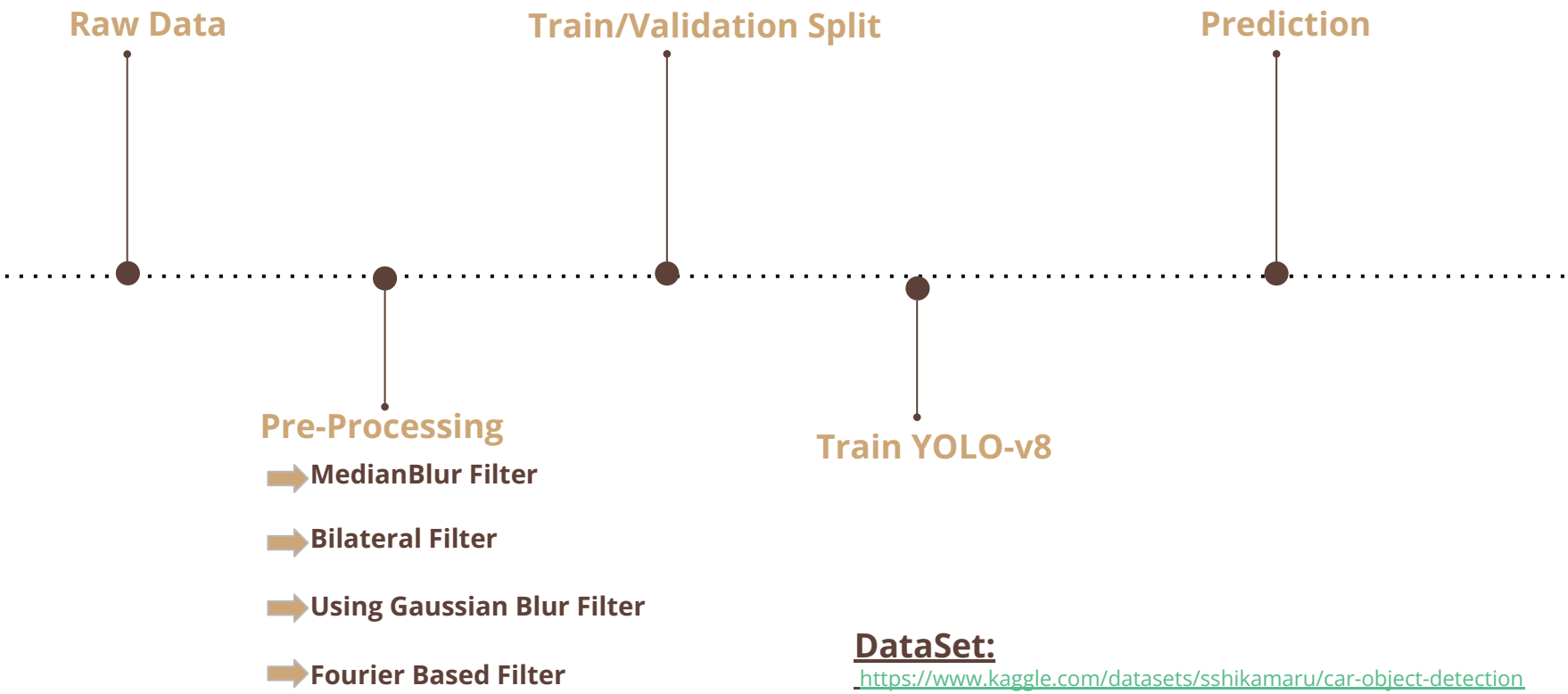
Object detection using image processing techniques and Deep learning

Pipeline:



DataSet:

https://www.kaggle.com/datasets/sshikamaru/car-object-detection

MedianBlur Filter

Original

Original with n=3

Original with n=5

Original with n=7

```
image = cv2.imread("/home/madhu/Car_detection/archive(1)/data/training_images/vid_4_29480.jpg")

fig = plt.figure()
fig.suptitle('"Original"', fontsize=18)
plt.imshow(image)

fig = plt.figure()
fig.suptitle('"Original" with n=3', fontsize=18)
plt.imshow(cv2.medianBlur(image,3))

fig = plt.figure()
fig.suptitle('"Original" with n=5', fontsize=18)
plt.imshow(cv2.medianBlur(image,5))

fig = plt.figure()
fig.suptitle('"Original" with n=7', fontsize=18)
plt.imshow(cv2.medianBlur(image,7))
```

Reference : https://homepages.inf.ed.ac.uk/rbf/HIPR2/median.htm

GaussianBlur Filter

Original

masc_gaus_2d

masc_gaus_1d

```python
def masc_gaus_1d(sigma, n):
    width = n//2
    dx = 1
    x = np.arange(-width, width)
    kernel_1d = np.exp(-(x ** 2) / (2 * sigma ** 2))
    kernel_1d = kernel_1d / (math.sqrt(2 * np.pi) * sigma)

    return kernel_1d

def masc_gaus_2d(sigma, n):
    width = n//2
    dx = 1
    dy = 1
    x = np.arange(-width, width)
    y = np.arange(-width, width)
    x2d, y2d = np.meshgrid(x, y)
    kernel_2d = np.exp(-(x2d ** 2 + y2d ** 2) / (2 * sigma ** 2))
    kernel_2d = kernel_2d / (2 * np.pi * sigma ** 2)

    return kernel_2d
```

```python
image = cv2.imread("/home/madhu/Car_detection/archive(1)/data/training_images/vid_4_29480.jpg", 0)

fig = plt.figure()
fig.suptitle('"Original"', fontsize=18)
plt.imshow(image)

kernel = masc_gaus_2d(sigma = 3, n = 5)
img_convolved = convolve(image, kernel)
fig = plt.figure()
fig.suptitle('masc_gaus_2d', fontsize=18)
plt.imshow(img_convolved,plt.cm.gray)

kernel1D = masc_gaus_1d(5,11)
img_convolved = convolve1d(image, kernel1D)
fig = plt.figure()
fig.suptitle('masc_gaus_1d', fontsize=18)
plt.imshow(img_convolved,plt.cm.gray)
```

Reference :https://datacarpentry.org/image-processing/06-blurring.html

Bilateral Filter

original

n=9 ,sigma r=10, sigma s =100

n=9, sigma r=10, sigma s =100

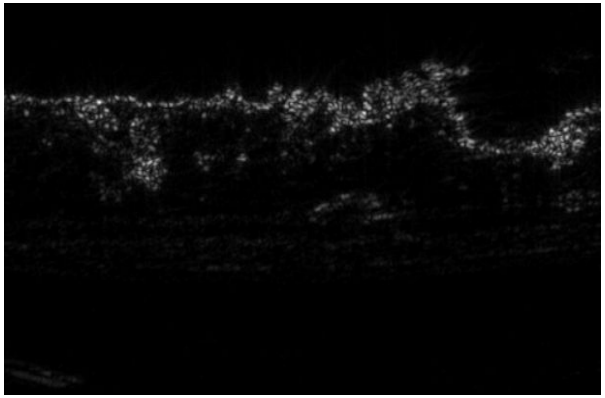
```
image = cv2.imread("/home/madhu/Car_detection/archive(1)/data/training_images/vid_4_29480.jpg")

fig = plt.figure()
fig.suptitle('Original image', fontsize=18)
plt.imshow(image)

fig = plt.figure()
fig.suptitle('n=9 sr=10 ss=100', fontsize=18)
plt.imshow(cv2.bilateralFilter(image,9,10,100))

fig = plt.figure()
fig.suptitle('n=20 sr=10 ss=150', fontsize=18)
plt.imshow(cv2.bilateralFilter(image,20,10,150))
```

Reference: https://www.geeksforgeeks.org/python-bilateral-filtering/

Fourier Based Filter

Original

filtered_500



filtered_100

```python
def kernel_filter(radius):
    filter_kernel = np.zeros((rows, cols), np.float32)
    cv2.circle(filter_kernel, (crow, ccol), radius, 1, -1)
    return filter_kernel
```

```python
def apply_fourier_filter(image, filter_kernel):

    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    fft_image = fft2(gray_image)
    fft_image_shifted = fftshift(fft_image)
    filtered_fft = fft_image_shifted * filter_kernel
    filtered_image = np.abs(ifft2(filtered_fft))
    filtered_image = np.uint8(filtered_image)

    return filtered_image
```

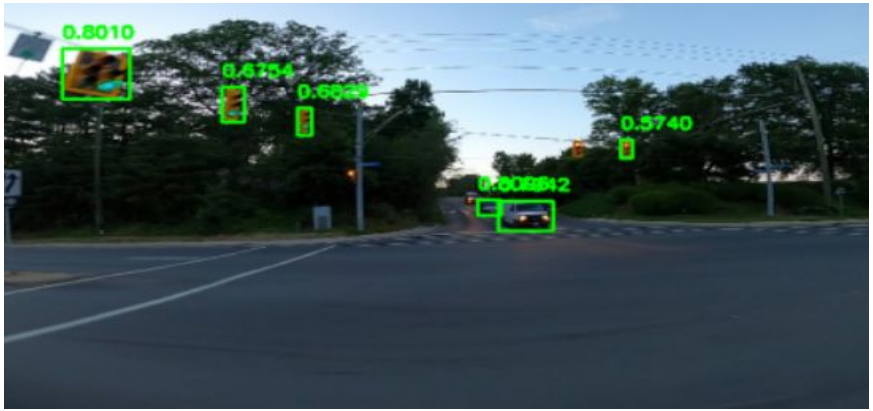
Object Detection using YOLO-v8 Metrics

➡ Why YOLO-v8 stands out ??

Metrics :

```
Ultralytics YOLOv8.1.29 🚀 Python-3.10.13 torch-2.1.2 CUDA:0 (Tesla T4, 15102MiB)
Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|██████████| 13/13 [00:02<00:00,  4.47it/s]
                   all        401        263      0.988      0.992      0.995       0.73
Speed: 0.1ms preprocess, 2.5ms inference, 0.0ms loss, 0.8ms postprocess per image
```

Prediction Example :



Reference : https://docs.ultralytics.com/