COMP 448/548 – MEDICAL IMAGE ANALYSIS

HOMEWORK #2

Implementation details

Specify the platform that you used for your implementation.

Explain how you made use of the pretrained network to design your own classifier:

1. How did you make the input size compatible with the network?
2. How did you normalize the input?
3. What parts of the architecture did you modify? How did you modify the last layer?
4. What loss function did you use in backpropagation?
5. How did you select the parameters related to backpropagation? For example, did you use any optimizer? If so, what were the parameters of this optimizer and how did you select their values?
6. How did you address the class-imbalance problem?

Additional comments, if you have any.

	Training portion of the training set				Validation portion of the training set				Test set			
	Class 1	Class 2	Class 3	Overall	Class 1	Class 2	Class 3	Overall	Class 1	Class 2	Class 3	Overall
With input normalization and with addressing the class-imbalance problem	0.841	0.895	0.790	0.846	0.95	0.96	1.00	0.967	0.461	0.571	0.685	0.573
With input normalization and <u>without</u> addressing the class-imbalance problem	0.769	0.857	0.636	0.746	0.861	0.923	0.769	0.935	0.519	0.739	0.191	0.479
<u>Without</u> input normalization and with addressing the class-imbalance problem	0.71	0.837	0.729	0.773	0.708	0.884	1.00	0.838	0.210	0.487	0.750	0.458

Before deciding on the model architecture, I performed in-depth research on popular medical image classification models. My criteria for selecting an architecture were overall accuracy and computational efficiency. I considered AlexNet, ResNet, and MobileNetV2 due to their popularity. I set up an experiment to train and test these three models over 5 epochs. To improve model generalization, I experimented with creating a validation dataset from the training dataset with different split ratios of 0.15, 0.2, and 0.30. I tested various configurations of learning rate and momentum values along with different optimizers such as SGD, Adam, and RMSprop. I also adjusted the step size and gamma. value for the learning rate scheduler. Initially, both training and testing accuracies were considerably low. Among these, SGD performed the best with a split ratio of 0.2.

Since the success in improving accuracy through traditional train-validation splits is limited, I opted to implement the k-fold cross validation method. This approach significantly increased the model's performance compared to using a fixed validation dataset.

1. To make the input size compatible with the AlexNet network, I resized all input images to 224x244 pixels, which is the required dimension expected by the model. This operation is accomplished in the code part “transformations” by Resize(224, 224) function.
2. In the preprocessing code, I included Normalize() function to normalize the input images. This function adjusts each image by subtracting the mean and diving by the standard deviation of the ImageNet dataset. The mean values are [0.485, 0.456, 0.406] and the standard deviation values are [0.229, 0.224, 0.225]. This normalization process is applied channel-wise(R, G, B for our task). Standardizing the input data in this manner helps improve the performance of the neural network. Because when input features are on different scales, it can significantly distort the error resulting in difficulty for gradient descent to navigate efficiently. Consistent data distribution is also important since pre-trained models like AlexNet are typically trained on very large datasets like ImageNet, where the data has been normalized.
3. I modified the last layer of the AlexNet network to meet the specific requirement of the assignment. While I retrained the pre-trained weights for all the other layers to leverage the learned features, I specially adjusted the final layer to output for three classes.
4. For the assignment involving three-class classification, I used the cross-entropy loss function during backpropagation. This loss function is ideal for such tasks as it compares the predicted probabilities directly with the true labels and helps optimize the network’s parameters. By minimizing this loss, I aimed to enhance the accuracy of my classification model.
5. I managed the backpropagation process using the SGD optimized with a learning rate of 0.001 and a momentum of 0.6. I also implemented a learning rate scheduler with a step size of 5 and a gamma value of 0.4. I chose these parameters based on experiments mentioned above in a training and test environment created by me.
6. In my implementation, the class-imbalance problem was addressed by incorporating a weighted loss function during the training process. When calculating the loss for a batch of training data, the weighted cross-entropy loss function penalized misclassifications of the minority class more than those of the majority class. The idea behind using weighted loss, where the weights are inversely proportional to class frequencies, is to balance this impact. Instead of increasing the penalty for misclassifications of the majority class, I increased the penalty for the minority class to ensure the model learns to classify all classes accurately.

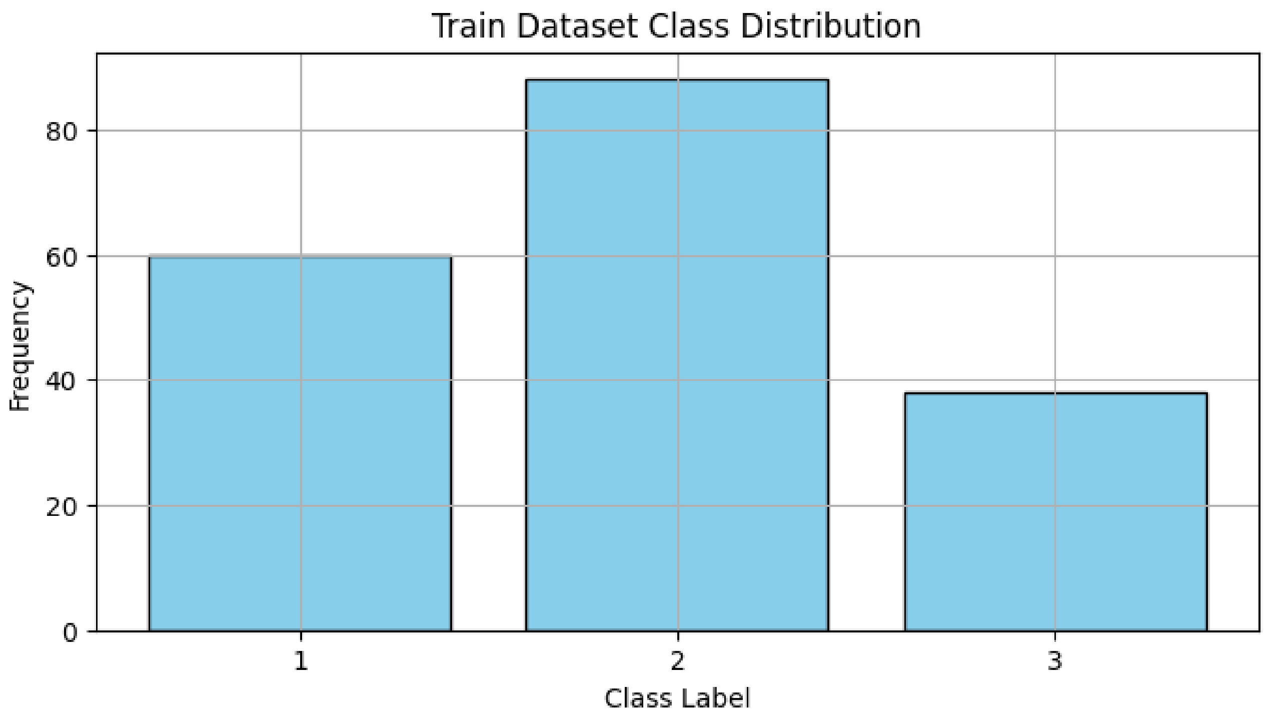

Figure 6.1. Training Dataset Class Distribution

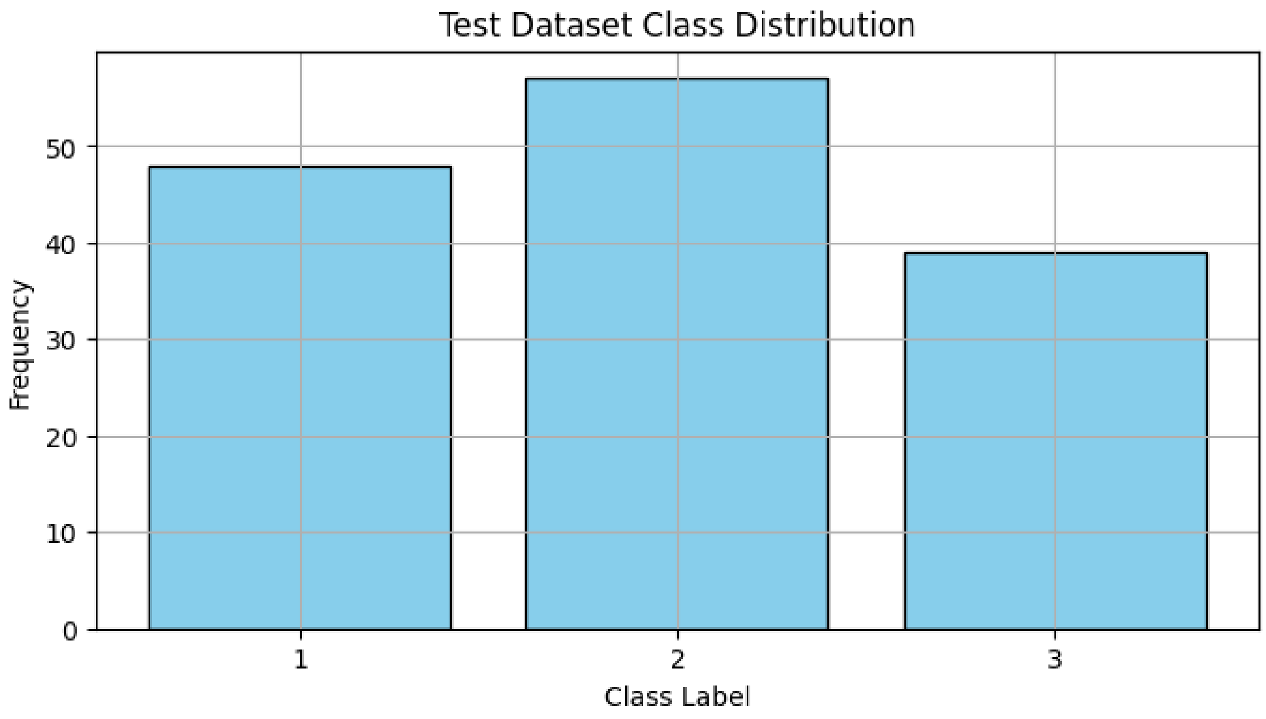

Figure 6.2. Testing Dataset Class Distribution