ECE763: Project-2 Report

Instructor: Dr. Tianfu Wu

Student Name: **Vinay Kumar** UnityID: **vkumar24@ncsu.edu**

GIVEN:
The naming conventions used in this proof of the AdaBoost steps are listed below:

- Init_step: The **weights** for each step for each input-data is $W_t(i)$. Initialize the **weights** $W_t(i)$ with **Uniform-Distribution** for all training data x_i in X_{train}:

$$\{(x_1, y_1), (x_2, y_2) \cdots (x_N, y_N)\} \in X_{train} \qquad \text{(labelled training data)}$$

$$\sum_{i=1}^{N} W_t(i) = 1 \qquad \forall\ t;\ \text{(abiding to the Uniform-Distribution assumption)} \tag{1}$$

$$W_1(i) = \frac{1}{N} \qquad \forall\ (x_i, y_i)\ \in\ X_{train}\ \text{(initialization of the weights)} \tag{2}$$

- Boosting_step: For each iteration $\mathbf{t}\ \in\ \{1, 2, \cdots \mathbf{T}\}$

$$\epsilon_t(h) = \sum_{i:\ h_t(x_i) \neq y_i} W_t(i) \tag{3}$$

$$= \sum_{i=1}^{N} W_t(i) \times \begin{cases} 1 & \text{if } h(x_i) \neq y_i \\ 0 & \text{if } h(x_i) = y_i \end{cases} \qquad \forall\ h;\ \forall\ (x_i, y_i)\ \in\ X_{train} \tag{4}$$

$$h_t = \arg\min_{h\ \in\ \Delta} \epsilon_t(h) \tag{5}$$

$$\epsilon_t = \epsilon_t(h_t) \tag{6}$$

$$\alpha_t = \frac{1}{2} \cdot \log\left(\frac{1 - \epsilon_t}{\epsilon_t}\right) \tag{7}$$

$$Z_t = \sum_{i=1}^{N} W_t(i) \times \begin{cases} e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \\ e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \end{cases} \qquad \forall\ (x_i, y_i)\ \in\ X_{train} \tag{8}$$

$$= \sum_{i=1}^{N} W_t(i) \cdot e^{(-y_i \alpha_t h_t(x_i))} \tag{9}$$

$$W_{t+1}(i) = \frac{W_t(i)}{Z_t} \times \begin{cases} e^{\alpha_t} & \text{if } h_t(x_i) \neq y_i \\ e^{-\alpha_t} & \text{if } h_t(x_i) = y_i \end{cases} \qquad \forall\ (x_i, y_i) \in X_{train} \tag{10}$$

$$= \frac{W_t(i)}{Z_t} \cdot e^{(-y_i \alpha_t h_t(x_i))} \tag{11}$$

$$\mathcal{H}_t(x) = sign\left(\sum_{\hat{t}=1}^{t} \alpha_{\hat{t}} \cdot h_{\hat{t}}(x)\right) \qquad \forall\ h_{\hat{t}} \in \Delta;\ \ (x \text{ is the test/valid un-labelled data}) \tag{12}$$

$$t \leftarrow t + 1 \qquad \text{(increment the loop)} \tag{13}$$

- Final_step:

$$\mathcal{H}(x) = sign\left(\sum_{t=1}^{T} \alpha_t \cdot h_t(x)\right) \qquad \forall\ h_t \in \Delta;\ \ (x \text{ is the test/valid un-labelled data}) \tag{14}$$

The face-detection algorithm used by Viola JOnes had few very important parts, inovations and implications too:

- **Integral Image**: I have implemented it as **modules/dataloader.py: create_integral_image()** method.
- **Haar-Features**: As described in the paper I have implemented the methods and classes to extract the haar features (without using any external library) in **modules/haar_features.py**. All five types of haar features have been extracted.
- **AdaBoost Algorithm**: Adaboost algorithm as used in ViolaJones paper has been implemented in **modules/adaboost.py**. I haveimplemented two different ways of classification Cascades during the testing phase (mentioned in paper and other literatures): (a). Sequential Cascade (2). Committee Cascading inside **modules/adaboost.py: Cascade()** class.
- **Thresholding & Polarity of each classifier:** It was one of the most difficult but interesting part of the project. I have used sequential counting of having encountered faces and yet to be encountered faces (and non faces) and vice-versa to decide thresholding and its polarity based on the orientation of the best value picked. Its has been implemented in **modules/adaboost.py: calc_threshold_and_polarity()**.

- **Weight Initialization & Normalization**: I have designed two methods to initialize the weights of the data samples before running AdaBoost. Using **Uniform** and Class-wise Uniform (**XUniform**) distribution. Both has been implemented in **modules/adaboost.py: init**$_{w}eights()$.**Error Calculation and Alpha Calculation** : $Ithasbeenimplemenedin$**m**
I have also enclosed the plots and a JupyterNotebook for visualizing the results.