COMPUTER VISION

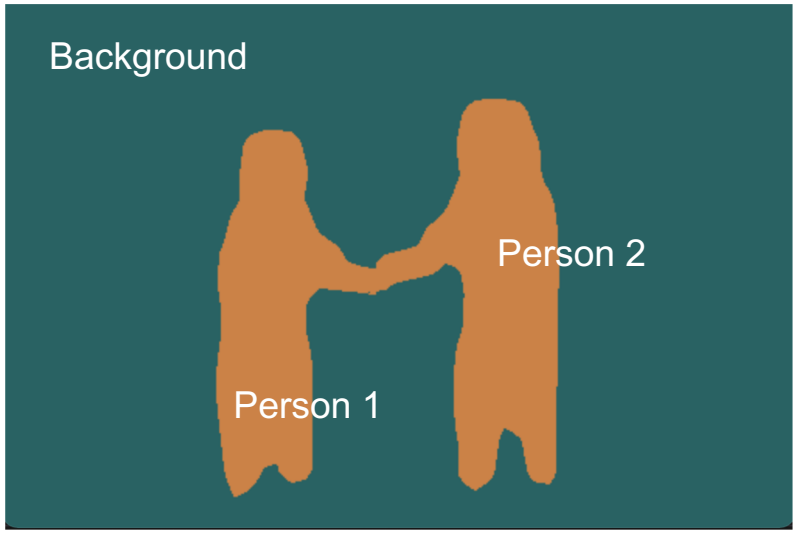
IMAGE SEGMENTATION

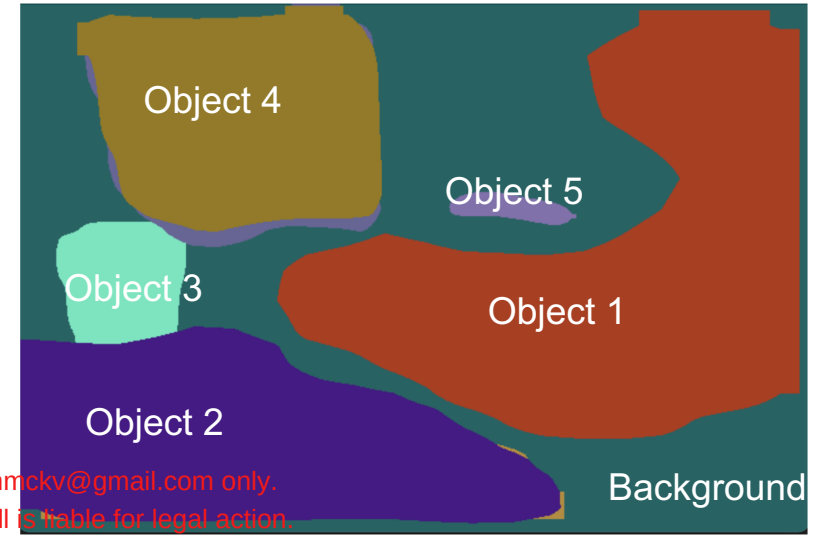
Agenda

- Semantic and Instance Segmentation: Task & Output
- Applications of image segmentation
- Approaches to solve semantic segmentation
  - Fully convolutional (FCN)
  - Encoder-Decoder (U-Net)
- Downsampling and upsampling methods
- Atrous (dilated) convolutions
- Mask R-CNN for instance segmentation
- ROI Align layer
- Loss Functions
- Hands-On -Python notebook

Semantic and Instance Segmentation: Task & Output

- Fine-grained inference of the image
- Label each pixel in the image
- **Semantic Segmentation**:- segment each pixel of the image and identify different objects
  - Extended classification where each pixel is classified into the n_classes.
  - No need to specify the difference between the instances
- **Instance Segmentation:-**
  - Specify each instance of the object

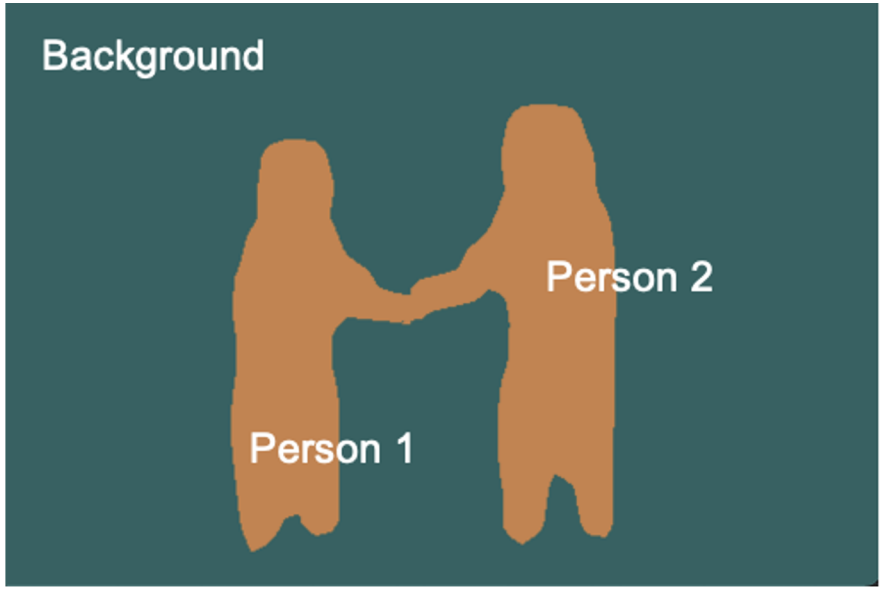




Semantic and Instance Segmentation: Task & Output

Input and Output format:-

- Input: an image
- Binary segmentation (pixelwise yes / no)
  - When you have two objects (or one object and a background)
  - Output shape: (batch_size, cols, rows, 1)
  - Sigmoid can be used in the output layer
- Multiclass segmentation (pixelwise probability vector)
  - When you have multiple objects
  - Output shape: (batch_size, cols, rows, n_classes)
  - Softmax can be used in the output layer



Applications of Image Segmentation

- Virtual reality
- Handwriting recognition
- Face Recognition
- Self-driving cars
- Fashion industry:- apparel trials
- Medical Image Segmentation
- Satellite (Or Aerial) Image Processing

Approaches to solve Semantic Segmentation

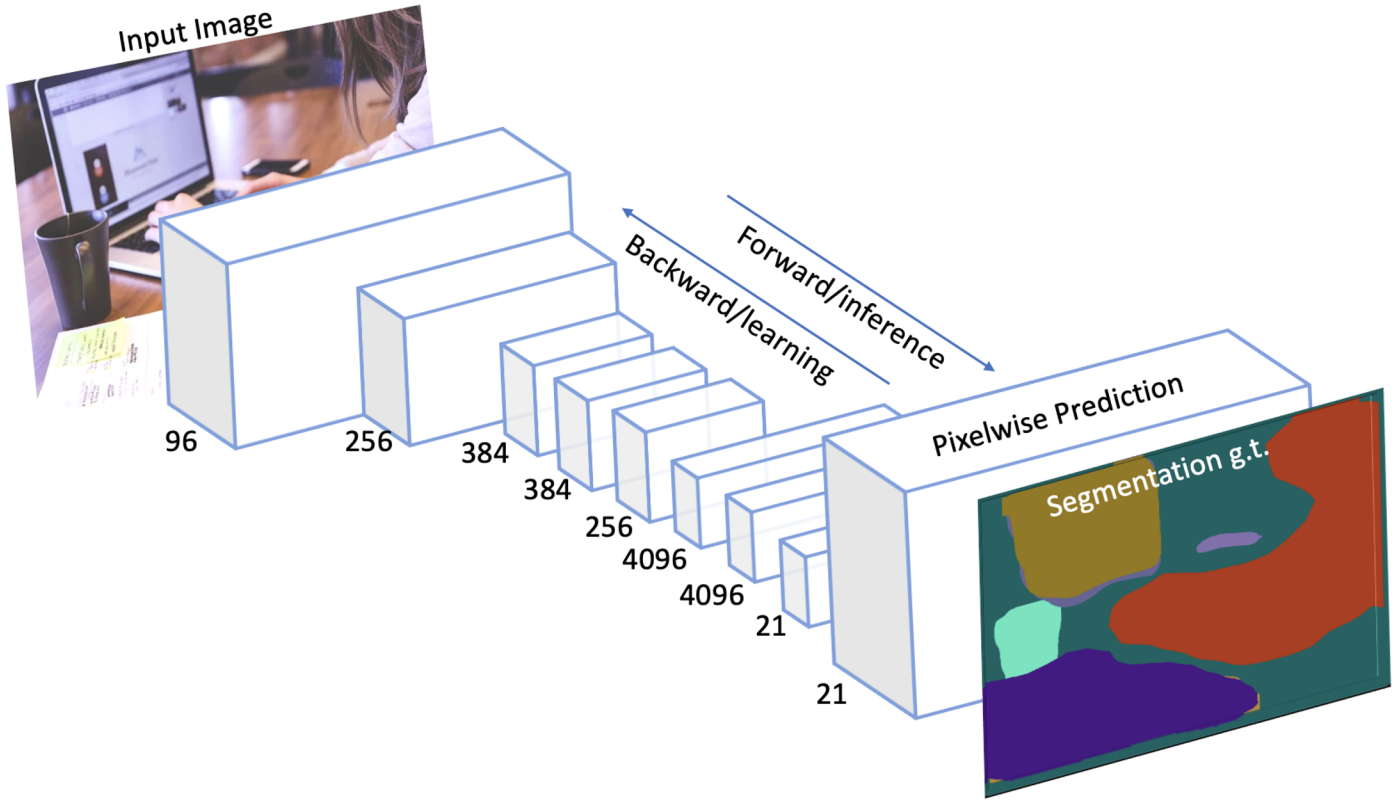
- Classical machine learning methods
- Fully Convolutional Network
- Encoder-decoder (UNet)
- Pyramid Scene Parsing Network (PSPNet)

Fully Convolutional Networks (FCN) - Architecture

- One of the simplest and popular architecture
- They employ solely locally connected layers, such as convolution, pooling and up-sampling
- Convnets are built on translation invariance
- Convnet basic components are : convolution, pooling and activation functions
- Each layer of data in a convnet is a three-dimensional array of size h × w × d
- First layer is an image
- Locations in higher layers correspond to the locations in the image they are path-connected to
- The network consists of :
  - A down-sampling path, used to extract and interpret the context and
  - An up-sampling path, which allows for localization

FCN for Semantic Segmentation

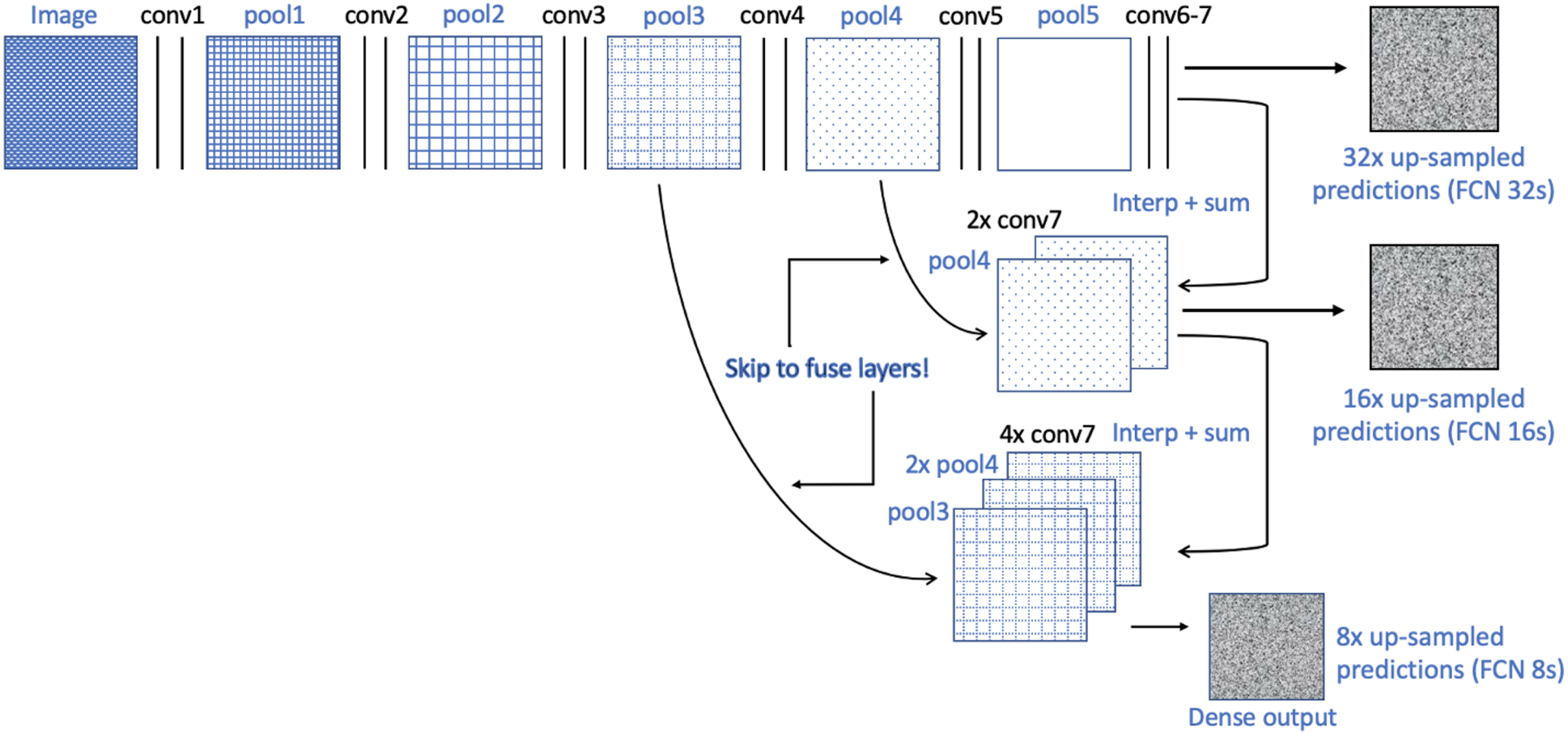
- The features are merged from different stages in the encoder
- Low resolution semantic feature maps are learned
- Up-sampling of these maps are done using transposed convolutions(deconvolution)
- These are initialized with bilinear interpolation filters



FCN - Down-sampling, Up-sampling and Skip Connections

- Down-sampling
  - Develops complex feature maps by down-sampling spatial resolution of the input image.
  - It helps to discriminate different classes.
  - Object localization information is lost.
- Up-sampling
  - Object location information can be recovered.
  - Input – multiple lower resolution images.
  - Output – high resolution segmentation map.
- Skip connections
  - Downsampling reduces resolution of Input image by large factor.
  - During up-sampling output obtained is coarse(rough and not smooth).
  - Adding skip connections from earlier layers in up-sampling stage generate accurate segmentation boundaries.
  - Combination of fine and coarse layers – local prediction with nearly accurate spatial structure.

FCN – Architecture (Semantic Segmentation)



FCN – Skip Connections

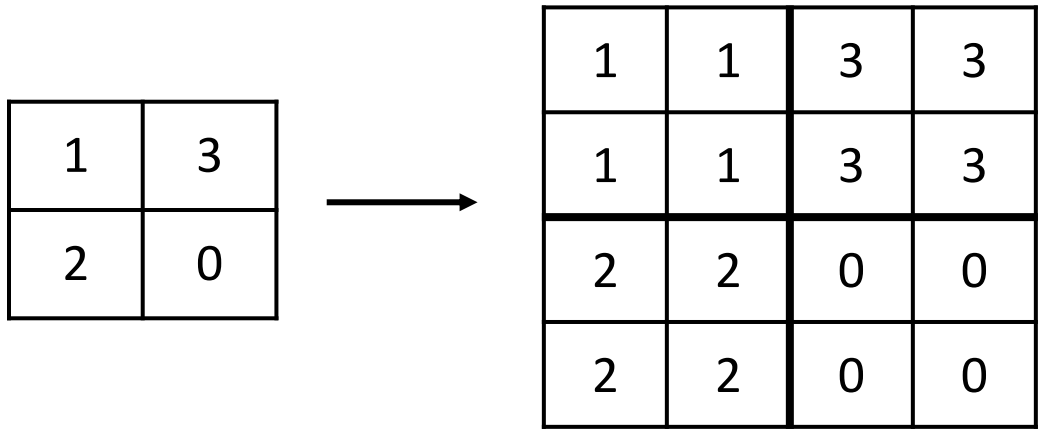
- **FCN-32s**
  - It up-samples at a stride of 32.
  - The predictions are back to pixels in single step (Layer with skip connections).
- **FCN-16s**
  - It up-samples at a stride of 16.
  - It combines output/predictions from the previous layer and pool4 layer.
  - Output has finer details than FCN-32s.
- **FCN-8s** :
  - It up-samples at a stride of 8.
  - It combines predictions from the previous layer and pool3 layer
  - Output has even precise boundaries compared to FCN-16s

- Skip connections are considered boosting methods in FCN which increase the performance of FCN by using Feature maps (Predictions) from previous layers

Upsampling Techniques

1. Nearest Neighbours

- Input pixel value is taken and copied to the K-Nearest Neighbors
- Where K depends on the expected output

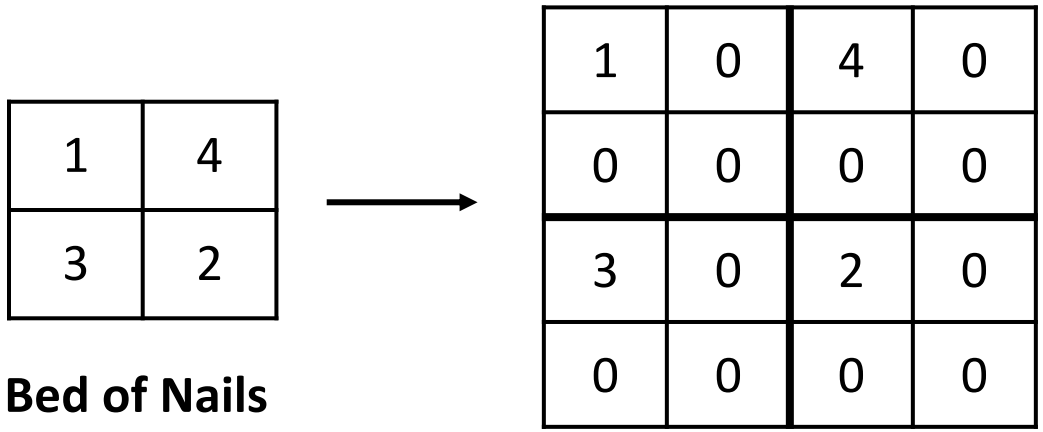


Input: 2 X 2 **Output: 4 X 4**

Upsampling Techniques

2. Bed of Nails

- We copy the value of the input pixel at the corresponding position in the output image and
- Filling zeros in the remaining positions



Bed of Nails

Input: 2 X 2

Output: 4 X 4

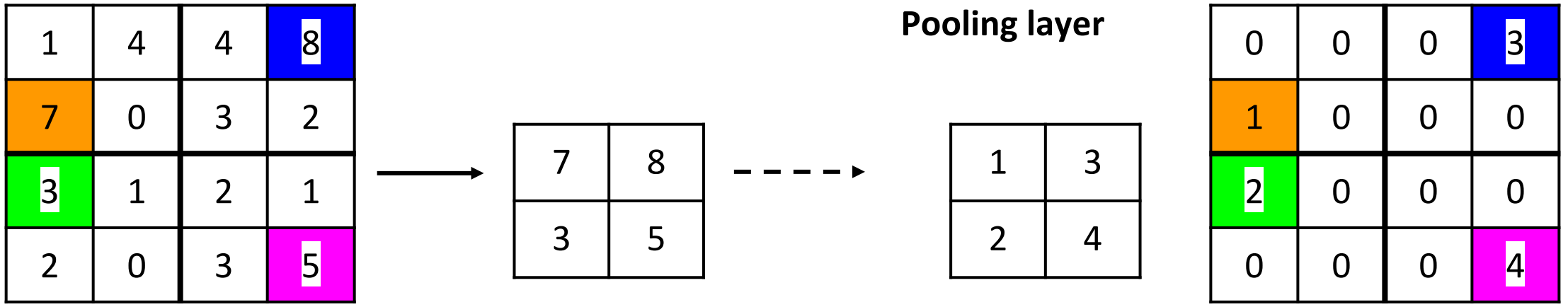
Upsampling Techniques

3. Max Un-pooling

- Max-Pooling layer in CNN
  - Takes the maximum among all the values in the kernel
- To perform max un-pooling -
  - First, the index of each maximum value is saved during the encoding step.
  - During the Decoding step, the input pixel is mapped to the saved index, filling zeros everywhere else

Upsampling Techniques

Max Un-pooling Example



Input: 4 X 4

Output: 2 X 2

Input: 2 X 2

Output: 4 X 4

Max Pooling
Remember each max element

Max Unpooling

Use positions for Pooling layer

Drawback of above techniques

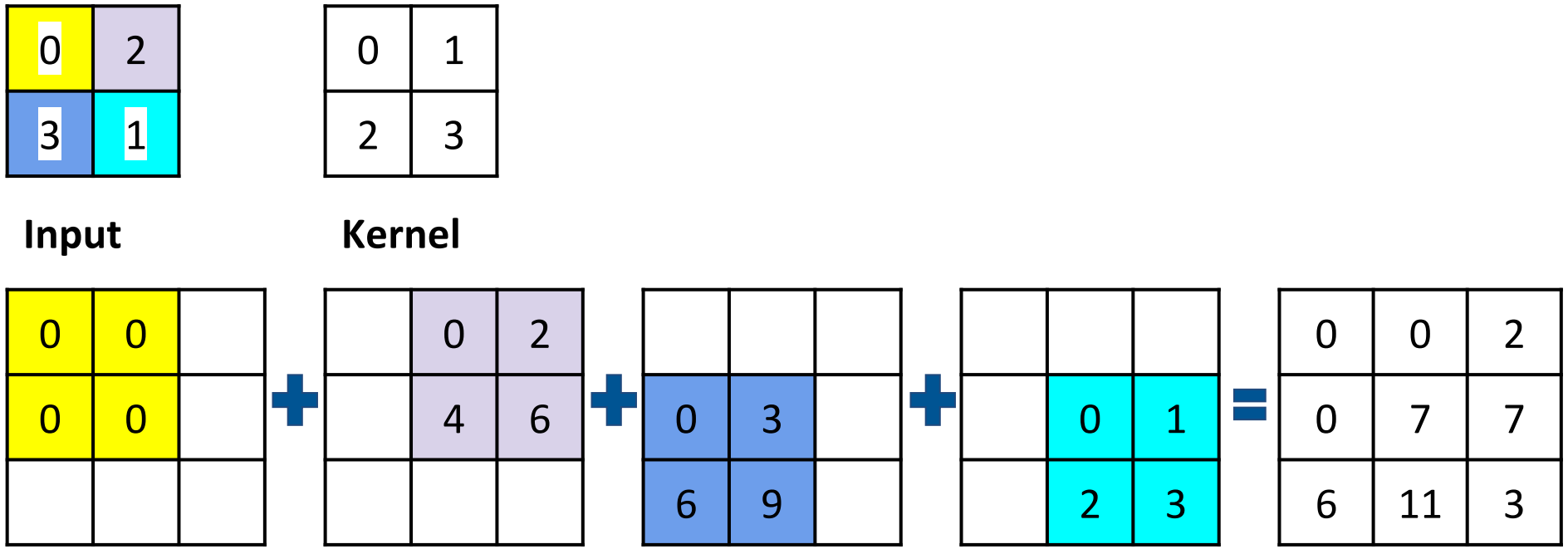
- All are predefined and do not depend on data
- Task-specific
- They do not learn from data and hence are not a generalized technique

Transposed Convolutions for Upsampling

- Used to upsample the input feature map to a desired output feature map
- Uses learnable parameters
- Basic operation involves -
  - Consider a 2x2 encoded input feature map
  - Which needs to be upsampled to a 3x3 output feature map
  - Take a kernel of size 2x2 with unit stride and zero padding
  - Now take the upper left element of the input feature map and multiply it with every element of the kernel as shown in figure 3
  - Repeat for all remaining elements

Transposed Convolutions for Upsampling

- For over-lapped resulting upsampled feature maps, simply add the elements of the overlapping positions
- Resultant map is the final upsampled feature map with desired size 3X3



Atrous (Dilated) Convolutions

- One disadvantage in Deep CNN (DCNN) is -
  - Repeated combination of max-pooling and striding at consecutive layers reduces
    - Spatial resolution of the resulting feature maps, typically by a factor of 32 across each direction
- One of the solution is Dilated convolution
- Allows us to compute the responses of any layer at any desirable resolution
- Allows for dense feature extraction using “rate r” parameter
- Weights of an atrous convolution kernel are spaced **r** locations apart
- i.e. the kernel of dilated convolution layers are sparse

Atrous (Dilated) Convolutions

- By controlling the rate parameter, we can arbitrarily control the receptive fields of the conv. layer
- This allows the conv. filter to look at larger areas of the input(receptive field) without a decrease in the spatial resolution or increase in the kernel size

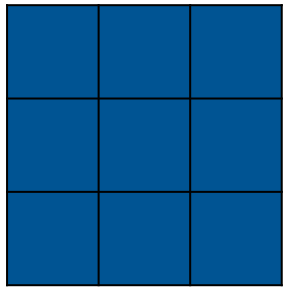


Figure 3a: 3X3 kernel

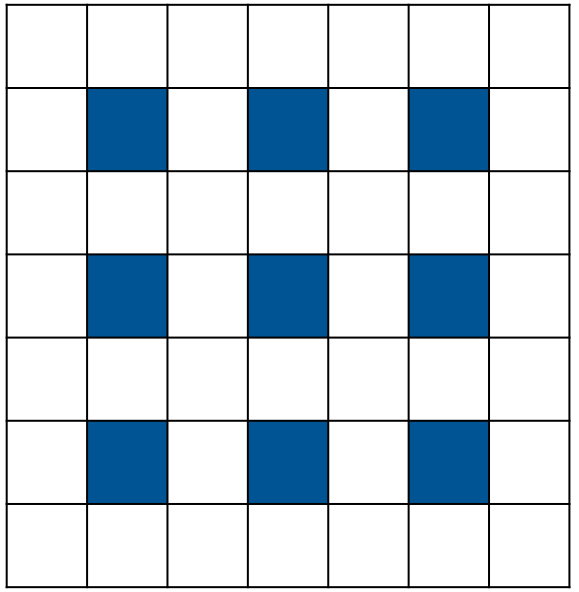


Figure 3b: 3X3 dilated kernel with r = 2

Figure 3: Standard vs Dilated kernel

Comparison between FCN & U-Net

FCN

- It has only one layer in the decoder (Up-Samples only once)
- For upsampling it uses bilinear interpolation technique which has no learnable filter.
- Variants of FCN-[FCN 16s and FCN 8s] add the skip connections from lower layers to make the output robust to scale changes

U-Net

- It has multiple upsampling layers
- It concatenates using skip connections instead of adding up
- It uses learnable weight filters instead of fixed interpolation technique

U-Net gives better performance because it has multiple up-sampling layers along with more skip connections which theoretically make it more robust to scale variations as compared to FCN

U-Net

- General CNN focuses on image classification task.
- U-Net does both localization and classification.
- It was evolved in 2015 to process biomedical images.
- It does classification on every pixel.
- Input and shares the same size. Eg: Input image size 2x2, Output image size 2x2.
- It has symmetric architecture.
- Three major parts :

  Left part - Down-sampling path / Contracting path - General convolution process (Encoder)

  Horizontal bottleneck

  Right path – Up-sampling / Expansive path - Transposed 2d convolution layers (Decoder)

U-Net Architecture

- One important modification from FCN architecture -
  - A large number of feature channels are also added in the upsampling part
- This allows the network to propagate context information to higher resolution layers
- As a result, expansive path is more or less symmetric to the contracting path
- Hence, the **U-shaped** architecture
- Only uses the valid part of each convolution i.e. -
  - Pixels which has full context in input image are **only** present in segmentation map

U-Net Architecture

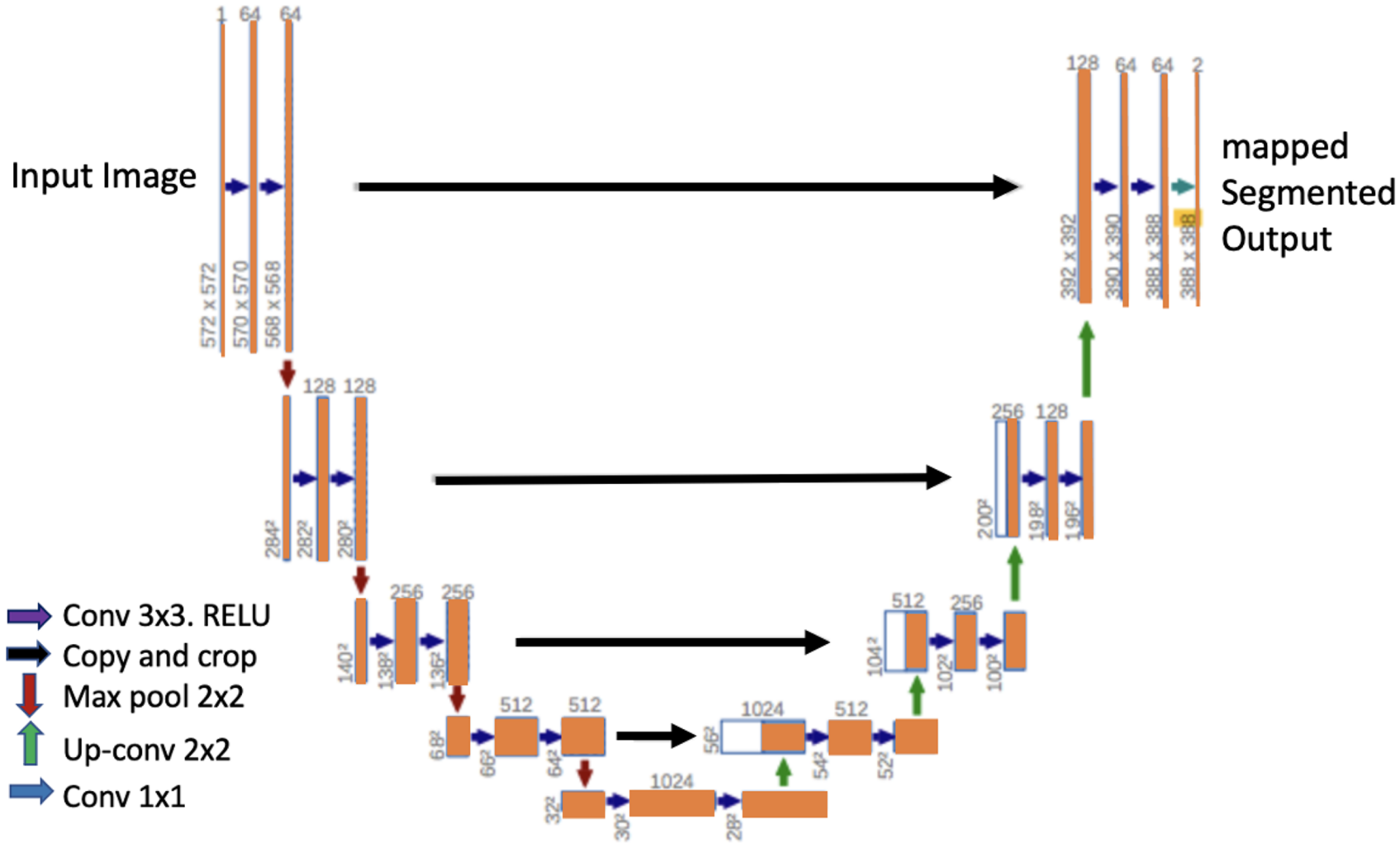


Figure 1: U-Net Architecture

U-Net Architecture

- **Contracting or down-sampling path**
  - It consists of 4 blocks
  - For each block - Two 3x3 convolution (+batch norm) followed by 2x2 max-pooling.
  - The number of features maps are doubled at each pooling layer (after each block) as 64 -> 128 -> 256 and so on.
- **Horizontal bottleneck**
  - It consists of two 3x3 convolution followed by 2x2 up-convolution.
- **Expanding or up-sampling path**
  - It is opposite/complimentary to the contracting path
  - It also consists of 4 blocks
  - For each block - Two 3x3 conv followed by 2x2 upsampling (transpose convolution).
  - The number of features maps here after every block gets halved.
- **Skip connections**
  - It is used to localize the object.
  - The feature maps are concatenated and then goes through some conv layers for additional processing.

Case Study on Semantic Segmentation

Instance Segmentation

- In semantic segmentation multiple objects of the same class are considered as a single entity
- In instance segmentation multiple objects of the same class are considered as distinct individual instances.
- In this section the discussion will be based on distinct individual instances using Mask RCNN algorithm.

Mask R-CNN

- It is Built on top of Faster R-CNN.
- It has an additional branch to predict segmentation mask on ROI in a pixel-pixel manner.
- It has 3 outputs, one for classification, other for regression and the third for object mask.
- It is divided into two parts
  - Region Proposal Network (RPN) – Candidate object bounding boxes
  - Binary Mask Classifier – Mask generation for every class

Mask R-CNN

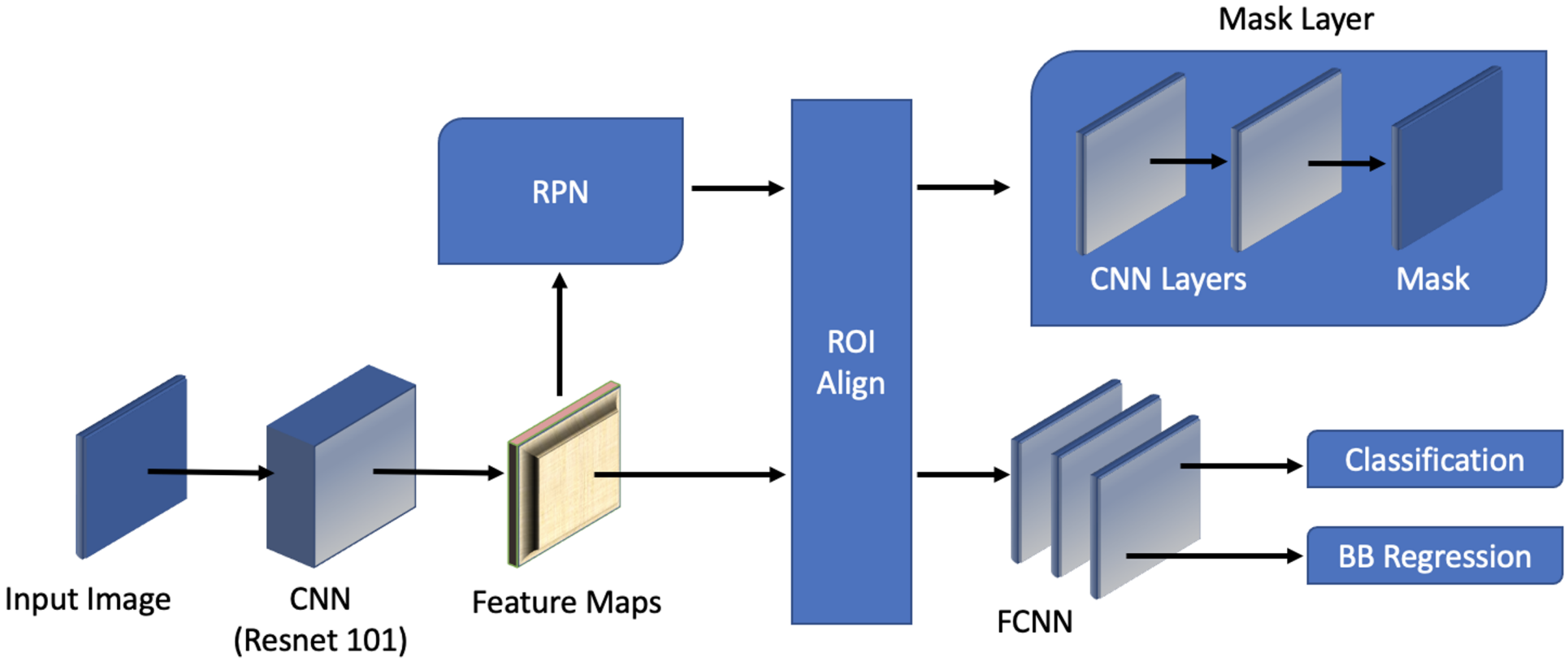


Figure 4: Architecture of Mask R-CNN

Mask R-CNN for Instance Segmentation

- Input image is passed on to CNN to extract features.
- Features are extracted using Resnet 101 architecture.
- Apply Region proposal Network (RPN) on the previously obtained features
- This basically predicts whether object is present or not in the regions
- ROI align is applied to convert all regions from RPN and feature map to same shape

These regions are then passed to

- FCN -
  - To predict class labels(classification) and bounding boxes(regression)
  - ROI are computed, for all predicted regions IOU is calculated
  - IOU > 0.5 is **only selected and others are neglected**
- Mask branch - predicting segmentation mask

Mask R-CNN for Instance Segmentation

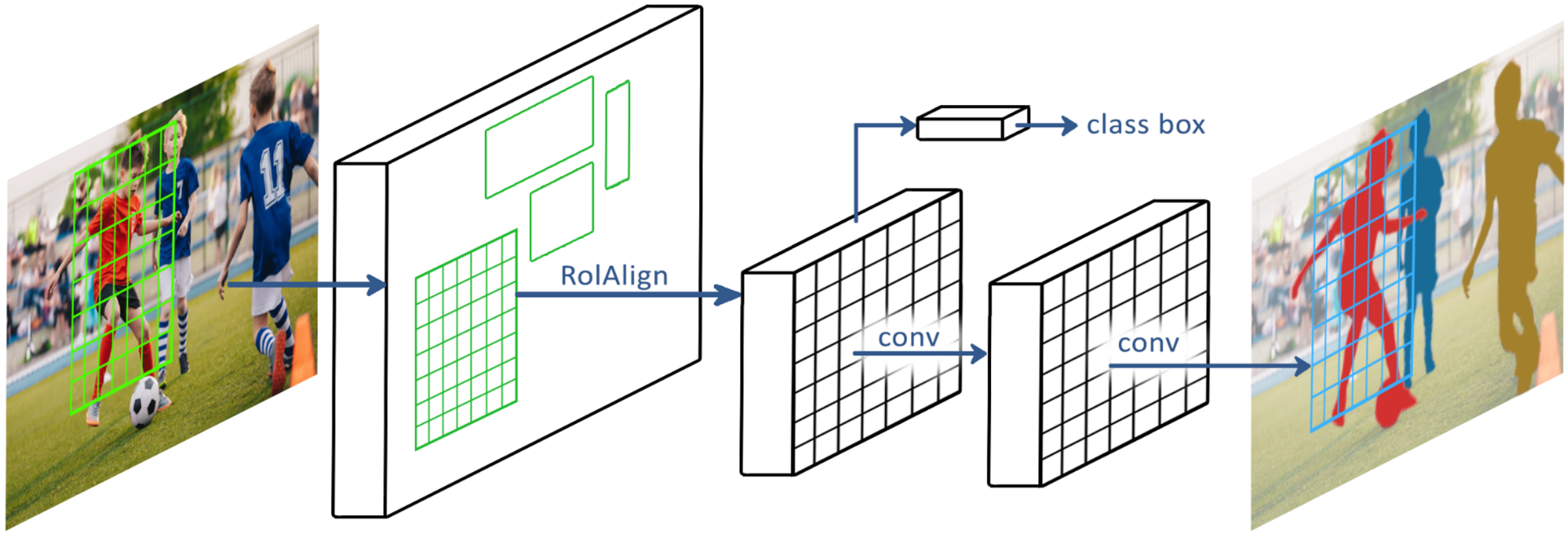


Figure 5: Mask R-CNN for Instance Segmentation

Loss Functions

- Deep Learning algorithms use stochastic gradient descent approach to optimize and learn the objective
- We need to ensure that the loss functions are able to cover even the edge cases

Loss Functions

Type	Loss Function
Distribution-based Loss	Binary Cross-Entropy Weighted Cross-Entropy Balanced Cross-Entropy Focal Loss
Region-based Loss	Dice Loss Sensitivity - Specificity Loss Log-Cosh Dice Loss
Boundary-Based Loss	Hausdorff Distance Loss Shape aware Loss
Compounded Loss	Combo Loss Exponential Logarithmic Loss

Table 1: Different Types of Loss Functions in Semantic Segmentation

Different Types of Loss Functions

1. **Binary Cross-Entropy**

- Cross entropy loss or log loss - measures the classification model performance.
- Output – Probability value between 0 to 1
- Increase or decrease in loss value - divergence between actual value and predicted probabilities.
- Binary cross entropy – Loss function used in binary classification task
- Binary cross entropy is the negative average of the log of corrected predicted probabilities

$$\text{Loss} = -\frac{1}{output\ size}\sum_{i=1}^{output\ size} y_i . \log \hat{y_i} + (1 - y_i) . \log(1 - \hat{y_i})$$

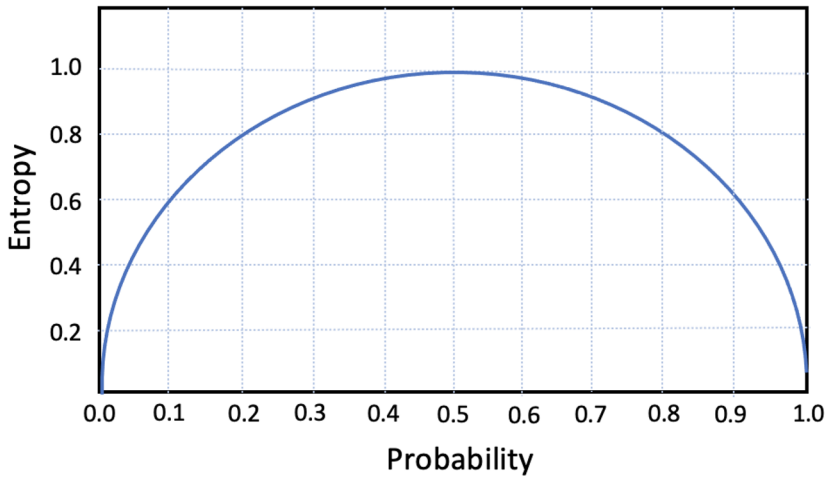


Figure 6: Binary Cross-Entropy Loss Function

Different Types of Loss Functions

Great Learning

2. **Dice Loss**

- It is used in medical image segmentation task.
- It is used to address data imbalance problem.
- It measures relative overlap between the prediction and the ground truth (IOU).

Dice Loss = 1 - Dice Coefficient

Dice Coefficient =

$$\frac{2\sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

Where p_i and g_i represent pairs of corresponding pixel values of prediction and ground truth, respectively.

Summary

- We have discussed -
  - Semantic & Instance Segmentation
  - FCN & U-Net approaches for semantic segmentation
  - Downsampling & Upsampling methods
  - Dilated Convolutions
  - Mask R-CNN and ROIAlign Layer
  - Different Loss Functions

Hands On Case Study

Thank You

Happy Learning :)