University of Central Florida Fall, 2022
College of Enginering and Computer Science Course Project
Name(s): Manu S Pillai, Sayali Bhosale, Mukund Dhar, Vaibhavi Madhav Deshpande
ID(s): 5487002, 5494025, 5499369, 5500901

CAP5415: Computer Vision

W-Net: A Deep Model for Fully Unsupervised Image Segmentation

Abstract

Even though supervised deep semantic segmentation algorithms for vision tasks have received a lot of attention recently, it is difficult to obtain sufficient supervised pixel-level labels in many domains. Our objective is to reconstruct the W-Net model [1] used for implementing fully unsupervised image segmentation. Concatenating two fully convolutional networks into an autoencoder—one for encoding and one for decoding—is the way they borrowed the recent concepts from supervised semantic segmentation as given in the paper on the W-Net model. During training, the encoding layer results in a k-way pixelwise prediction and both the autoencoder's reconstruction error and the encoder's normalized cut are minimized. On both the BSDS300 and BSDS500 dataset, our implementation's image segmentations performed well for an unsupervised method. The original input image could be distinctly identified from the reconstruction image as well.

1 Introduction

Image segmentation is the process of partitioning an image into multiple image segments, also known as image regions or image objects (sets of pixels). Recent deep learning advances have paved path for better and more complex image segmentation models but the improvement comes with its own pitfall of large pixel level annotated image datasets. As its expensive and difficult to procure pixel level annotated image datasets for segmentation, unsupervised methods for image segmentation is an emerging field of interest. For this project, we have selected and implemented the paper "W-Net: A Deep Model for Fully Unsupervised Image Segmentation" [1] in Pytorch. The model is proposed by Xia et al. for image segmentation task in unlabelled environment. The model has two U-net architectures. The first u-net is an encoder which gives the segmentation of the image and second u-net acts as a decoder which reconstructs image from that segmentation. We implement the W-Net architecture and compute the segmentation results for images in the wild. In section 2, we explain the WNet model architecture and in section 3 we provide the implementation details. Section 4 we explain about the results and section 5 & 6 discusses the implementation and concludes the work respectively.

2 Model Architecture

The W-Net architecture is shown in Figure 1. As per the actual work from the authors, the architecture has 18 modules, each module has two 3×3 convolutional layers, followed by ReLU activation[2] and batch normalization [3]. There are a total of 46 convolutional layers. The architecture is given an autoencoder like backbone, where the first U-net acts like an encoder and the second U-net acts like an decoder. Each U-net consists of 9 modules, half of the 18 modules in the architecture. The encoder gives the segmentation map of the input image, which the decoder, with the remaining 9 modules, takes to reconstruct the input image.

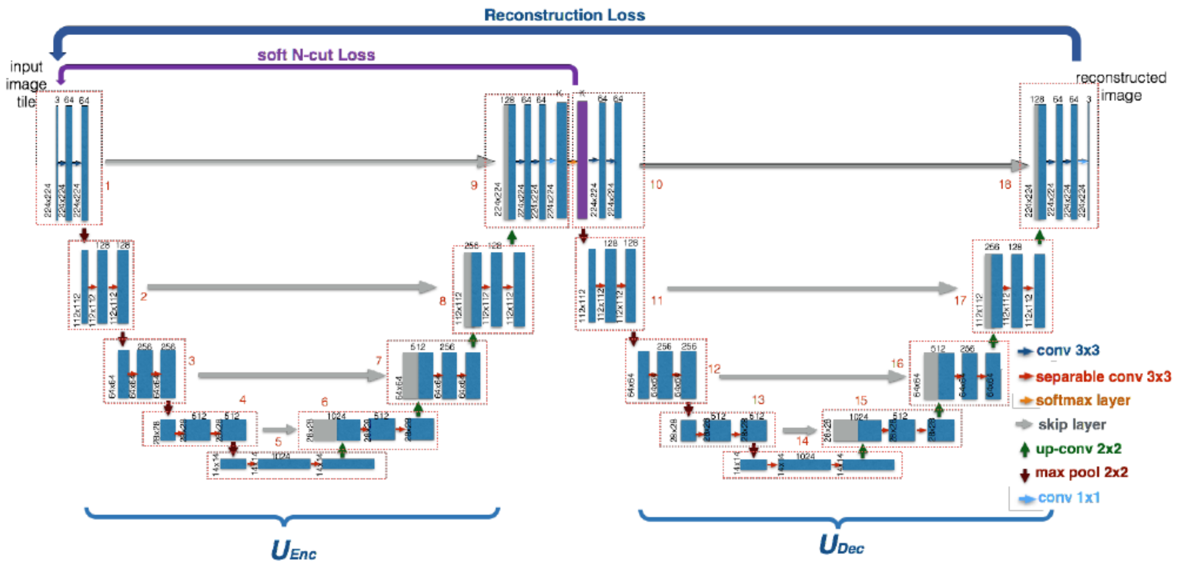


Figure 1: The W-Net architecture [1]

Encoder

Firstly, the input image is resized to 224×224 pixels. Then its passed through a convolutional layer of kernel size 3×3. A 2×2 max pooling layer is applied to the output feature map and its passed to the second module. The operation reduces the dimension of the input image from 224×224 to 112×112, while it doubles the number of feature channels from 64 to 128, so, a depth-wise separable convolution is performed instead of a regular convolution. This operation consists of two types of convolution i.e. a depth-wise convolution and a pointwise convolution which helps in adjusting the feature channels in the network. This procedure is able to achieve better performance but with the same number of parameters. Following the reason, instead of doing a regular convolution, all modules from 2 till 8, and modules from 11 till 17 perform depth-wise separable convolution. Only modules 1, 9, 10 and 18 consist of two regular convolutions of size 3×3.

Module 3, 4 and 5 follows the same architecture like module 2 and we obtain a feature map of size 14×14 with 1024 feature channels in the end of module 5. Now, from module 5 to 6, the feature map dimensions are doubled using the upsampling procedure. The upsampled output is then concatenated with the output of module 4 so as to reduce the spatial information loss due to downsampling. The subsequent module 7, 8 and 9 follows the same procedure, which concatenates the output of module 3, 2 and 1 to the output of the previous module respectively. During the upsampling module stages, we also half the number of channels in the feature maps. The final convolutional layer of the encoder consists of a 1x1 pointwise convolution which maps each channel component of the feature vector to the desired number of classes `K`. In the end, a Softmax layer rescales the logits of `K` class between 0 and 1.

Decoder

The decoder follows a similar procedure as the encoder. It helps to reduce the dimensionality and doubles the number of feature channels by means of a 2×2 max pooling in between modules 10, 11, 12, 13 and 14. Afterwards, it up-samples the feature vectors by doubling the dimensionality and halving the number of feature maps. Final layer of the decoder consists of a 1×1 pointwise convolution that reconstructs the input image from the final feature maps.

Hyperparameter	Value
Batch size	2
Input image size	224
`K`	64
No. of epochs	35
Dropout rate	0.65
Encoder feature channels	{64, 128, 256, 512}
Decoder feature channels	{1024, 512, 256}

Table 1: Hyperparameters and the values used for training the model

3 Implementation Details

3.1 Dataset

The original work uses the PASCAL VOC2012[4] dataset to train the W-Net architecture and evaluates the trained network on the Berkely Segmentation Database (BSDS300 and BSDS500)[5]. BSDS300 and BSDS500 have 300 and 500 images, respectively. For each image, the BSDS dataset provides human-annotated segmentation as ground truth. Due to hardware constraints, in our implementation, we train the model on the combined BSDS300 and BSDS500 dataset. Note that, as the method is designed for unsupervised image segmentation, we do not use any ground truth labels in the training phase; we use the ground truth only to evaluate quality of the model segmentations.

3.2 Codebase

We implement the W-Net architecture using the popular python deep learning library PyTorch. There are two major python files, `train.py` and `predict.py` that contains all the necessary codebase for training and evaluating the W-Net model. The `model.py` file consists the architecture definition.

To train the model, `train.py` can be executed using `python train.py`. To generate a segmentation map for an image, `predict.py` can be executed using `python predict.py <image_file_path>`. It can be noted that, the python file can handle directory path as well, in which case, the program will predict the segmentation map for all the images in the directory and save them into a new directory `output`.

With the codebase, we provide our trained model weights which can be used to generate segmentation map.

3.3 Experimental Details

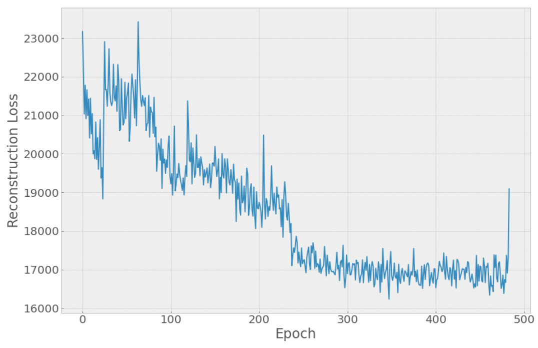
3.3.1 Training

In this section we provide the training details of our model. Table 1 shows the hyperparameters and the values we used for training the model. Its should be noted that, as our batch size was really low, we changed the "Batch Norm" layer in the implementation to "Layer Norm". In the original work, the optimizer for training was not given, we have used "Adam" for the same.

We optimize the model using two losses, the soft normalized cut (soft-N-cut) loss and reconstruction loss. Two separate optimizers are used while training, one that updates the encoder parameters with the soft-N-cut loss, and another to update the whole model. Both the encoder and the decoder parameters uses the reconstruction loss. Equation 1 and 2 gives the reconstruction and soft-n-cut loss we used to train the model respectively.

$$L_{recons} = ||X - U_{Dec}(U_{Enc}(X; W_{Enc}); W_{Dec})||_2^2 \tag{1}$$

where W_{Enc} denotes the parameters of the encoder, W_{Dec} denotes the parameters of the decoder, and X



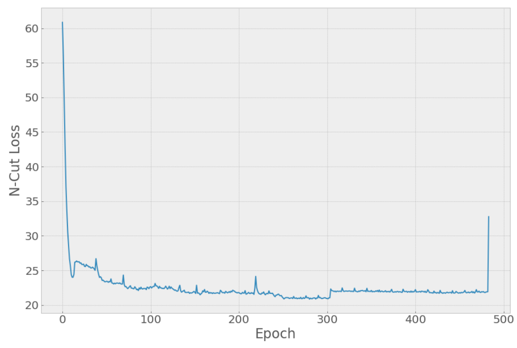


Figure 2: Reconstruction & N-cut loss of the model during training

is the input image.

$$
\begin{aligned}
L_{soft-Ncut}(V,K) &= \sum_{k=1}^{K} \frac{cut(A_k, V-A_k)}{assoc(A_k,V)} \\
&= K - \sum_{k=1}^{K} \frac{assoc(A_k, A_k)}{assoc(A_k,V)} \\
= K - \sum_{k=1} K & \frac{\sum_{u \epsilon V} p(u=A_k) \sum_{u \epsilon V} w(u,v) p(u=A_k)}{\sum_{u \epsilon V} p(u=A_k) \sum_{t \epsilon V} w(u,t)}
\end{aligned} \tag{2}
$$

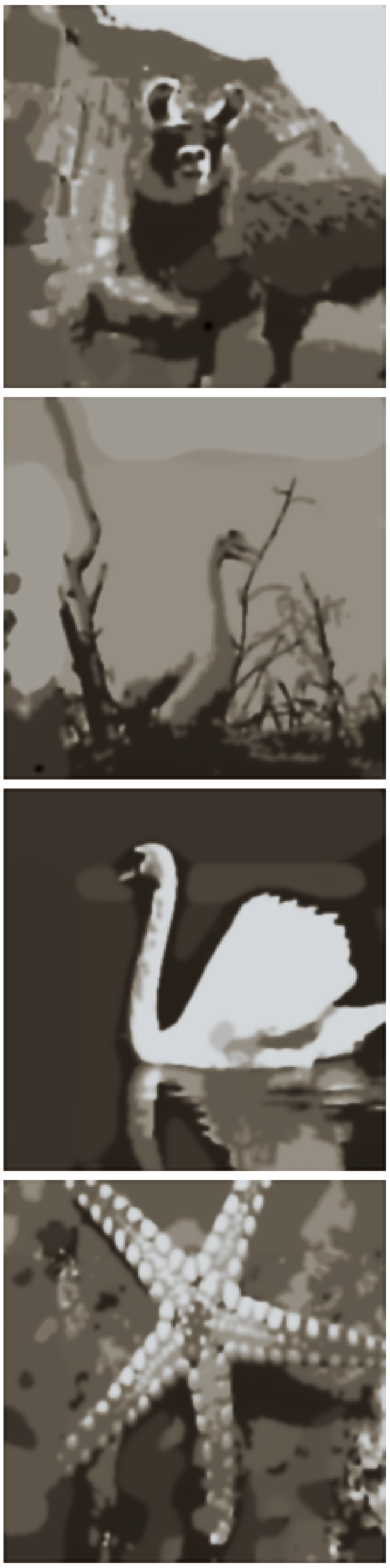
where A_k is set of pixels in segment k, V is the set of all pixels, w measures the weight between two pixels, $p(u = A_k)$ measures the probability of node u belonging to class A_k.

4 Results & Discussion

The reconstruction loss & the N-cut loss of the model during training are shown in figure 2. It is observed that both losses follow almost the same pattern compared to the original work. they fastly reduce at the beginning and after a higher number of iterations they flattens out. Figure 3 shows the segmentations and reconstructed image of the test set.

We also show the segmentation results on multiple in-the-wild images that were not a part of the BSDS dataset, see figure 4. In both the cases, we can see that, the segmentation results are good considering the unsupervised nature. When compared to the performance of the original implementation, we observed that the reconstructed W-Net performs a bit worse. Both losses are greater in magnitude, and the performance metrics scores are lower as well. The following causes could be to blame for this. First, it's important to note that the reconstruction loss's initial magnitude is approximately 23,000 compared to 17,500, while the soft N-cut loss's initial magnitude is approximately 60 instead of 19. That is both are higher compared to the original. We had to make multiple guesses because the weight and optimizer's initialization were not mentioned in the paper. In addition, the paper does not include a crucial piece of information, the number classes "K". This could radically change the N-cut loss which is utilized to update the encoder parameters and hence the reconstruction loss is likewise impacted. Finally, a learning rate schedule was not implemented in which the learning rate is reduced by a factor of 10 for every 1000 iterations. We have done this manually, but the learning rate could have been made significantly smaller toward the end. Hence, the loss continues to fluctuate around a specific point. The stochastic gradient descent with a small batch size or the high learning rate that keeps overshooting the optimum value could be the cause of these oscillations. The learning rate should be decreased, and the batch size should be increased to prevent this.

(a) Input image


(b) Reconstructed image

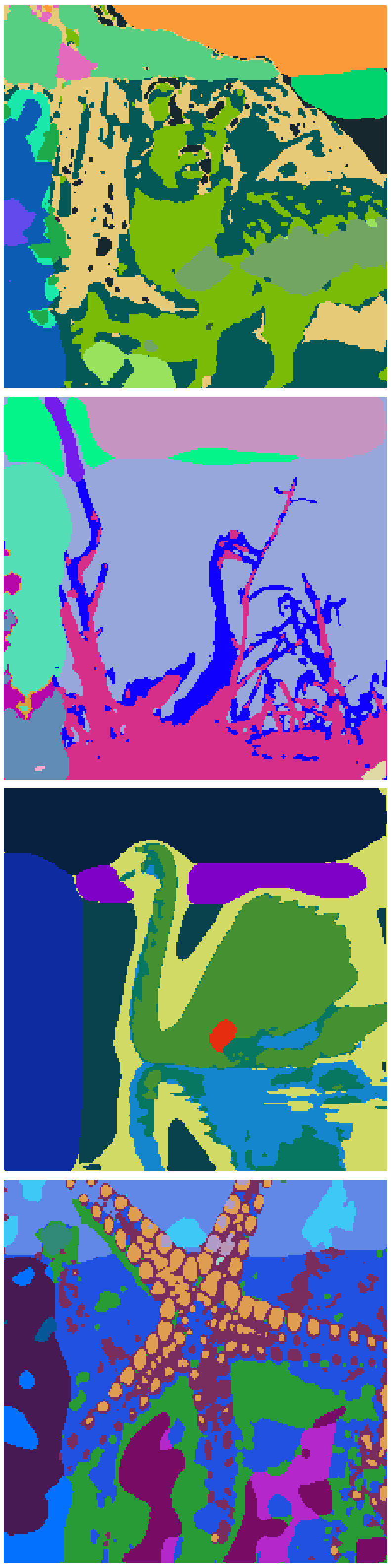
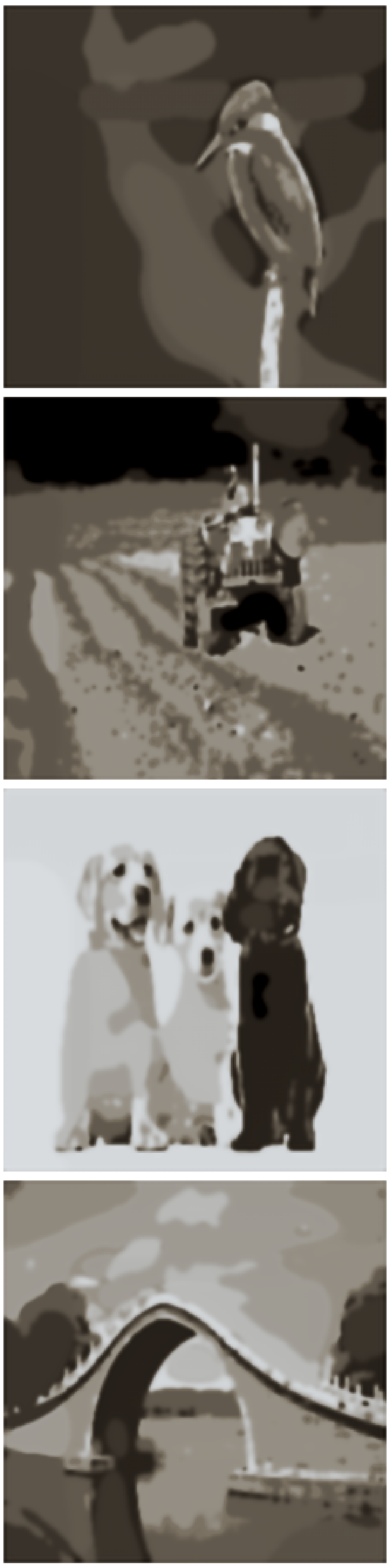

(c) Predicted segmentation map

Figure 3: Input image, its reconstructed image and predicted segmentation map using the trained model

(a) Input image


(b) Reconstructed image

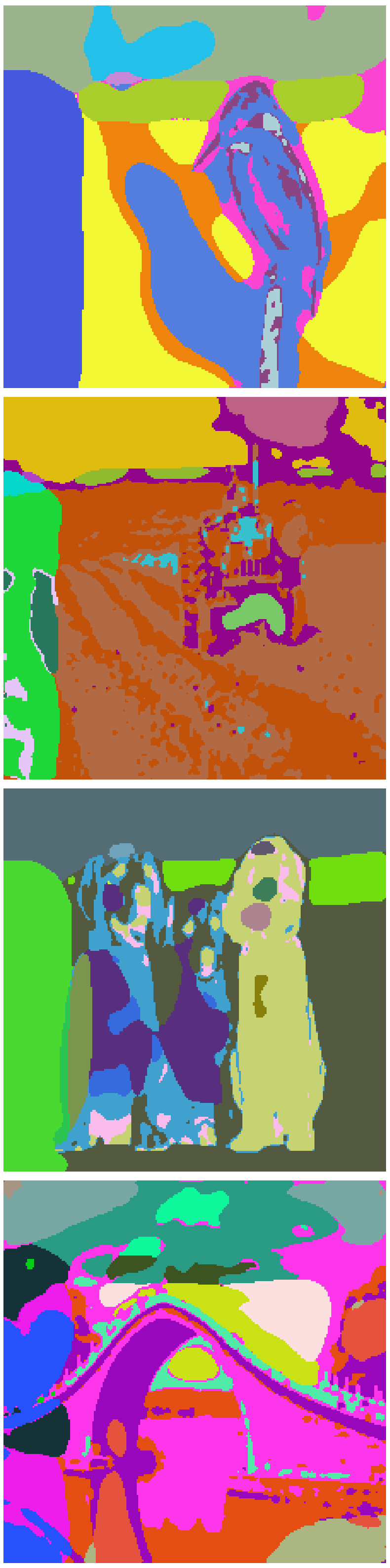

(c) Predicted segmentation map

Figure 4: In-the-wild input image, its reconstructed image and predicted segmentation map using the trained model

5 Conclusion

In this work, we implemented the W-Net model that is used for fully unsupervised image segmentation. We optimize the model using two losses, the soft normalized cut (soft-N-cut) loss and reconstruction loss. In our implementation, we train the model on the combined Berkeley Segmentation Dataset (BSDS300 and BSDS500) dataset. Our implementations showed higher losses than the original base paper's losses. Also, performance metrics are close enough to actual results. We believe the reason behind these changes is the difference in the hyperparameters.

Contribution

Manu S Pillai: Implemented the model architecture in PyTorch. Helped in preparing the project report and latex formatting to compile the final version of the report.

Mukund Dhar: Worked on writing the train.py and predict.py scripts as well as helped in writing the project report.

Sayali Bhosale: Worked on the dataloader and reconstruction loss scripts as well as helped in writing the project report.

Vaibhavi Madhav Deshpande Implemented the softncutloss and dataset.py scripts as well as helped in preparing the project report.

References

[1] X. Xia and B. Kulis, "W-net: A deep model for fully unsupervised image segmentation," *arXiv preprint arXiv:1711.08506*, 2017.

[2] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *International Conference on Machine Learning*, 2010.

[3] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *ArXiv*, vol. abs/1502.03167, 2015.

[4] M. Everingham and J. Winn, "The pascal visual object classes challenge 2012 (voc2012) development kit," *Pattern Analysis, Statistical Modelling and Computational Learning, Tech. Rep*, vol. 8, no. 5, 2011.

[5] D. Martin, C. Fowlkes, D. Tal, and J. Malik, "A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics," in *Proc. 8th Int'l Conf. Computer Vision*, vol. 2, pp. 416–423, July 2001.