Object Contour Detection with fully convolutional encoder-decoder network

Summary

◦Project Repository is - https://github.com/captanlevi/Contour-Detection-Pytorch

◦Objective of the paper was to create a deep learning model for contour detection with a fully convolutional encoder decoder network which is different from previous low level edge detection .

◦In the repository a conda environment file have been provided to recreate the environment for the experiment.

◦The Model is Trained and validated on PASCALVOC-2012 Dataset and give pretty decent perfomance with a high precision and recall in contour detection.

◦By combining with widely used segmentation algorithms paper claims that their model can generate high quality segmented object proposals .

Data Preprocessing

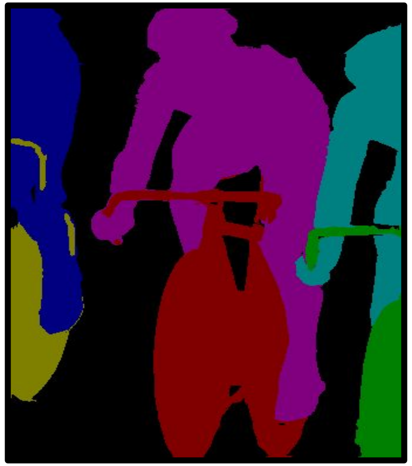
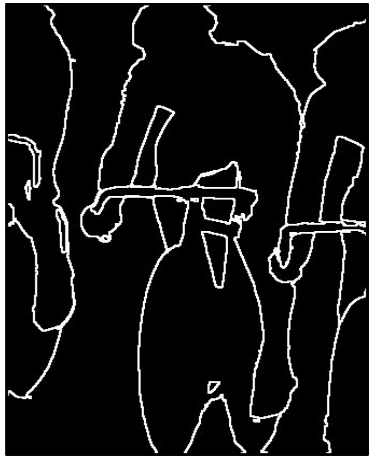
◦Dataset we used is PASCALVOC which have 10,383 images and is primarily used for the task of image segmentation.

◦So one of the main preprocessing tasks were to convert labels given for the task of image segmentation to the task of contour detection as mask .

◦So for that we implemented widely used algo : Morphin Mask- which converts the label image used for the segmentation tasks to the label mask used for contour detection

◦Summarized way of how this algorithm works is that you take 3*3 filter and put it across the images as the image labels currently are for the task of image segmentation.

◦Now for this to convert into contour labels we can observe that if there is variation among pixel values in 3*3 matrix depicts then there lies contour in that part of the image and as the pixel values are changing that part is converted to 1 rest to 0,using this algorithm as backbone we have created this file(`extract_contours.py`)to convert and create a new set of images.

◦Main problem faced during such tasks is that images from dataset are of varying sizes but to feed in the network we need a batch of given specific size so to do that some of the data augumentation and preprocessing tasks were being done.

Data augementation

◦So some of the data transformation tasks performed were:-.

1. **Random Cropping**:- 4 Images of shape(224,224,3) were being cropped from a single image of varying shape.(please explain this a little bit more-code oriented)
2. **Color jittering**:- Pytorch has an amazing facility to manipulate image data, one of which was to use color jitter to augument the data through variation in color,contrast, brightness of image which will help network to learn different data better.
3. **Image Flipping**:- The 4 image which were randomly cropped were being flipped horizontally which led to increase in dataset and thus we created a batch of 8 images from a single image.

Data Handling

◦Other Problem we faced during taking data was the enormity of size of the dataset as to load such a large dataset it definitely requires a lot of ram to take such a huge dataset at a one go which on any cloud service will take a lot of time to load and stream such huge data.

:- so solution for this problem was to take data one at time (of minibatch size –(8,224,224,3) so data was taken in a way so that only a small amount of ram space would be used at time and even high speed gpu performance can be maintained.

MultiProcessing

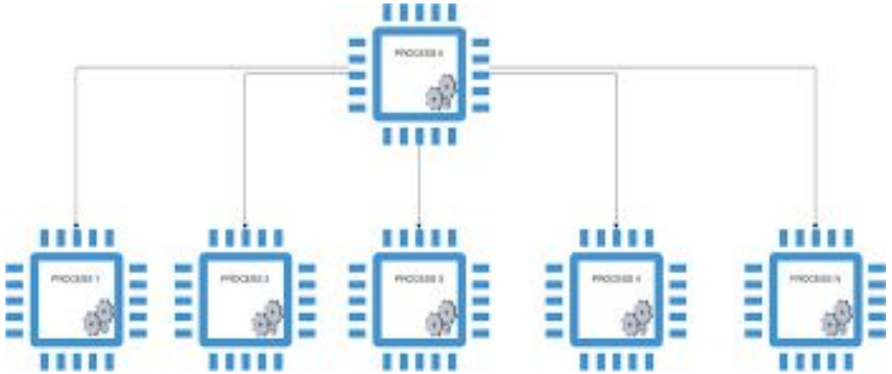
One of the biggest problems faced in this work is the slow speed of data streaming from google drive. As training was done on google colab, many disc seeks were made on google drive. If this goes on in a blocking way the speed benefit offered by GPUs will not be utilized to the maximium.

To counter this I have used multiprocessing to run several dataloaders in parallel, they take up the data from google drive(or your PC) 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.

So tldr...

1. Many workers (each worker is slow) fills up the queue from one side.
2. The model (that is fast) removes the data placed in the queue from the other side.

`Refer Train.py`

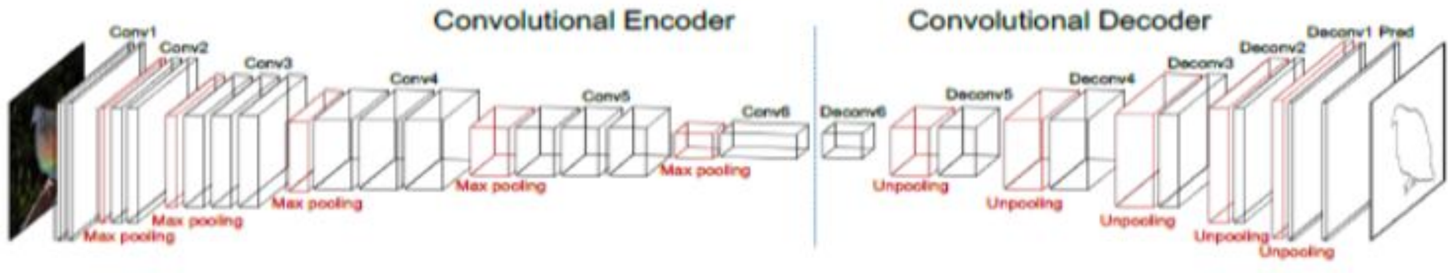


Encoder-Decoder Model

◦Our Model is a prime example of end to end deep learning , the network in which consists of mainly two parts Encoder and Decoder.

◦Encoder uses a pretrained VGG-16 model which in itself is a highly recognized model and used for the perspective of encoding features of the images

◦Decoder network uses mostly large subnetwork of deconvolutional layers to retain the shapes and sizes being diminished by convolutional layer.



Model Attributes

◦Main aspect of this model is that it is **fully convolutional model** not even a single linear layer has been used by the model.

◦As you can see the image in the previous slide the Encoder part of the network is normal VGG network but the tricky part is Decoder network.

•To make the network fully convolutional **no Linear layer** have been added and in decoder part of network ,deconvolutional layers/ Transposed Convolutional layers have been used.

•A total of six deconvolutional layers have been used in which the kernel size of first deconv layer is 1 size of Kernels of remaining of the layers is 5.This shows the **simplicity** of the model performing such large scale tasks as just how even though not increasing much complexity in model and just keeping it fully convolutional the model is performing so beautifully.

•Activation Function used everywhere is **Relu,** except the last layer of the whole model which uses **sigmoid** activation function.

Optimizers & Loss Function

◦Optimizer used for the model was **Adam with learning rate of e-4** which turns out to be best for the model

◦As the model labels have large amount of Black pixels (0) in comparision to the white(1-normalized) pixels this creates an imbalance in the data so to overcome this problem a specialized binary-cross entropy loss was being used in which the **loss for not detecting white pixel properly is ten times the loss for black pixel.**

◦Network was trained over for **30 epochs** with all the training images being processed each epoch.

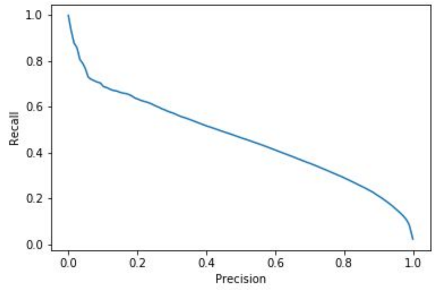
◦Note that training patch was fixed for 224*224 and the learnt parameters can be used for images of arbitrary size because of its fully convolutional nature.

Results

◦Model after getting trained was tested on validation dataset

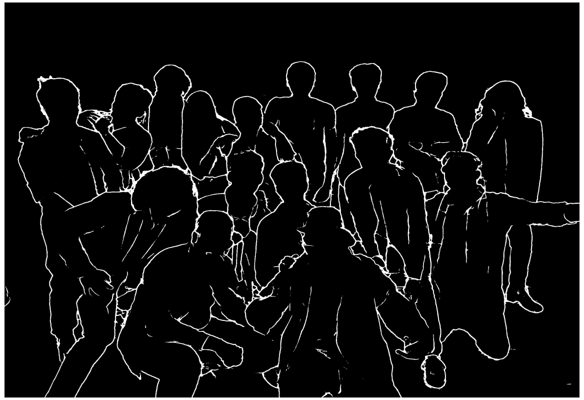
•Measures to test are primarily F1 score calculated for different thresholds values(used for converting the output label of the model from decimal values(as calculated by sigmoid layer)

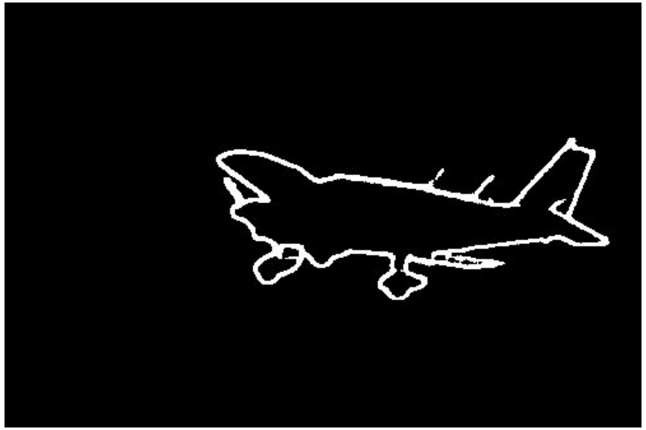
•Best value of the threshold was being chosen with the help of precision- recall curve and F1 score which 0.5 in comparison to original model with F1 score 0.57



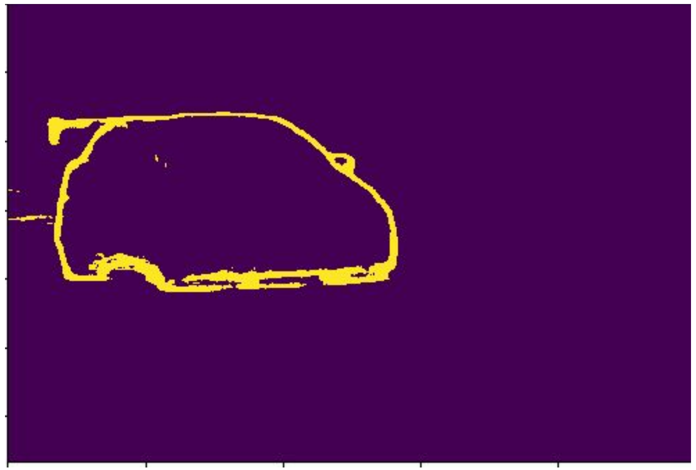
Predictions

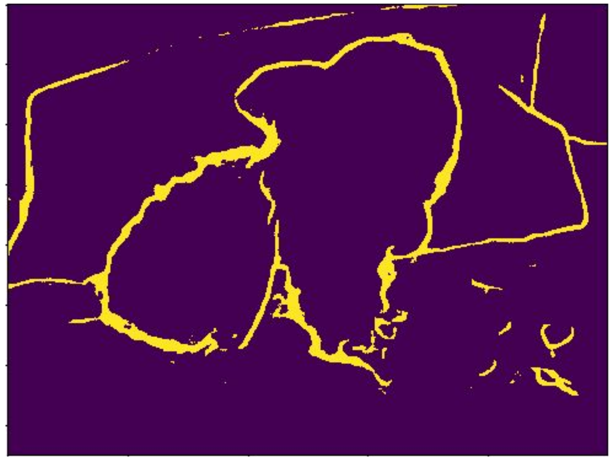
Here are some of the predictions by our model:-





Predictions





Quantitative Results

Taking a closer look at the results, we find that our CEDN (Convolutional Encoder Decoder Network) algorithm can still perform well on known objects but less effectively on certain unknown object classes, such as food . It is likely because those novelclasses, although seen in our training set (PASCAL VOC),are actually annotated as background. For example, there is a “dining table” class but no “food” class in the PASCAL VOC dataset.

- 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 “applicance”.

Things we did’nt do

Our Model perform quite impressive but performed an F1 score of 0.50 but it did’nt reach the 0.57 score of actual model because of few reasons:-

- We have not trained the model additionally on BSDS500 Dataset of total 500 images. That is why our boundaries are being suppresed and not resurfaced.
- We have not validated the model on MS COCO val 2014 dataset.
- We have not created object propsal from our contour because not much specifications were given on use of parameters in the research paper but definitely wanna try such an intresting thing later!
- Dropout layers were not being used in the model.

Takeaways

This project was really important to get to know about how different problems can be solved by deeplearning.

We learnt a great deal of things:-

- How to handle large dataset and not to take all the data at once
- How to improve the results of model and training data through data augementation.
- How to make dataset provided acquainted to your needs.
- Learnt about Encoder-Decoder Networks
- Learnt about how to use pretrained networks and Transfer learning.

THANK YOU!!!!