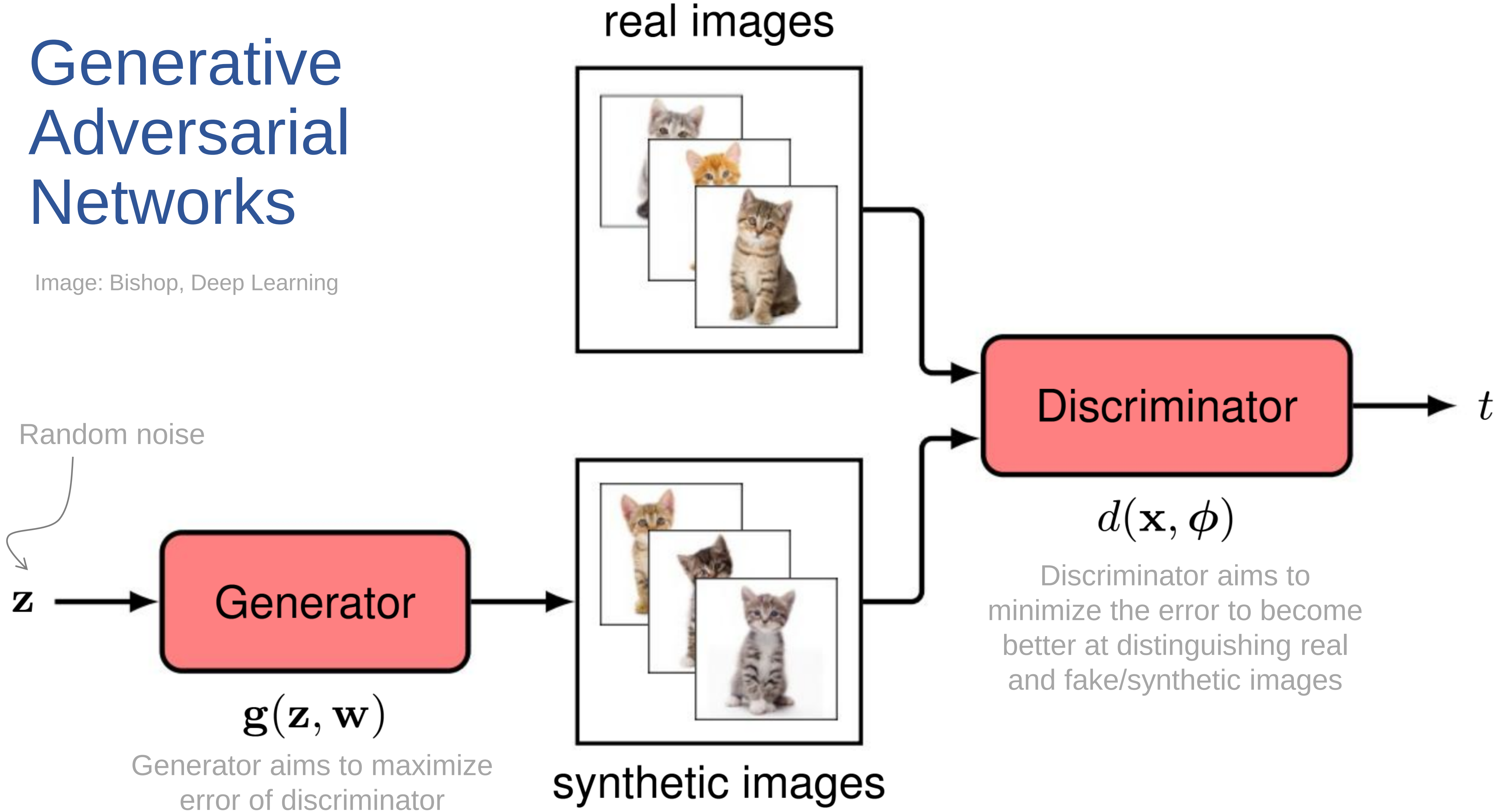
Super Realistic Generative Adversarial Networks

bicubic (21.59dB/0.6423) | SRResNet (23.53dB/0.7832) | SRGAN (21.15dB/0.6868) | original

Ledig et al.. (2017). Photo-realistic single image super-resolution using a generative adversarial network. CVPR

Generative Adversarial Networks

Image: Bishop, Deep Learning



Generator aims to maximize error of discriminator

Discriminator aims to minimize the error to become better at distinguishing real and fake/synthetic images

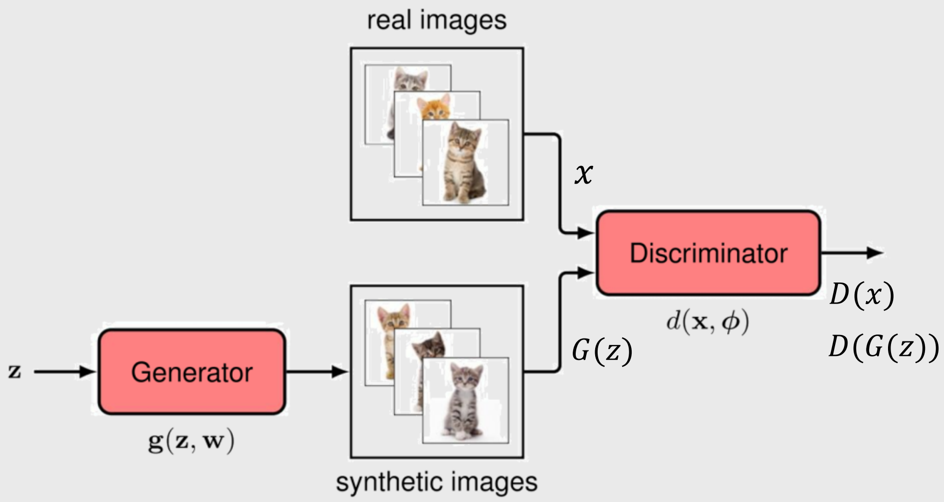
Generative Adversarial Networks

min-max loss

real image from dataset, i.e., label '1' so D should output '1' for real

Synthetic image from generator, i.e. label '0' so D should output '0' for fake

$$\min_G \max_D V(D, G) = \mathbb{E}_x\left[\log\left(D(x)\right)\right] + \mathbb{E}_z\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$



Generator aims to maximize error of discriminator, i.e., G to minimize:

$$\mathbb{E}_z\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

low value means generator create realistic images, i.e., if D(G(z)) = 1 means generator fool Discriminator and its wins

Discriminator aims to minimize the error to become better at distinguishing real and fake/synthetic images, i.e., D to maximize the probability of assigning the correct label to both training examples and samples from G by maximizing:

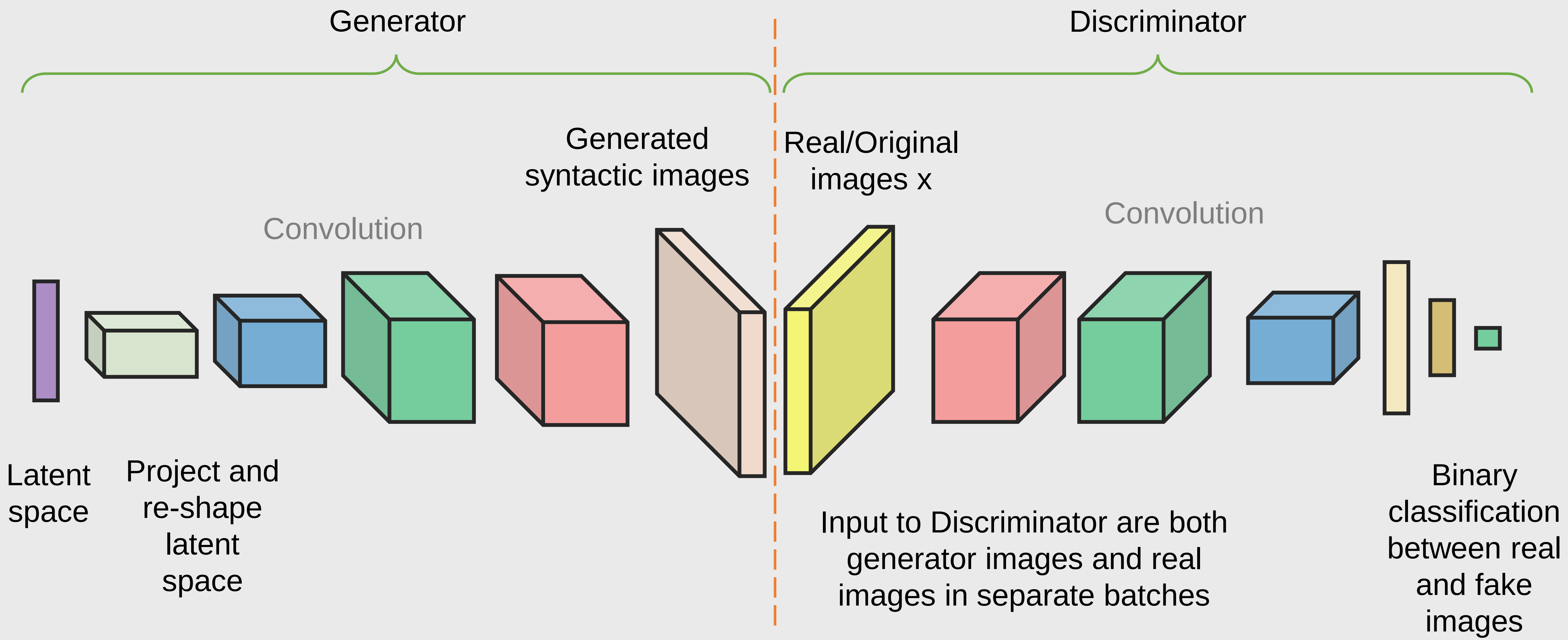
$$\mathbb{E}_x\left[\log\left(D(x)\right)\right] + \mathbb{E}_z\left[\log\left(1 - D\left(G(z)\right)\right)\right]$$

low value means discriminator is able to identify the real vs fake (i.e., if D(x) = 1 and D(G(z)) = 0 will give log 1 + Log 1 = 0) that will let Discriminator win

Goodfellow et al. (2014). Generative adversarial nets. *NIPS 2024*

Deep Convolutional GAN

The task of the generator is to produce data which the discriminator predicts as being ‘real’, meaning that it closely resembles the training dataset.



Radford et al (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. ICLR 2016

StyleGAN to Generate China City Scape

My student project

(Jia 2022)