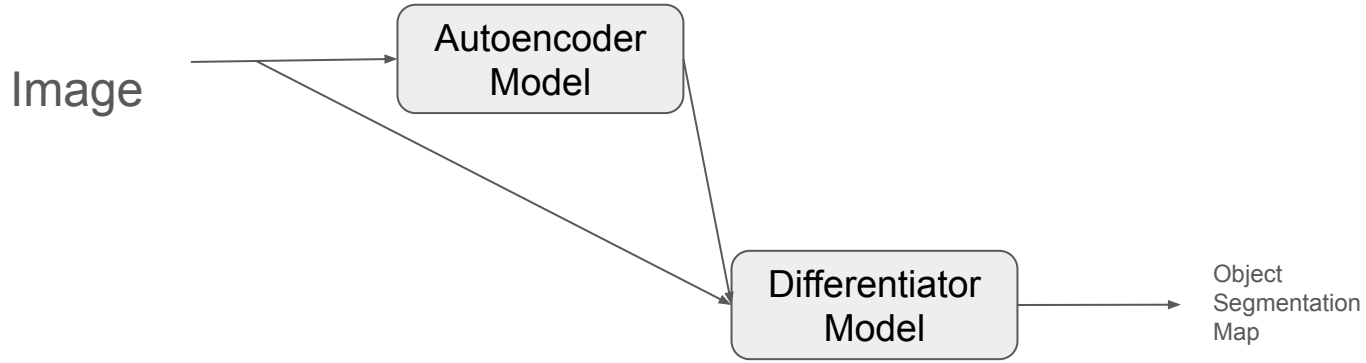
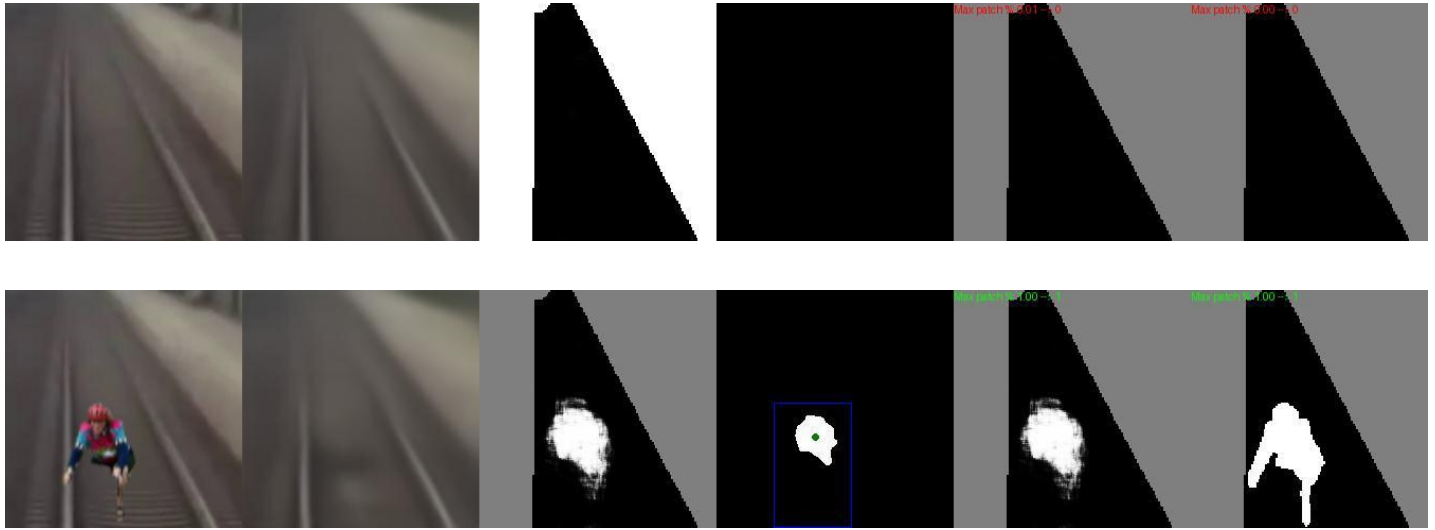


Previous Approach: The original work from IROS journal only created segmentation map for object in image containing rail

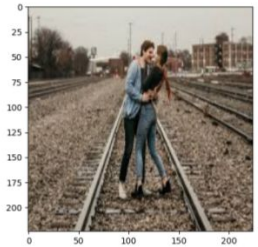
Ref: https://github.com/ethz-asl/railway-anomaly-detection/tree/master

Results

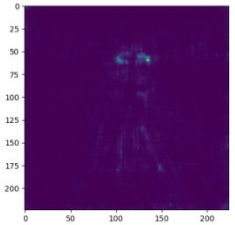
Sample Outputs from Repository (on validation set)



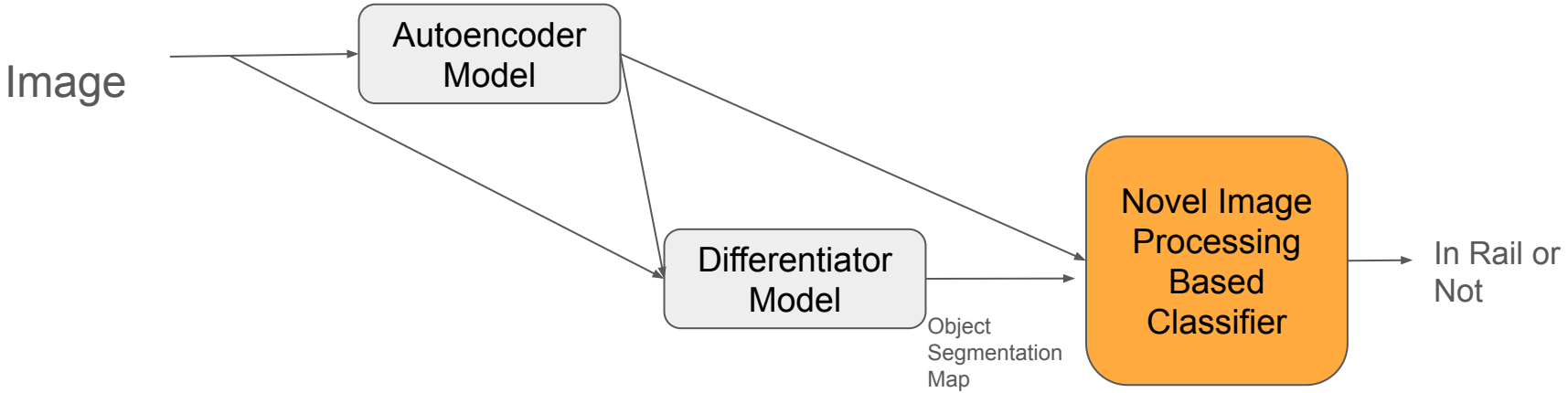
Sample Output on Custom Image



New Target Image



Segmentation Map



New Approach: I create a useful modification, where I use the results to definitely classify whether an object is within rail or not

Ref: https://github.com/antibloch/railway-anomaly-detection

Novel Image Processing Based Classifier

Segmentation Map

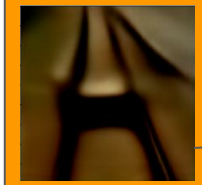
Average filter convolution

Min-Max Normalization

Connected Component Labelling

Finding Maximum area component

Object Image



Autoencoder Output

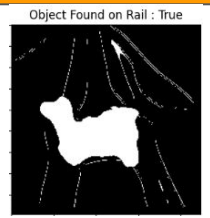
Otsu Binarization

Sato Filter

Otsu Binarization

Rail Image

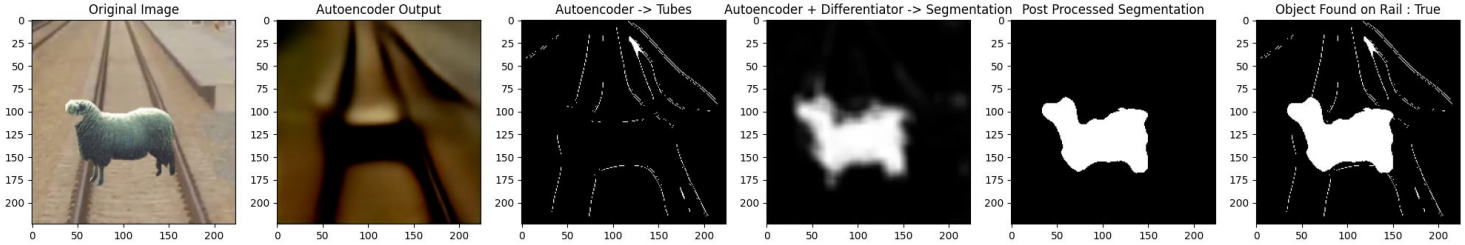
```
Compoent_epochs =[]
For each connected component in Rail_image:
    For E in dilation_epochs:
        Dilate the component pixel by E epochs with a dilation kernel.
        If intersection with object image:
            Component_epochs.append(E)
            Break


If Standard deviation of Component_epochs< Threshold (ideal  is 1.0):
    Return "Object lies on rail"
Else:
    Return "Object not on rail"
```



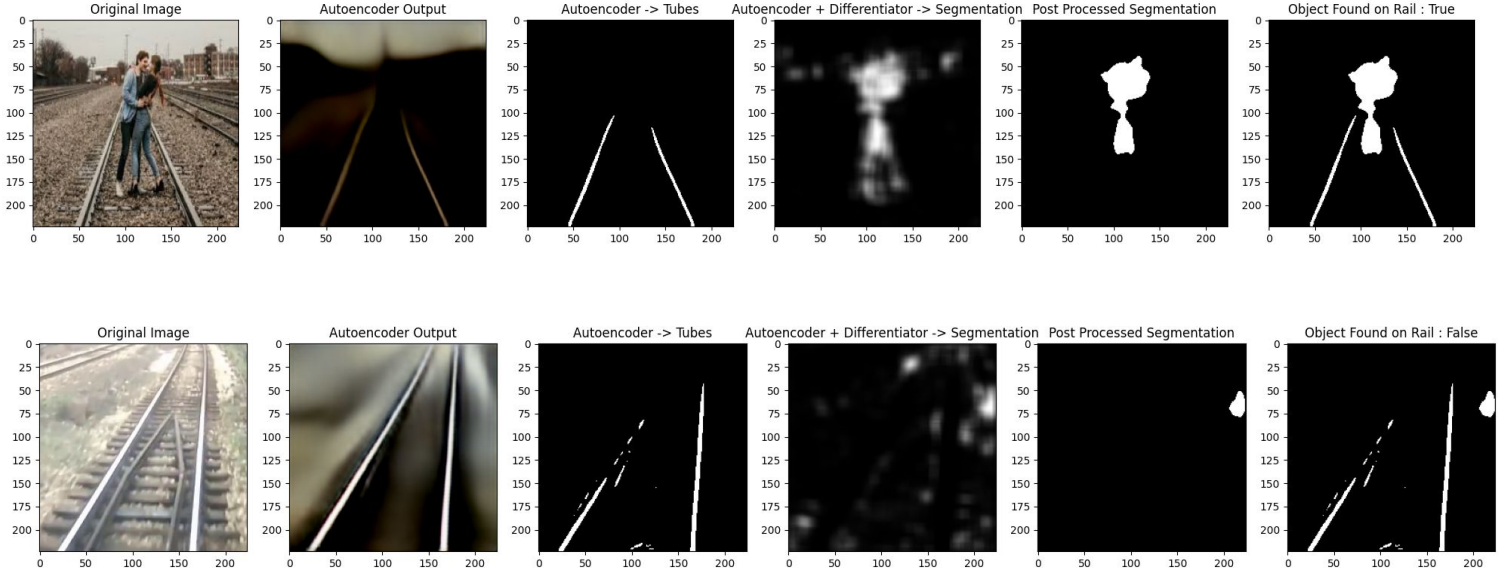
In Rail or Not

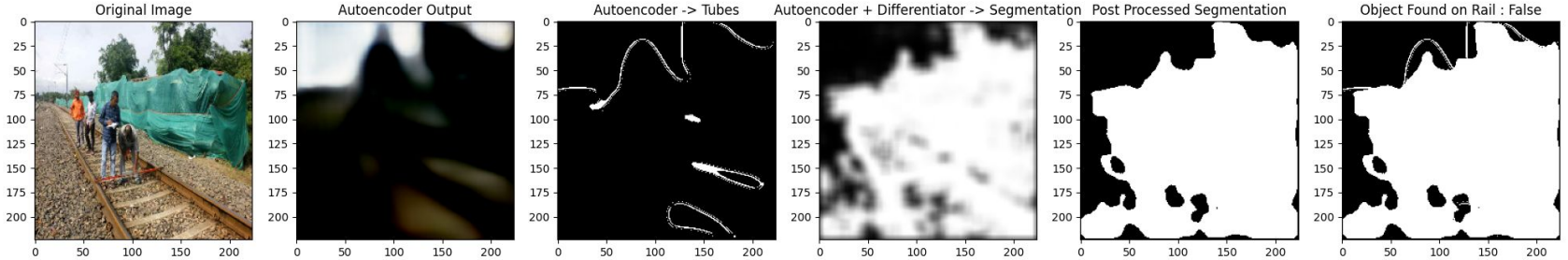
Results

Performance on Training Set Sample



Performance on Testing Set Sample





As the sample becomes out of distribution, the autoencoder's output fails, and consequently performance of pipeline fails. But for railway project this would not be case, as we will have dataset associated with target railway