Programming Assignment 3

Katarina Vuckovic
CAP5415 Computer Vision
Nov 27, 2021
Instructor Dr. Yogesh Singh Rawat

I. Part 1: CNN Image Classification

A. Introduction

The purpose of this part of the assignment is to design a convolutional neural network (CNN) that will perform image classification on the CFAR dataset. In this work, the performance of two CNNs is compared. The requirement for the first CNN is to have more than 2 convolutional (conv) layers and more than 1 fully connected (FC) layer. The requirement for the second CNN is to have more conv layers than the first CNN.

B. Dataset

The project uses the CFAR10 dataset for evaluation. The dataset consists of colored images that are labeled by one of ten possible classes (airplane, automobile, bird, cat, deer, dog, from, horse, ship, and truck). The size of the images are 32x32. The total number of samples in the dataset is 6000 out of which 5000 training samples and 1000 testing samples. The CFAR dataset is available from the *torchvision* library.

C. Neural Network Architecture

This section describes the architecture and parameters used in designing the two CNNs. Both CNNs are trained over 30 epoch using cross-entropy loss and Stochastic Gradient Descent (SGD) optimizer with learning rate 0.001. Furthermore, the last FC layer is followed by a softmax layer. The batch size is 50.

The first CNN network design consists of three conv layers and three FC layers. Each conv layer is followed by a ReLU activation function and a maxpool layer. A flattening function is used after the three conv layers and before the FC layers. .

The second FC network is the same as the first expect is as an additional conv layer. The parameters (i.e. number of filters, number of neurons...etc) in the conv and FC layers are also different. The details on the parameters for the two layers are listed in Table I.

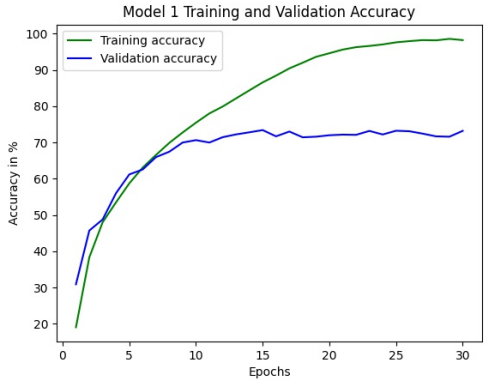
TABLE I: CNN Models

layers	Model 1	Model 2
conv1	conv(3,16,3,p=1)	conv(3,4,3,p=1)
activation function	ReLU	ReLU
pool	maxpool(2,2)	maxpool(2.2)
conv2	conv(16,32,3,p=1)	conv(4,16,3,p=1)
activation function	ReLU	ReLU
pool	maxpool(2,2)	maxpool(2.2)
conv3	conv(32,64,3,p=1)	conv(16,32,3,p=1)
activation function	ReLU	ReLU
pool	maxpool(2,2)	maxpool(2.2)
conv4	-	conv(32,64,3,p=1)
activation function	-	ReLU
pool	-	maxpool(2.2)
flatten	(1,64*4*4)	(1, 64*2*2)
FC1	(64*4*4,512)	(64*2*2,128)
activation function	ReLU	ReLU
FC2	(512,64)	(128,64)
activation function	ReLU	ReLU
FC3	(64,10)	(64,10)
activation function	ReLU	ReLU

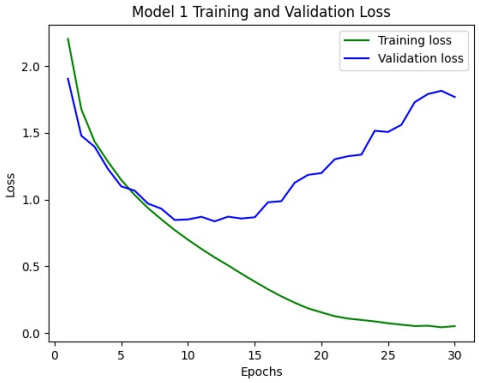
D. Results and Discussion

Figure 1a shows the training and testing accuracy calculated at each epoch for model 1 while Figure 1b shows the training and testing loss for the same model. As may be seen from the figures, the model converges after about 10 epochs with an

accuracy of 72%. Similarly, Figure 2 shows the accuracy and loss for model 2. The model converges at the 18^{th} epoch with an accuracy of 67%. In this case adding another convolutional layer did not improve the performance of the algorithm.

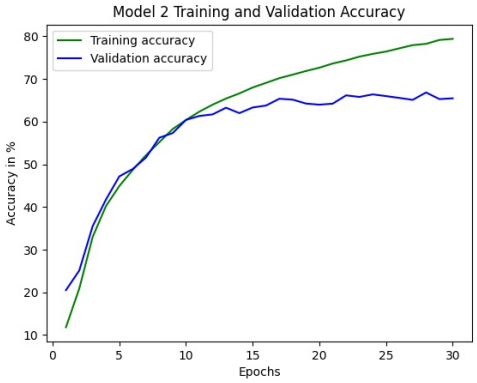


(a) Training vs. validation accuracy at every epoch

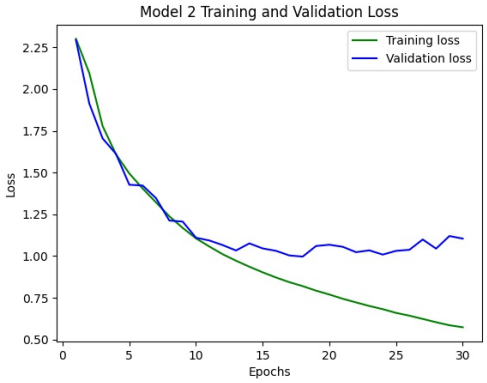


(b) Training vs. validation loss at every epoch

Fig. 1: Results for Model 1

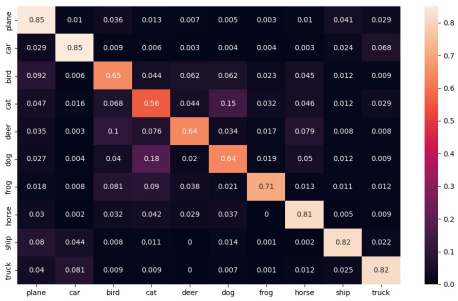


(a) Training vs. validation accuracy at every epoch

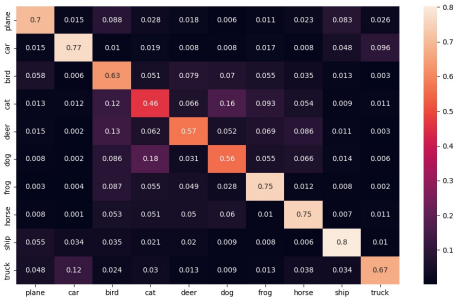


(b) Training vs. validation loss at every epoch

Fig. 2: Results for Model 2



(a) Model 1 Confusion Matrix



(b) Model 2 Confusion Matrix

Fig. 3: Confusion Matrices

Figure 3 shows the confusion matrices for model 1 and 2. The values on the diagonal represents the accuracy for each model. Figure 3a shows that for model 1 the highest accuracy of 85% is shared by 'car' and 'plane' class while the lowest

accuracy of 56% is for the 'cat' class. The cat is most often mistaken for the 'dog' class. The confusion matrix for model 2 in Figure 3b shows a similar trend but with reduced accuracy. The highest accuracy of 77% is achieved for the class 'car' and the lowest accuracy of 46% is achieved for class 'cat' which is again most commonly mistaken for class 'dog'.

E. Conclusion

From these results, we may conclude that just adding another layer to the neural network did not improve the accuracy. In general, creating deeper networks tends to improve the performance but sometimes just adding another layer is insufficient. Improving the performance of the classifier may still be possible by increasing the depth of the network, however other network parameters would have to be adjusted as well. The learning rate and the number of epochs may need to be changed. Furthermore, deeper networks typically require some sort of regularization such as dropout.

II. Part 2:Image Segmentation

A. Introduction

The purpose of this part of the assignment is to implement two thresholding based algorithms for image segmentation. The first is the binary image segmentation where the threshold is manually set while the second is the Otsu tresholding where the algorithm learns the optimal treshold from the input image.

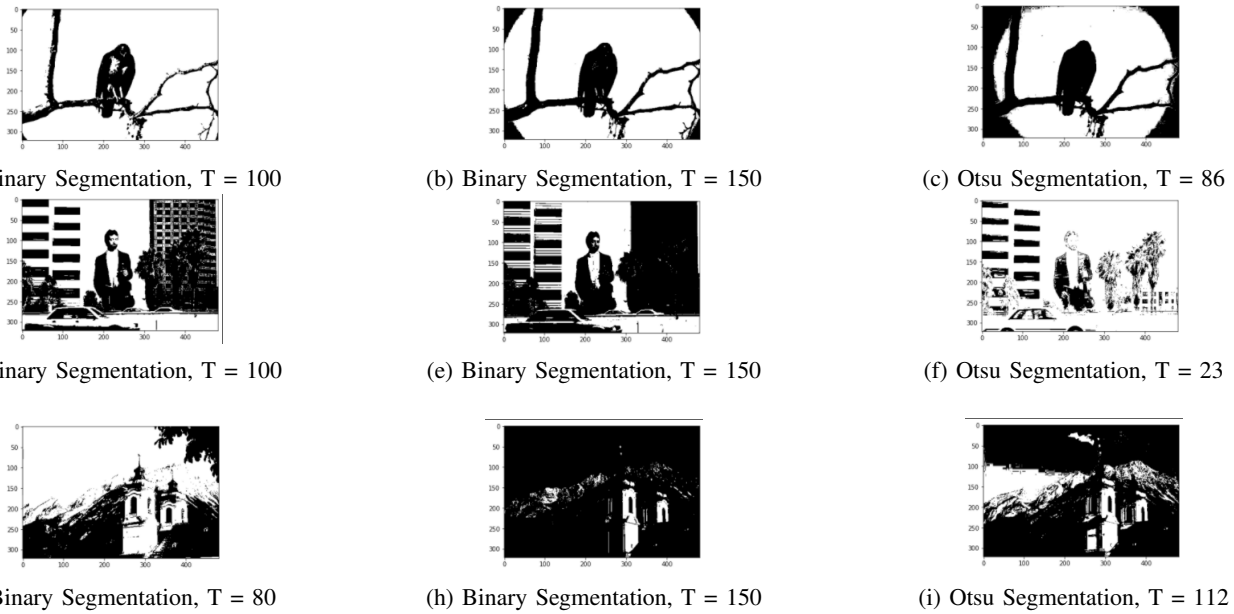
B. Thresholding Segmentation

In binary segmentation, all pixels are split into either foreground or background based on some threshold. If the pixel value is above the threshold then the pixel is set to the maximum value and if the pixel is below the threshold it is set to the minimum value. For example, if the range of pixel values is between 0 and 255 and a threshold is set to 100, then all pixels greater than 100 will be set to 255 and all pixels below will be set to zero.

The challenge in binary classification is to determine the optimal threshold value. One method is to look at the histogram of the image and pick a threshold where a local minimum occurs. To address the problem of selecting the optimal threshold in binary classification, the Otsu thresholding algorithm is proposed. This algorithm estimates the optimal threshold by maximizing the between-class variance of the segments in the histogram.

C. Results

The binary classifier and Otsu were tested on three input samples. The results are presented in Fig. 4. For each sample, there are three output results presented, the first two are from binary and the third is from Otsu thresholding. The binary was tested on two different thresholds which were estimated using the histogram. The threshold used for each output including the Otsu is represented by T in Fig.4. For the first input, the threshold for binary was set to 100 and 150, but the optimal Otsu is 86. In the second input the threshold for binary was also set to 100 and 150, but the Otsu was only 23. Finally, in the last input, the threshold for binary were estimated to 80 and 150 but the Otsu was 112.



(a) Binary Segmentation, T = 100 (b) Binary Segmentation, T = 150 (c) Otsu Segmentation, T = 86

(d) Binary Segmentation, T = 100 (e) Binary Segmentation, T = 150 (f) Otsu Segmentation, T = 23

(g) Binary Segmentation, T = 80 (h) Binary Segmentation, T = 150 (i) Otsu Segmentation, T = 112

Fig. 4: Comparison of Binary and Otsu Thresholding Segmentation on Three Different Samples

D. Conclusion

From these results, we concluded that is it is difficult to set the optimal treshold manually since most histograms have many local minima and therefore it is hard to select which one is the best. As may be seen from the results, non of the estimated threshold were close to the Otsu. Therefore, Otsu is a better way of implementing a thesholding technique as it does not require the designer to manual select the treshold and typically the Otsu algorithm treshold works better than the manual settings.