ASSIGNMENT2, ADVANCED IMAGE PROCESSING

Rishubh Parihar

CDS, Indian Institue Of Science, Bengaluru

1. NORMALIZED CUTS

We implemented the one-step Normalized Cut algorithm, which generated two segments of the given input image. For implementation, we resized the images to make the smaller size equal to 50 pixels and other dimension was scaled accordingly. We used the σ_x value of 10 and σ_i value of 100 for all our experiments. Additionally, We used a radius value of 12 pixels in our implementation. We observed that the n-cut results are highly sensitive to these parameters and found these values to work best across images. We started the algorithm by first computing the adjacency weight matrix W and the degree matrix D. We then solve the following standard form of eigenvector problem:

$$A * x = \lambda * x \tag{1}$$

$$where, A = D^{-1/2}(W - D)D^{-1/2} \tag{2}$$

We used the LANCOZ solver provided by scikit learn to obtain the top-k eigenvectors corresponding to the least eigenvalues. Once we have obtained the eigenvector x, we use it to segment the image by creating soft and hard segmentation masks. We normalized the x vector in the range $(0, 1)$ to generate a soft segmentation mask, where values near 0 represent one class and near 1 represent another class. Further, we threshold the unnormalized x vector at value 0 to obtain hard segments. We observed that the results are very sensitive to the selected threshold; hence we displayed both the soft segmentation mask and the hard segmentation mask for our experiments. We performed segmentation on synthetic images Fig. 1 and on real images Fig. 5. Note that we resized the images to a small resolution of $(50, h)$ before processing due to computational limitations. We ignored the 0^{th} eigenvector in our experiments as it has given trivial results.

Fig. 1: We present results for the N-Cut algorithm on two synthetic images. We show the cuts generated by the top three eigenvectors estimated by the algorithm. For both the images, we can observe that the algorithm does not well separate the segments by the first cut (second least eigenvalue). However, observing the soft segments provides insights that the algorithm can differentiate between the shape and the background image. Specifically, the first eigenvector for the circle shape is able to separate the circle from its neighboring background pixels. The second and third eigenvectors are also able to distinguish the circle shape from the background. For the rectangle shape, the third eigenvector resulted in the best cut, as can be seen from the soft segments. We can note that, looking at just the hard segments can mislead the conclusions (as in row 1) and hence visualization of soft segments is required.

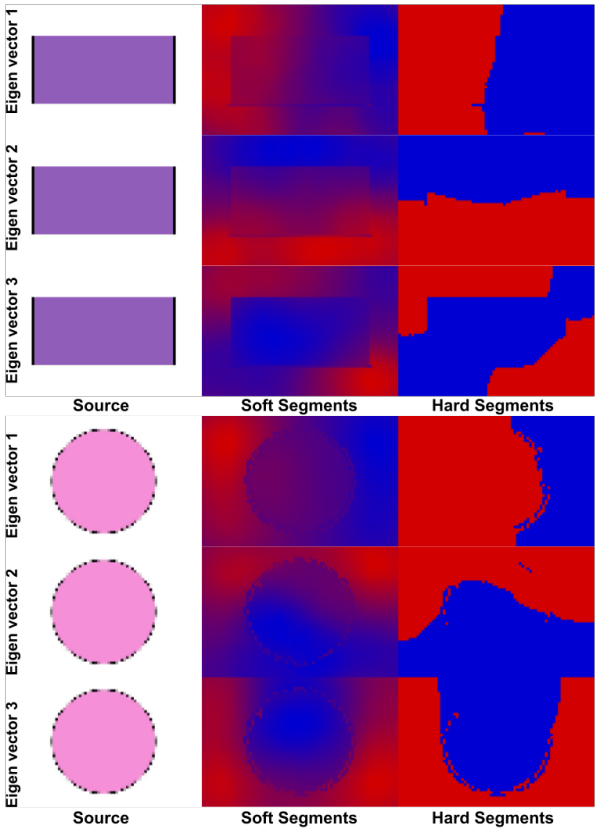

Fig. 1. Segmentation results using N-Cut algorithm on two synthetic images. Here we show cuts generated by the top (least eigen values) three eigen vectors.

Fig. 2: Here we present our results on a set of real images with different levels of complexity. (Note: Fig. 2 is shown in the last page of the report.) We computed the first four eigenvectors to estimate the segmentation masks. For the image with sheep and people, the first eigenvector can find the face and body structure as seen in soft segments. Additionally,

the fourth and second eigenvector can properly separate her body's backside and ears for the sheep image. In the fourth image, the girl's face is segmented in second, third, and fourth eigenvectors. For the airplane image, we can see that some separation of the front of the plane from the background is present in the first and the third eigenvectors. Eigenvectors one and four can separate the tail and back body of the aircraft. Finally, the boat image is the most challenging of all the images. However, we can observe in the first row that the N-Cut can localize the shadow of the boat and the boat from the background. Also, the background trees are segmented in the third and the fourth eigenvectors. These results suggest that a simple N-cuts algorithm in the pixel space has a good potential to segment real-world images. One can extend this method and merge this with the features from a deep network and perform N-cuts in the feature space. This could result in cleaner segments and incorporate the rich image priors in the process.

2. FCN FOR SEGMENTATION

In this section, we will discuss our implementation of the Fully Convolutional Network for semantic segmentation. We used two networks: a pre-trained FCN (on COCO) and another custom FCN network built using the Resnet-18 backbone. We used the Pascal-VOC-2012 dataset for training and evaluation, which has 1464 training images and 1449 validation images. There are 21 semantic classes in this dataset, ranging from train to the couch. The details of the models are given below:

1) **FCN network:** We used a pre-trained FCN network trained on the Coco semantic segmentation dataset. Coco and the Pascal dataset has the same set of classes; hence the pre-trained model can be directly used without alteration. This network has a backbone of Resnet-50 and is taken from the official implementation of Pytorch FCN models. We froze the network's weights to perform the inference on a set of input images.

2) **Custom FCN:** We trained a custom FCN model with a pre-trained backbone. Specifically, we use a Resnet-18 model pre-trained on the ImageNet dataset for classification tasks and removed the last pooling and the fully connected layers. The chopped model maps the input image into $16 * 16 * 512$ volume. We added one convolutional layer of $1 * 1$ convolutions. This layer has $n = 21$ filters, which is the number of semantic classes. Finally, we added two learnable upsampling layers with 9 and 16 filter sizes and 4 and 8 strides. The resulting output now has the same size as the input, with 21 channels corresponding to each class. We froze all the weights except in the last three added layers.

Implementation Details: We trained the custom FCN model with two losses, cross-entropy loss and focal loss. Focal loss has shown promising results in tackling the important class imbalance problem in the current segmentation datasets. In most semantic segmentation datasets, background occupies the majority($> 50\%$) of the pixels, resulting in a strong bias in the network to predict background class. We used a batch size of 32 and a learning rate of 0.0002 and trained for 50 epochs.

Table 1. Results for NN, VGG and Custom model on 6-way classification task) & LSTM

	CE Ep5/Focal Ep5	CE EP10/Focal Ep10	CE Ep15/Focal Ep15	CE Ep50/Focal Ep50
val acc.	**0.645**/0.643	0.644/**0.645**	0.646/**0.650**	0.730/**0.739**

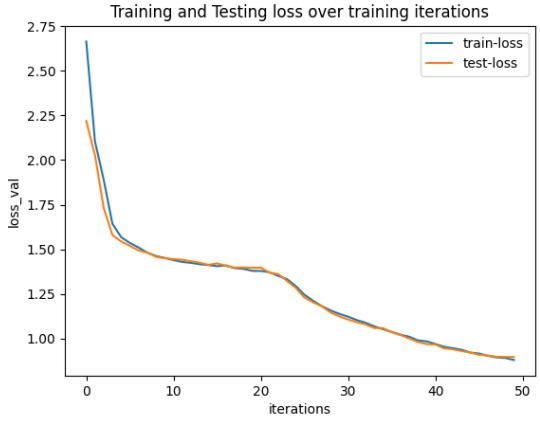

Fig. 2. Plot of training and test losses for Custom FCN model trained with Cross-Entropy loss

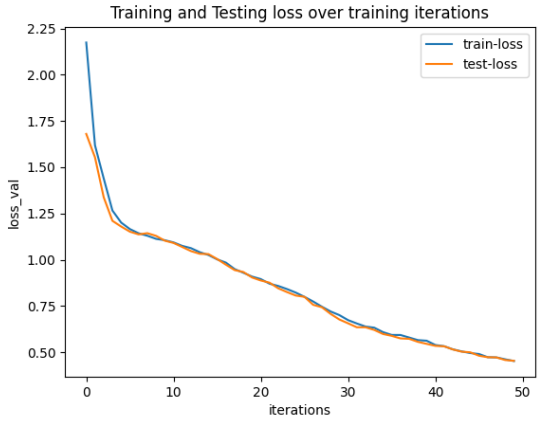

Fig. 3. Plot of training and test losses for Custom FCN model trained with Focal loss

Results: We show the training process through the loss curved in Fig. 2 and 3 for focal loss and the cross-entropy loss. We can observe that the focal loss is able to converge to a lower loss earlier as compared to the cross-entropy loss. From now onwards, we use the focal loss for further experiments. Focal loss is reduced to the value ¡ 1.0 within just 15 epochs which took 25 epochs with cross-entropy loss. Additionally,

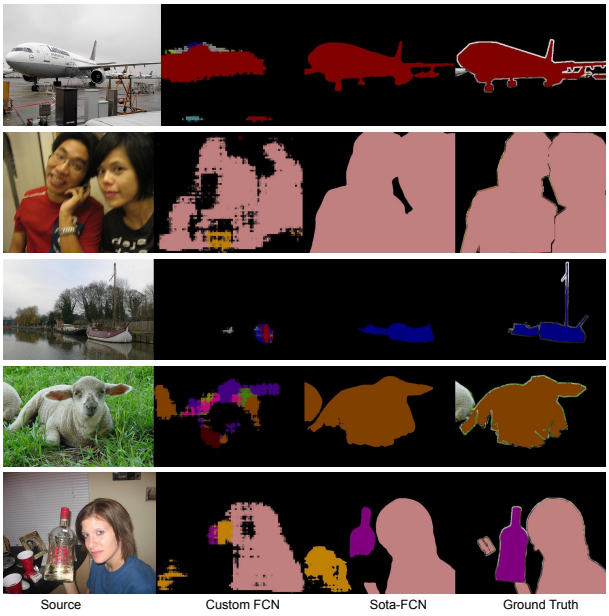

Fig. 4. SIFT feature extraction on a pair of images. The extracted features are marked in the red

we show the pixel-wise classification accuracy of these two loss functions in Table. 1.

We show qualitative results of our Custom FCN model and the off-the-shelf FCN network in Fig. 4. One can observe that for the person class, both the Custom FCN model and the Sota-FCN model are able to localize the persons. However, our custom model struggles to find the exact boundary of the person. Also, the Sota-FCN model can find a nearly perfect segmentation mask for the airplane image, whereas the Custom FCN model can segment out just an approximate shape of the aircraft. For the challenging image of the boat and sheep, Custom FCN failed to create good segments, and Sota-FCN could segment the boats' sheep and region nicely. We can observe from all of these results that the Custom FCN model cannot produce refined segments for the scene objects. The primary suspect is that the backbone is trained on image classification is kept frozen during our training. To perform well on classification tasks, a model needs to learn a global context and be indifferent to the translation and rotation of the object. However, semantic segmentation requires perfect localization of the object. This results in the loss of important image information in the pre-trained backbone, causing the performance degradation of the Custom FCN model.

Conclusion: In this work, we implemented two segmentation algorithms and evaluated them on a set of images. The first algorithm is N-cut algorithm that uses graph data-structure to model an image and performs min cut on the graph to estimate the image segments. The second algorithm is a deep model: Fully Convolutional Network. We trained the custom model with crossentropy loss and focal loss on Pascal VOC dataset. Further, we presented comparison results for all of the presented algorithms.

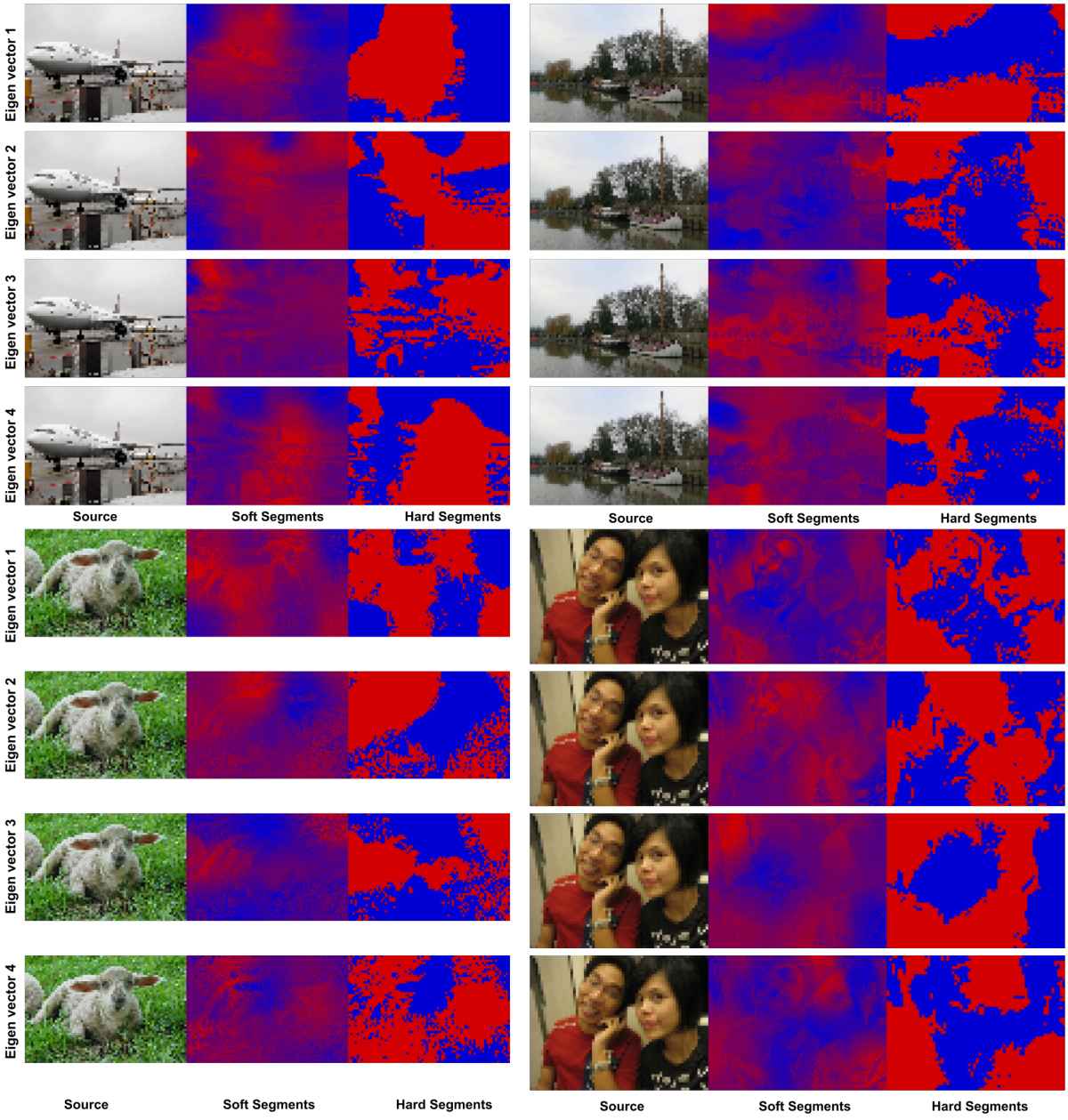

Fig. 5. Results of N-Cut segmentation algorithm on a set of real images. We present segments found from top four eigenvectors. We also show soft segments and hard segments for the input images.