Object Contour Detection with a Fully Convolutional Encoder- Decoder Network [Paper ID : 65]

BY : ISHITA AGGARWAL – 2018A3PS0343P
SHALU SINHA – 2018A3PS0432P
PARINISTHA DEV DAS – 2018A1PS0200P

Author Affiliation

- The paper was presented at CVPR 2016. The authors are as follows:

  - Jimei Yang - Adobe Research
  - Brian Price - Adobe Research
  - Scott Cohen - Adobe Research
  - Honglak Lee - University of Michigan, Ann Harbour
  - Ming-Hsuan Yang - University of California, Merced

Paper Overview: Aim

- The paper aims at developing a fully convolutional encoder decoder network for object contour detection.
- The trained model generalizes well to unseen object classes from the same super-categories, yielding significantly higher precision in object contour detection than existing low level edge detection models.
- The model matches the state of the art precision and recall.

Paper Overview: Methodology

- The encoder is constructed using **VGG-16 network** upto the **fc-6 layer**.
- **A light weight decoder** is designed to allow unpooling from its corresponding maxpooling layers. All the convolution layers except deconv6 use **5×5 kernels** all except the one next to the output label are followed by **ReLU activation function**.
- **DenseCRF** is used to label thin unlabeled (or uncertain) area between occluded objects in the PASCAL VOC dataset.
- The encoder is initialized **with pre-trained VGG-16 net** and the decoder with random values. During training, **the encoder parameters are fixed** and only the **decoder parameters are optimized** to allow integration with other decoders for joint training.

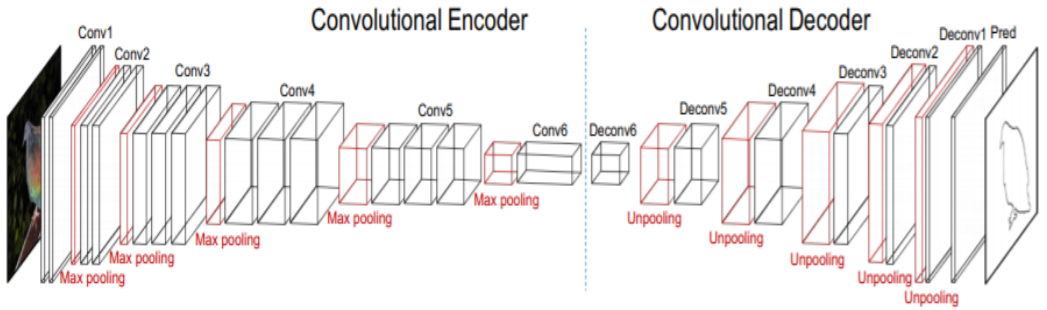


Fig1. Network architecture of fully convolutional encoder decoder

Paper Overview: Outcome

- CEDN obtains good results on those classes that share common super-categories with PASCAL classes, such as vehicle, animal and furniture.
- CEDN fails to detect the objects labeled as "background" in the PASCAL VOC training set, such as food and appliance.
- CEDN works well on unseen classes that are not prevalent in the PASCAL VOC training set, such as sports.

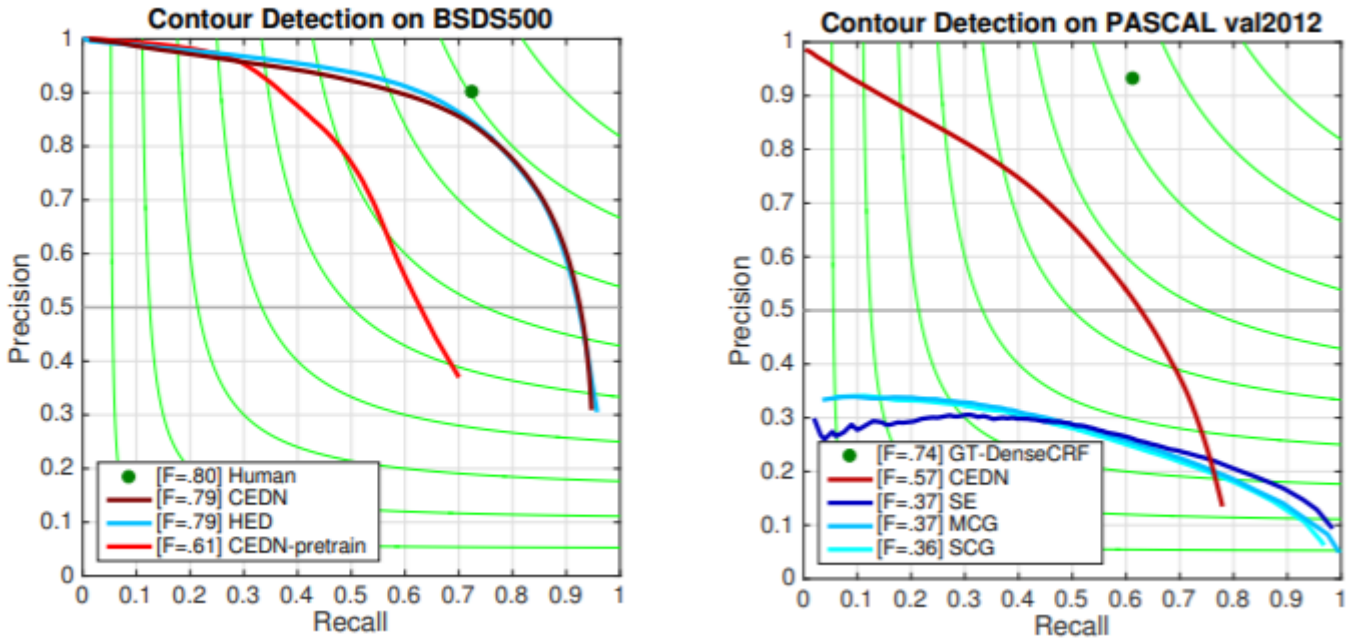


Fig 2(a) P-R curve for BSDS500 dataset (b) P-R curve for PASCAL VOC 2012 dataset

Background Concepts

- **Edge detection:**
  - **Edges** are abrupt pixels at which the luminance, color or stereo change sharply and is mostly used to denote image points where intensity differences between pixels are significant.
  - **Edge detection** mainly focuses on the changes in brightness, color, and texture, and aims to extract visually salient edges from natural images. It is usually considered as a low-level technique.
- **Contour detection:**
  - **Contours** are referred to as the boundary pixels of meaningful objects. The term 'contour' is used to denote object boundary.
  - **Object level contour detection** does edge detection and organizes related edges together to detect the border of objects (object-level contours) in images. Thus, in many cases contour detection is performed by processing the detected edges further.

Fig 3 (1st column) Input image
(2nd column) Edge detection
(3rd column) Contour Detection

Background Concepts

- **Encoder Decoder Networks:**
  - An encoder is a network (FC, CNN, RNN, etc) that takes the input, and outputs a feature map/vector/tensor. These feature vectors hold the information and features that represent the input.
  - The decoder is also a neural network (usually the same network structure as encoder but in the opposite orientation) that takes the feature vector from the encoder, and gives the best closest match to the actual input or intended output.
  - The encoders are trained with the decoders. There are no labels (hence unsupervised). The loss function is based on computing the delta between the actual and reconstructed input. The optimizer will try to train both encoder and decoder to lower this reconstruction loss.
  - Once trained, the encoder will give a feature vector for input that can be used by the decoder to construct the input with the features that matter the most to make the reconstructed input recognizable as the actual input.

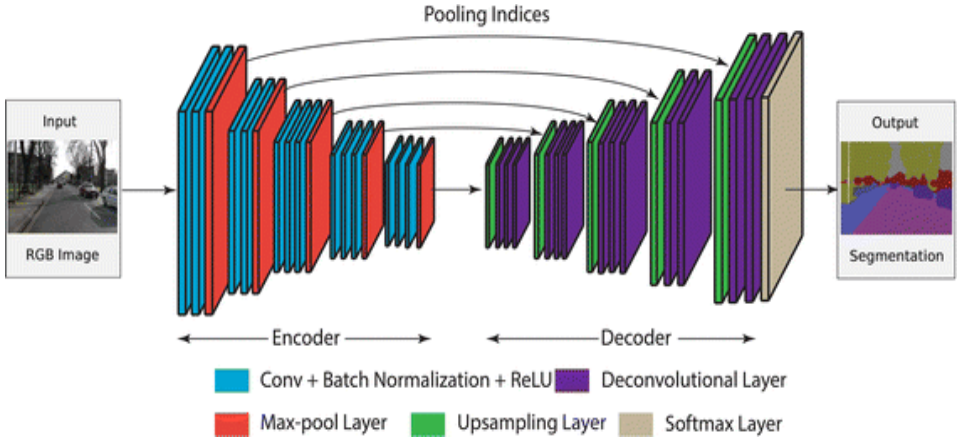


Fig 4 Encoder - Decoder

Background concepts

- **Image Labelling :**
  - Image Labelling is identifying the contents of an image (e.g. : trees, people, animals, etc) and annotating it for the picture.
  - Image contour detection, in this paper, is formulated as a binary image labelling problem. “1” indicates contour and “0“ indicates “non-contour”.
- **Precision :** The number of correct results divided by the number of all returned results.
- **Recall :** The number of correct results divided by the number of results that should have been returned.

Dataset Details : PASCAL VOC 2012

- The PASCAL VOC dataset is a widely-accepted benchmark with high-quality annotation for object segmentation is used.
- VOC 2012 release includes 11540 images from 20 classes covering a majority of common objects from categories such as “person”, “vehicle”, “animal” and “household”, where 1464 and 1449 images are annotated with object instance contours for training and validation respectively.
- Together there are 10582 images for training and 1449 images for validation

Dataset Details: PASCAL VOC2012

- The original PASCAL VOC annotations leave a thin unlabeled (or uncertain) area between occluded objects which is rectified by using denseCRF network which fills the uncertain area with neighboring instance labels so that refined contours at the labeling boundaries are obtained.

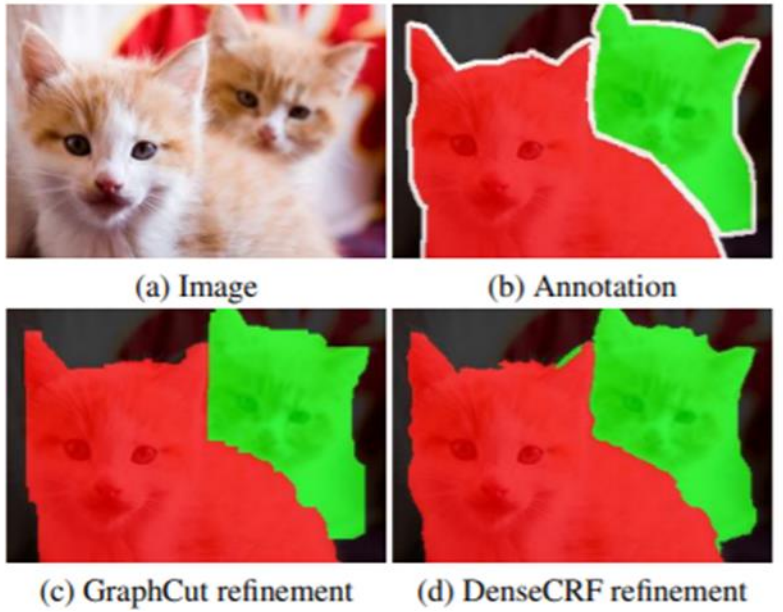


Fig 5 . Contour refinement. The polygon based annotations (a) cannot be directly used for training due to its inaccurate boundaries (thin white area reflects unlabeled pixels between objects). We align them to image boundaries by re-labeling the uncertain areas with dense CRF (d), compared to Graph Cut (c)

Dataset Details : BSDS500

- The BSDS500 dataset is a standard benchmark for contour detection s designed for evaluating natural edge detection that includes not only object contours but also object interior boundaries and background boundaries .
- It includes 500 natural images with carefully annotated boundaries collected from multiple users. The dataset is divided into three parts: 200 for training, 100 for validation and the rest 200 for test.
- This dataset is used to fine-tune the CEDN model on the 200 training images from BSDS500 with a small learning rate (10−5) for 100 epochs.

Implementation: Data Pre-processing

- The PASCAL VOC2012 dataset is primarily used for image segmentation, we first needed to convert labels given for the task of image segmentation to the task of contour detection as mask. These labels are placed in **SegmentationObjectFilledDenseCRF.**

- **Conversion to contour labels** : If there is variation among pixel values in 3x3 matrix depiction then there lies contour in that part of the image. As the pixel values are changing the required part is converted to 1 rest to 0.

- We generated a new set of images using this algorithm (extract_contours.py) and placed them in **improved_contours.**

Implementation : Data Augmentation

- 3 kinds of data augmentation were performed on the test set :
  - **Random Cropping**: 4 Images of shape : (224,224,3) were cropped from a single image.
  - **Color Jittering**: We used color jitter to augment the data through variation in color,contrast, brightness of image which would help the network learn through different data.
  - **Image Flipping**: The 4 images which were randomly cropped were flipped horizontally, creating a batch of 8 images from a single image.
  - A class ‘dataloader’ was defined in train.py for these augmentations.

Implementation : Training

- We have processed data one at a time in mini batches of size (8,224,224,3).
- We have used multiprocessing because of slow speed of Google Drive Stream in Google Colab. Multiprocessing runs several data loaders in parallel, they take up the data from Google Drive and then put it into a thread/process safe Queue , the data from the queue is popped one by one and fed to the trainer object that uses it to train the model.
- Training was done for 30 epochs on the training set of PASCAL VOC 2012.
- The file train.py contains all the relevant code for training and multiprocessing

Implementation: Validation and Testing

Validation on PASCAL VOC2012

- The models after each epoch were saved in the 'models' folder.
- The results on the validation set were generated using objects from dataloader class and stored in the '/VOC2012/results' folder.
- The evaluation for AP, P-R Curve and Maximum F Score was calculated using class Evaluator in the file eval.py.

Testing on BSDS500 dataset

- For testing the pre tuned network, first some preprocessing was done on the ground truth labels of BSDS500. These were converted from .mat to .png files
- After this an object of the class **bsdsdataloader** was instantiated with the appropriate path dictionary and model was instantiated to model30.pth (Model saved after 30th epoch).
- The results generated were stored in the '/BSDS500/results' folder and the evaluation was done similar to PASCAL VOC on the test set of BSDS500.

Fine – tuning the network

- **Optimiser Adam** was used with a **learning rate of 10e-5**.
- The network was trained on the trainval set of BSDS500 and then new results were generated and evaluated.

Results: PASCAL VOC2012

Fig6 P-R curve for PASCAL VOC2012 dataset

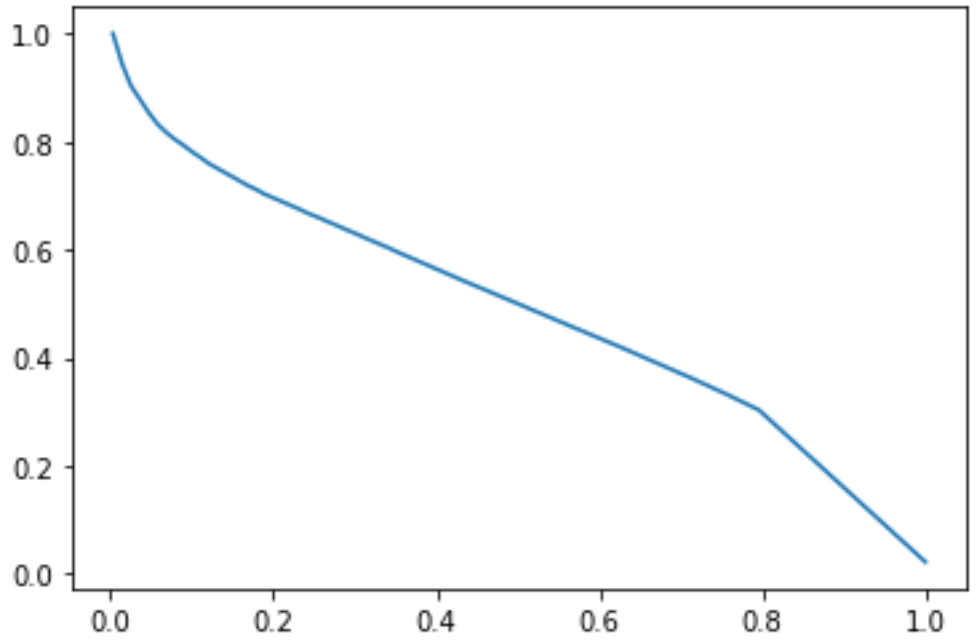
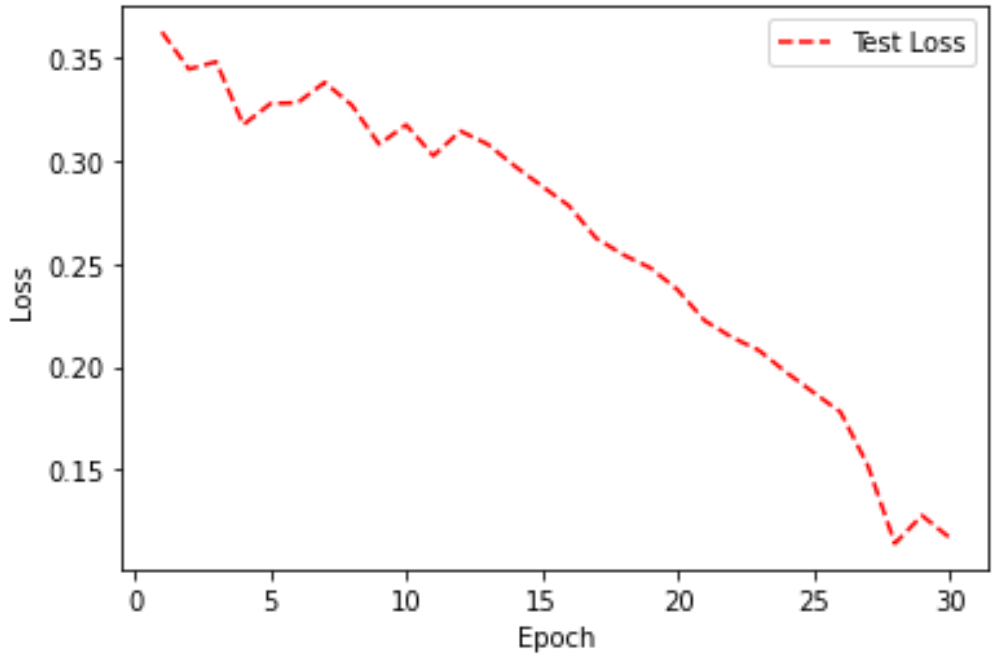
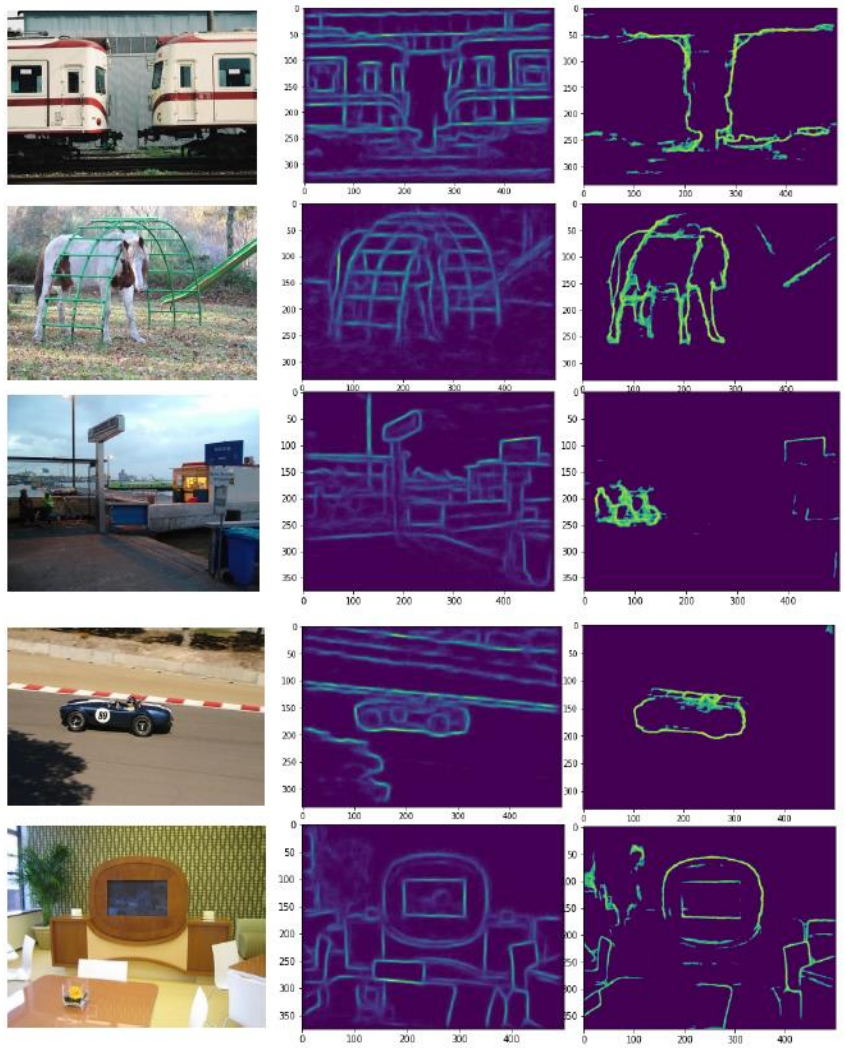


Fig 7 Loss vs epoch curve for PASCAL VOC2012 dataset



AP value = 0.002760524499654934
F-score = 0.5



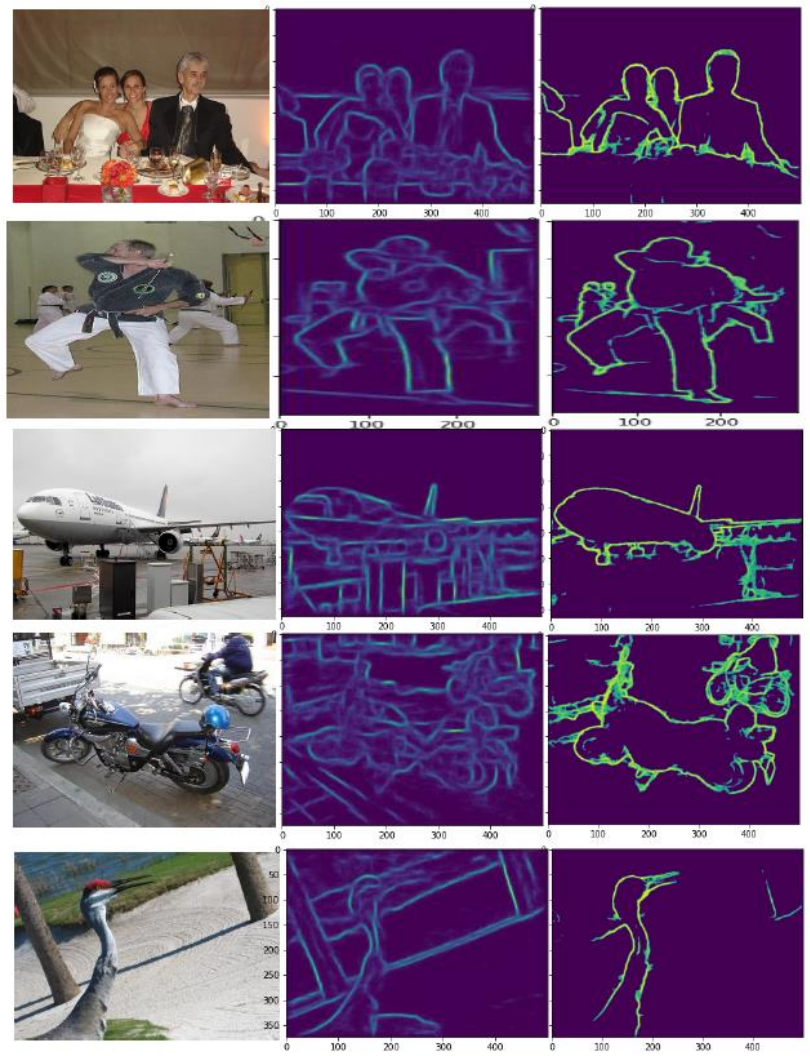
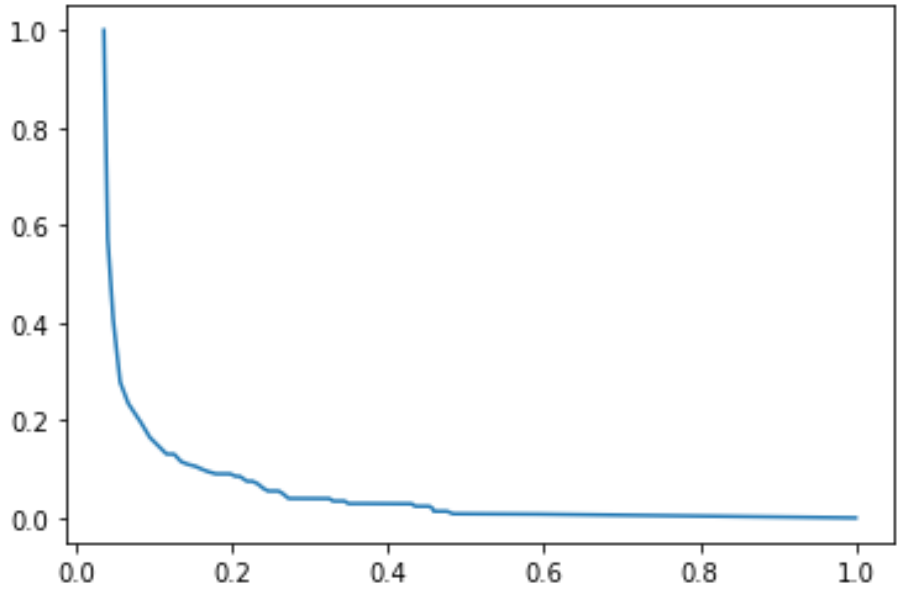


Fig 8 (a) 5 worst images obtained for PASCAL VOC2012 dataset (b) 5 best images obtained PASCAL VOC2012 dataset

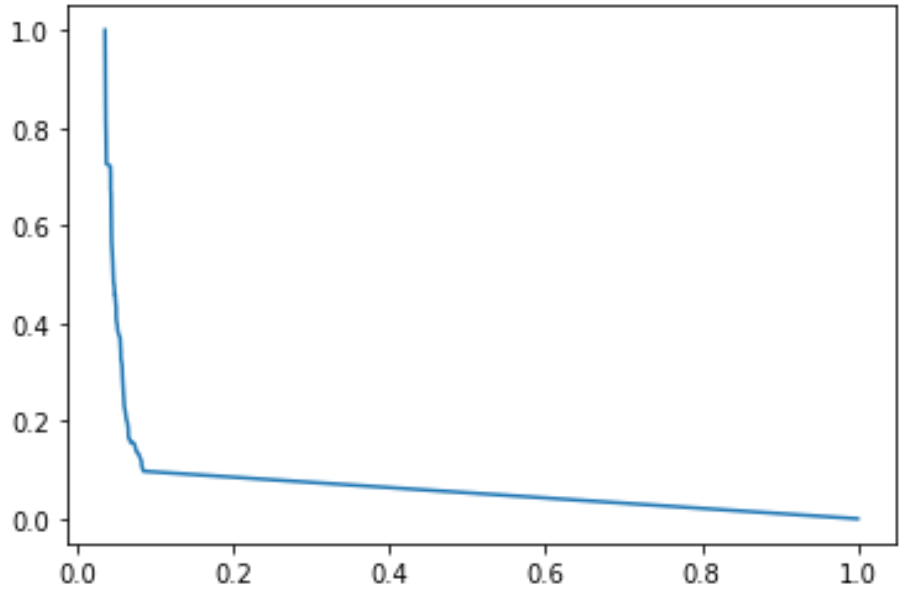
Results: BSDS500 dataset

Fig 9 P-R curve before training the model on BSDS500 dataset



AP = 0.035
F-score = 0.1279336295418785 at
threshold = 0.003222258025089167 7

Fig 10 P-R curve for the fine-tuned model i.e. after training on BSDS500 dataset



AP = 0.03500000000000003 and
F-score = 0.09846697382732532 at
threshold= 0.0017444638273758596

Figure 11. Results obtained on BSDS500 test set. In each row from left to right we present (a) input image, (c) contour detection with pretrained CEDN and (d) contour detection with fine-tuned CEDN

Challenges Faced

- Training and testing the model on a large dataset.
- Implementation of DenseCRF model in order to label the unlabelled area among occluded objects.
- Understanding various new concepts like semantic segmentation , edge detection etc..
- Evaluating BSDS500 and further fine- tuning it as the ground truth images were .mat files.

Future Scope

- The current model doesn't work well when tested on MS COCO val2014 dataset due to presence of noisy annotations (like trying to identify the object categories that are identified as background in PASCAL VOC2012 dataset).
- Hence the future prospects lie in developing large scale semi-supervised learning methods for training the object contour detector on MS COCO with noisy annotations, and applying the generated proposals for object detection and instance segmentation.

Learning outcomes

- Handling large datasets by analysing in batches rather analysing the whole data at once.
- Improving results of the model and training data through data augmentation.
- Knowledge of encoder – decoder networks.
- Knowledge of pre- trained networks and transfer learning.
- Way of reading a research paper and implementing it.

THANK YOU!