PROJECT REPORT

June 7, 2019

Aditya Raj

0.1 Introduction

Image Segmentation is the task of separating Image into different segments i.e into background and foreground. The given problem statement deals with segmenting cell nuclei from the histology images. Semantic segmentation has been performed on the provided dataset with a FCN (Fully Convolutional Network) Autoencoder model. Python and Keras library have been utilized for implementation of the proposed framework. The Encoder and decoder models are defined separately using Functional API and Sequential modelling techniques respectively, for the purpose of further experimenting on with the architecture. A data generator has been used to optimize the computation. The training images have been augmented, increasing the number of training samples from 590 to 1770 (rotation and flipping operation on training images have been performed) to prevent the network from overfitting the dataset. Normalizaiton along the mean as pre-processing has been done. Finally the masks obtained have been thresholded using the Otsu's method. Dice coefficient [4] has been employed for evaluating training. MSE (Mean square error) as loss function has been optimized using Adam for further updating weights with backpropagation.

0.2 Methodology

Various methods have been proposed for semantic segmentation, among which FCN based and Faster R-CNN (Regional CNN) based frameworks have been much experimented with. For this project FCN-Autoencoder has been modelled.

0.2.1 DataSet and Preprocessing

The given dataset has 590 training samples, which have been augmented to 1770 number of samples, including 590 flipped images and 590 images rotated at a 90 degree. The training images have been resized to 320*320 and converted to grayscale. Mean based normalization as shown, has been performed on X (Training Samples) to help network converge faster.

$$X_{norm} = \frac{X - X_{mean}}{X_{std}} \tag{1}$$

0.2.2 Proposed Architecture

The proposed FCN based Autoencoder consists of two sub-models an encoder and a decoder. The Encoder unit for the designed autoencoder consists of four weight layers, each convolutional, with 3 x 3 size filters. In between convolution layers, a simple max pooling operation is employed with kernel dimension 2 x 2. The decoder model has four weight layers similar to the encoder, each convolutional, with kernel dimensions identical to the encoder in an attempt to reconstruct the input. In place of a maxpooling layer in encoder the decoder has an upsampling layer with filter dimension 2 x 2. For adding non-linearity, Relu activaiton for encoder unit and Leakyrelu for decoder unit has been used, to prevent back propagating gradients from vanishing or exploding, a classic machine learning heckle often faced when using sigmoid activation.

Table 1: Detailed layer configurations for Autoencoder Model

Encoder Unit				
Layer	**Type**	**Output shape**	**Parameters**	**Kernel size**
in_Enco	Input Layer	320x320x1	-	-
Enco_1	conv2D	320x320x64	640	3×3
relu_1	ReLU	-	-	
Enco_2	Max pooling	160x160x64	-	2×2
Enco_3	conv2D	160X160X128	73856	3×3
relu_2	ReLU	-	-	
Enco_4	Max pooling	80x80x128	-	2×2
Enco_5	conv2D	80X80X128	147584	3×3
relu_3	ReLU	-	-	
Enco_6	Max pooling	40x40x128	-	2×2
Enco_7	conv2D	40X40X256	295168	3×3
relu_4	ReLU	-	-	
Decoder Unit				
Layer	**Type**	**Output shape**	**Parameters**	**Kernel size**
in_Deco	Input Layer	40x40x256	-	-
Deco_1	conv2D	40x40x256	590080	3×3
Leakyrelu_1	ReLU	-	-	
Deco_2	Up Sampling	80x80x256	-	2×2
Deco_3	conv2D	80X80X128	295040	3×3
Leakyrelu_2	ReLU	-	-	
Deco_4	Up Sampling	160x160x128	-	2×2
Deco_5	conv2D	160X160X128	147584	3×3
Leakyrelu_3	ReLU	-	-	
Deco_6	Up Sampling	320x320x128	-	2×2
Deco_7	conv2D	320X320X64	73792	7×7
Leakyrelu_4	ReLU	-	-	
Deco_8	conv2D	320X320X1	577	3×3
Sigmoid_1	Sigmoid	-	-	

For evaluating the training performance Dice coefficient has been used, Considering two sets X and Y this coefficient can be used to measure the similarity among the two sets. For the task of Semantic Segmentation, this metric can indicate if the model is learning meaningful relationship between the input image and the corresponding mask, higher the dice coefficient the better. If the two sets are identical (i.e. they contain the same elements), the coefficient is equal to 1.0, while if X and Y have no elements in common, it is equal to 0.0. Otherwise it is somewhere in between. The expression for dice coefficient is:

$$DiceCoefficient = \frac{2 \times |X \cup Y|}{|X| + |Y|} \tag{2}$$

The masks obtained from the decoder is then thresholded using otsu's method for the purpose of calculating IOU.

0.2.3 Hyperparameters

The weight initialization for encoder and decoder models is with the keras inbuilt initializer Glorot uniform, that takes the number of input and output units in the weight tensor into consideration. The padding has been set to same, which ensures that the output feature map size is the same as the input feature map size, hence the down sampling is carried out only witht the maxpooling layer, if $k \times k$ is the pooling kernel size, the feature map dimensions, $M \times N$ say, would reduce to $\frac{M}{k} \times \frac{N}{k}$. This makes it easy to tune hyperparameters such as Image size and kernel dimensions for the convolution operation.

The optimizer used is Adam, which is a gradient descent optimization that utilizes the first and second moment of gradients for its computation. The weights are updated every eight training samples, and total number of epochs was chosen to be 150, as the dice coefficient and loss of autoencoder stopped updating after 150 epoch.

The non-linear activation for the encoder and decoder were experimentally selected to be Relu and Leakyrelu respectively. Choosing Leakyrelu for the encoder fixes the dying ReLU problem, since it doesn't have zero-slope sections however for the encoder using relu or Leakyrelu did not make much difference in terms of training performance metric. The dimension of the convolution kernels are kept small (3×3) and the stride was one, to ensure vast information extraction to be used in later layers and complex feature learning in comparison with larger filter sizes which learn generic features. As a trade of between the kernel dimensions the number of filters per convolution layer, lower kernel sizes but higher number of filters have been used.

0.2.4 Results

The following section shows the Dice coefficient curve and the FCN- Autoencoder loss curve for the training of the model.

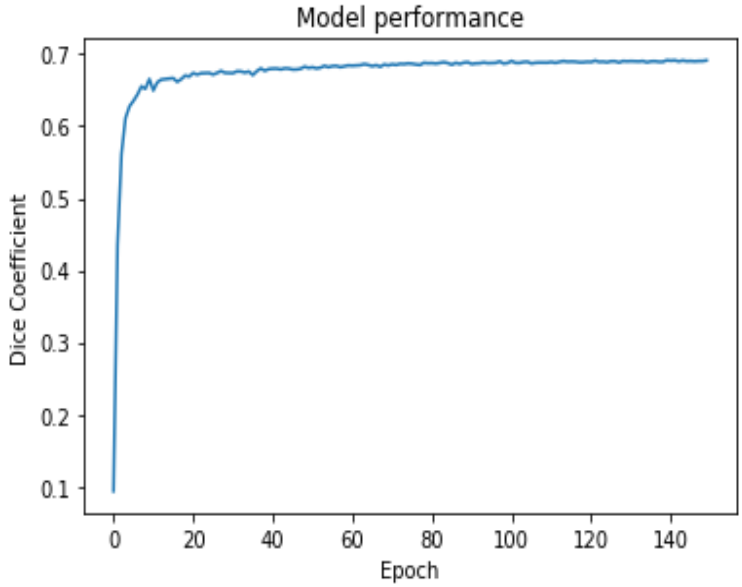

Figure 1: FCN Autoencoder Training Dice Coefficient Plot for 150 epochs

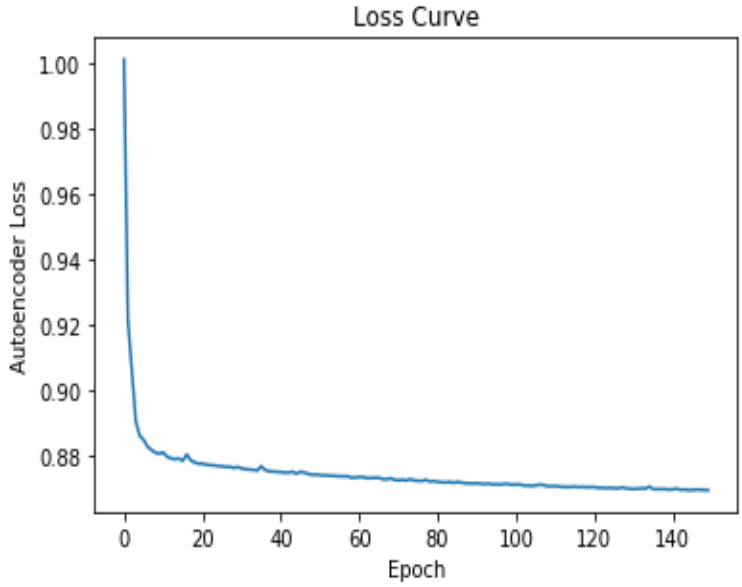

Figure 2: FCN Autoencoder Training Loss curve for 150 epochs

0.3 Ablation Study

- The model with depth of three, four and five weight layers were implemented, the training performance was found to be superior when using four weight layers compared with three and comparable performance was achieved when using five weight layers however the training time increased substantially.
- A multiscale unit was implemented where features maps extracted from kernels with multiple dimensions such as 1×1, 3×3 and 5×5 were averaged, however no significant improvement over the dice coefficient was observed.
- Using mini-batch Stochastic gradient descent (SGD) and Adam as optimizer obtained comparable dice coefficient values however the time taken by SGD to converge was higher.
- various Image sizes from 240×240 to 320×320 were used, using an image with dimensions less then 320×320, reduced the number of trainable parameters hence the performance decreased.

0.4 Future Work

- When using FCN frameworks, Olaf Ronneberger et al. developed UNET [5] an end to end trained FCN based model, which has worked remarkably for the task of semantic segmentation. The architecture contains an encoder and a decoder, where the encoder captures context in the image using convolution and pooling layers and the latter using transposed convolutions learns precise localization.
- The Focal Loss by Tsung-Yi Lin et al. [3] designed to address the one-stage object detection scenario in which there is an extreme imbalance between foreground and background classes during training, can be implemented for improved results.
- Skip connections and Attention could be implemented with the proposed Autoencoder to achieve better results, as the hyper parameter tuning would have taken a considerable amount of time this has been left for future work.

References

[1] A Simple Guide to Semantic Segmentation: A comprehensive review of Classical and Deep Learning methods for Semantic Segmentation, Medium blog

[2] Understanding Semantic Segmentation with UNET: A Salt Identification Case Study, Medium blog

[3] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Doll ÌĄar.Focal loss for dense object detection.arXiv preprintarXiv:1708.02002, 2017.

[4] Milletari,F., Navab, N., Ahmadi, S.A., 2016. V-net: fully convolutional neural networks for volumetric medical image segmentation. In: 2016 Proceedings of the Fourth International Conference on 3D Vision (3DV), IEEE. pp. 565âĂŞ571

[5] Olaf Ronneberger, Philipp FischerThomas Brox, U-Net: Convolutional Networks for Biomedical Image Segmentation, MICCAI 2015: Medical Image Computing and Computer-Assisted Intervention âĂŞ MICCAI 2015 pp 234-241

[6] Deciding optimal kernel size for CNN, Medium blog