



# Cartoonify Human in an Image

Group-14

*Shariq Suhail*  
*Giri Pranay*  
*Bhupati Badhavath*  
*Likhith Lanka*



# Characteristics of Cartoonish Images

- Cartoon images are sharper.
- Cartoon images have more saturated colours than real life image.



# Cartoonification Algorithm

1. Detecting and boldening of the edges.
  2. Smoothing and quantizing the colors in the image.
- The resulting images from step-1 and step-2 are combined to achieve the cartoonish effect.



# Creating Homogeneous Color Regions

- Smoothing operation has to be done to get homogeneous color regions.
- Performing smoothing operation using gaussian filtering will blur the edges. But the edges have to be preserved to get the cartoonish effect.



# Bilateral Filtering

- A **bilateral filter** is a non-linear, edge-preserving, and noise-reducing smoothing filter for images.
- Similar to gaussian filtering , it replaces the intensity of each pixel with a weighted average of intensity values from nearby pixels.
- The difference is that the weights are further adjusted depending on how different the pixel values are.

# Bilateral Filtering

$$BF[I]_p = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(\|I_p - I_q\|) I_q$$

*space weight*      *range weight*

range kernel for smoothing  
differences in intensities

spatial kernel for smoothing  
differences in coordinates

$$w(i, j, k, l) = \exp\left(-\frac{(i - k)^2 + (j - l)^2}{2\sigma_d^2} - \frac{\|I(i, j) - I(k, l)\|^2}{2\sigma_r^2}\right)$$

$$p = (i, j) \quad , \quad q = (k, l)$$



## Bilateral Filter for edge preserving

- A pixel that is close in color to the centroid pixel will have a higher weight than a pixel at the same distance with a more distinct color.
- This extra step in the weight calculation is important because it preserves the edges.
- Parameters:  $\sigma_{\text{Color}} = 9$ ,  $\sigma_{\text{Space}} = 7$ ,




## Drawbacks of Bilateral Filtering

The bilateral filter run-time is dependent on the kernel size. It is orders of magnitudes slower than other smoothing operators like Gaussian blur.

Running more iterations with a smaller kernel is faster than applying one bilateral filter with a large kernel size with less iterations.

We have used 9x9 kernel for 7 number of iterations.





## Detecting and boldening of the edges.

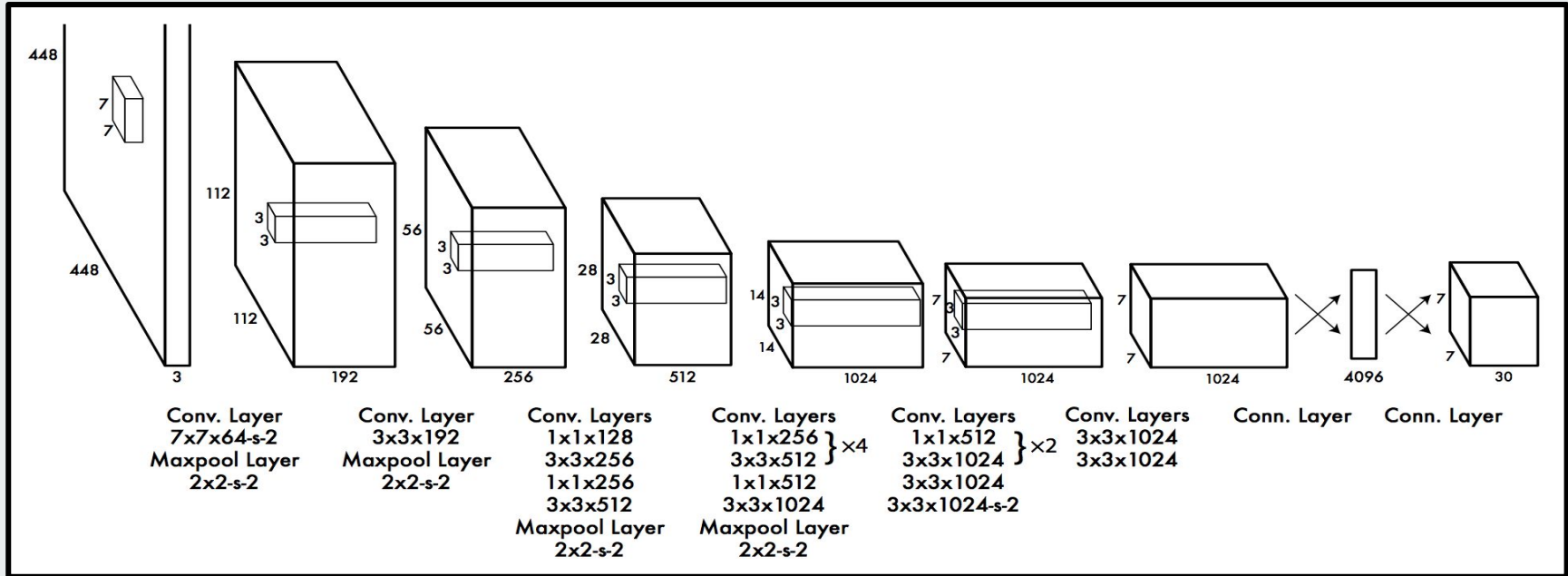
- Adaptive thresholding algorithm has been used for detecting and boldening edges in an image.
- As a pre-processing step, median filter is applied to reduce any salt and pepper noise that may be in the image.
- The median filter is set to  $7 \times 7$  which is small enough to preserve edges in the image. Too large a kernel size would result in washed out edges.



# Human Detection from input image

- Our project is to cartoonify **humans** in an image. Hence we have to detect bounding boxes around persons in an image.
- Apply cartoonification algorithm to each of the bounding box.
- We have used **YOLO** object detection algorithm to detect persons in an image.

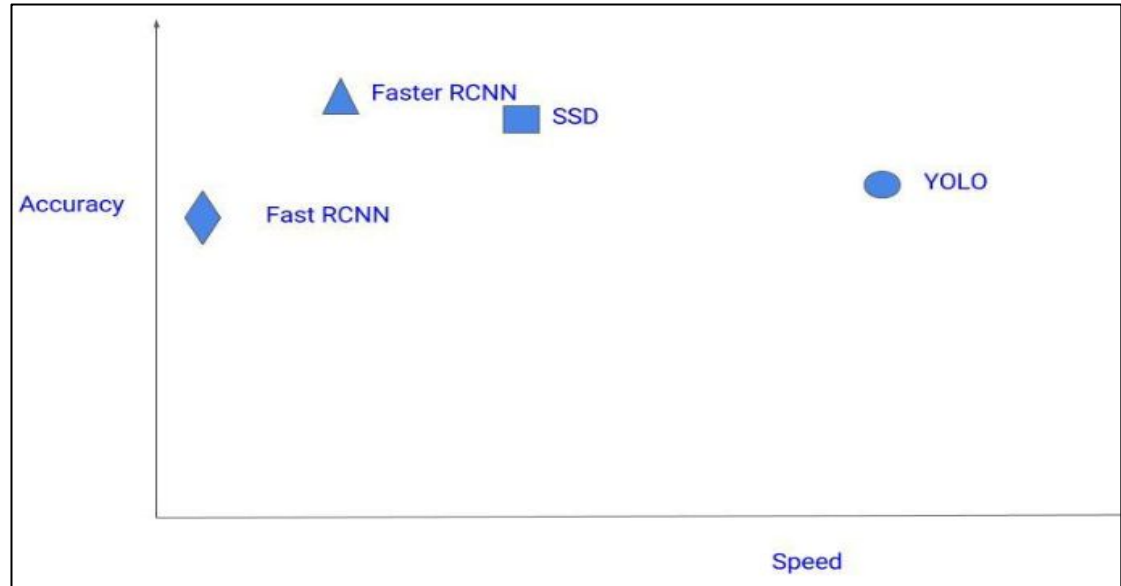
# YOLO (You Only Look Once)



We have used **ImageAI** library as it has pre-trained model of YOLO network on Coco dataset.

## Reason for using YOLO network

Though other methods like Faster RCNN give better accuracy, they use Region Proposal Networks which makes them slow compared to YOLO network.



# Results of Cartoonification algorithm



Input image



After applying Bilateral filter



After applying Adaptive filter



Cartoonified  
Image





Input

Output







Input  
Image

person 95.673

person 98.356



Person  
Detection





Cartoonified  
Image



Input

Output







person 97.831



person 99.868







# Problems Faced

For variety of input images, it would be unrealistic to expect a one-size-fits-all approach to produce consistent results.

Parameters like bilateral kernel size, number of iterations, sigmaSpace, sigmaColor have to be tuned for every image.

Quantifying the goodness of the output image is difficult. (No metric is available)

Evaluation is done by checking how well the output image is.

---



**THANK YOU**

—