Final Project Report

Submitted by: Harshvardhan Bhatnagar

1. Introduction

Our project goal was to use machine learning to determine whether a person in an image is wearing a helmet or not. The dataset used was from Kaggle Helmet Detection. The dataset contained 764 images that had been labeled for the two categories:

- With Helmet (label: 0)
- Without Helmet (label: 1)

The problem of helmet detection is important because it can be used in a variety of applications such as road safety and construction site safety. This particular dataset was chosen because of bounding box information provided a good starting point, reducing some steps from the process of inference.

2. Background

The data was initially available as 764 images along with their respective annotation files which provided information about the bounding-boxes in each image and the target label for each bouding box (`With Helmet`, `Without Helmet`). The data was first preprocessed to extract the parts of the images based on the bounding box information. These images were stored in a directory, along with two csv files:

- `objects.csv`: which contained the bouding box information, the name of the cropped image, and the name of the original image as well as the target label.
- `images.csv`: which simply contained the name of the name of the original image.

For all future steps, we used an output value of 0 as a positive label (i.e. `With Helmet`) and an output value of 1 as a negative label (i.e. `Without Helmet`).

Before training, we first used a label encoder to convert the target labels from strings to integers. The label for `With Helmet` was set to 0 and the label for `Without Helmet` was set to 1. The data was then split into training and validation sets based on the original images. Another approach was also used with stratification based on the target labels however, the results were only marginally better and it did not seem realistic since we were looking at parts of the same image in the training and validation sets.

We prioritized the ROC AUC score over classification accuracy because in this case we would like to minimize the number of false positives. This is because we would like to avoid classifying

a person without a helmet as having a helmet. This is because the consequences of a false positive are much more severe than those of a false negative.

3. Methods

Since the dataset was relatively small, we decided to use a held-out test set for validation. We used a 80-20 split for the training and validation sets respectively. We also used a label encoder to convert the target labels from strings to integers.

We decided to use CNNs for this problem because of their ability to extract features from images. In order to train our CNNs all images needed to have the same dimensionality which was estimated to be 45x45 pixels, using simple scatter-plots of the resolutions of the cropped images extracted earler. We defined a class called `ImageLoaderResizer` which could be used to load the cropped images and resize them based on the information in `objects_df`. This class was then used to create a `Dataset` object which was then used to create a `DataLoader` object. This `DataLoader` object was then used to train the CNNs.

3.1 Initial Experimentation

We attempted using a simple CNN with 3 convolutional layers and 2 fully connected layers, and an output layer that used a Sigmoid Function. The results were not very good with a ROC AUC score of 0.65. Following several unsatisfactory experiments, we defined an `ObjectClassifier` class which could be used to train and evaluate CNNs while varying the number of convolutional layers, their respective kernel sizes, output channels for the layer, and dropout rates. This class was then used to train and evaluate several CNNs with varying hyperparameters. The results of these experiments are discussed below. After initial experiments it was also determined that using:

- `ReLU` activation function for the convolutional layers, along with pooling layers with a kernel-size of 2
- two fully connected layers, and a `LogSoftmax` activation function for the output layer worked best.
- Using the `Adam` optimizer with a learning rate of 0.001

Seemed to work best. These parameters were used for all future experiments.

3.1.1 Model 1

After experimentally reaching the intermediate conclusions above, we trained a CNN with 2 convolutional the Kernels had the following parameters:

- Kernel 1: 3x3, 16 output channels
- Kernel 2: 3x3, 32 output channels
- dropout probability: 0.4

The fully connected layers had 128 and 2 output channels respectively. The dropout probability was set to 0.4. The model was trained for 50 epochs with a batch size of 32. The Loss Function used was `NLLLoss` and the optimizer used was `Adam` with a learning rate of 0.001. The results were as follows:

- training loss on final epoch: 0.055
- training accuracy on final epoch: 0.980

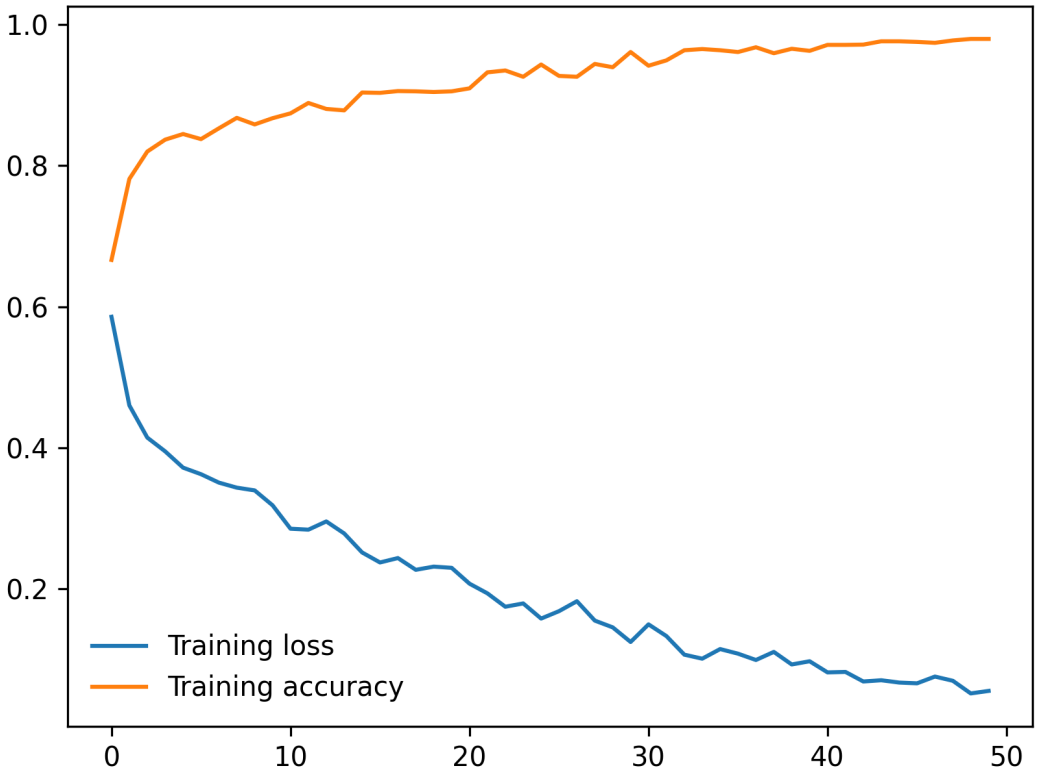


Figure 1: Training Loss and Accuracy for Model 1

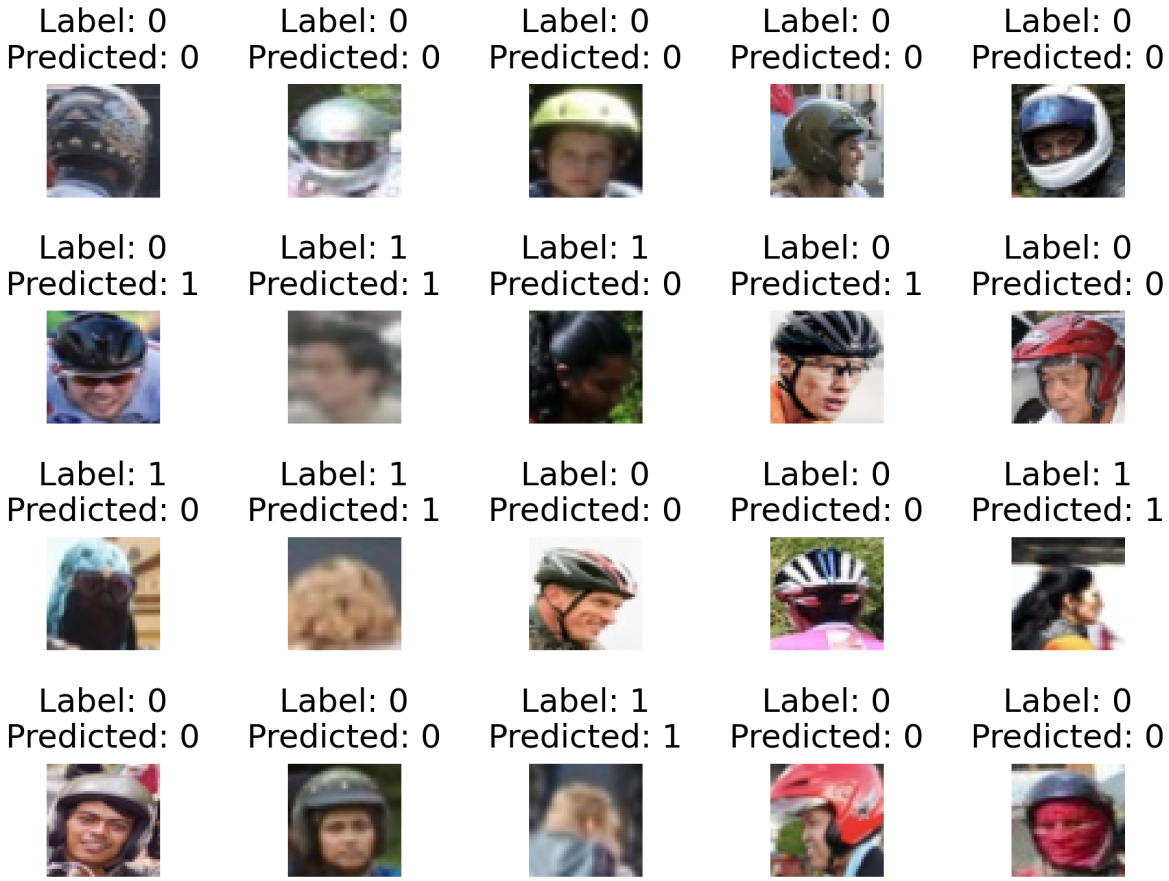


Figure 2: 20 Random Sample Predictions from Model 1

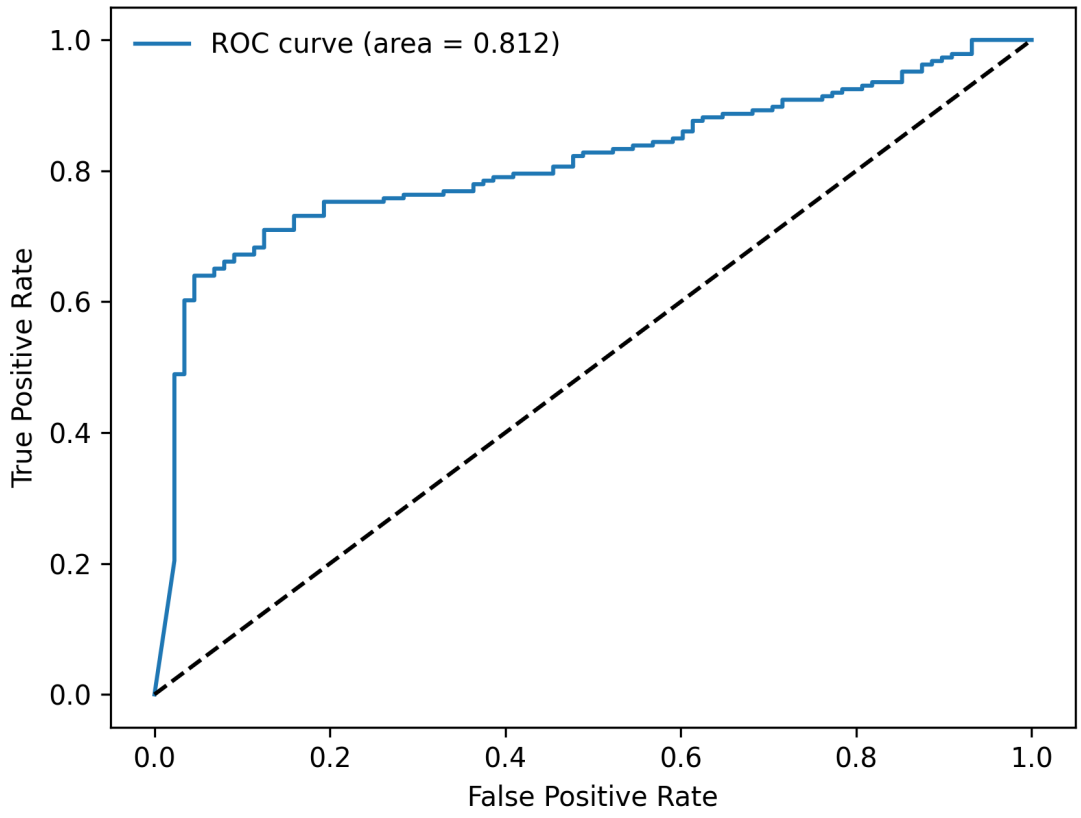


Figure 3: ROC Curve for Model 1

The ROC-AUC Score was thus determined to be 0.812, along with a test classification accuracy of 0.884. The model was then saved to `models/model_1.pt`.

3.1.2 Model 2

We then trained a CNN with 3 convolutional layers. The Kernels had the following parameters:

- Kernel 1: 3x3, 16 output channels
- Kernel 2: 3x3, 32 output channels
- Kernel 3: 3x3, 64 output channels
- dropout probability: 0.4

The fully connected layers had 128 and 2 output channels respectively. The dropout probability was set to 0.4. The model was trained for 50 epochs with a batch size of 32. The Loss Function used was `NLLLoss` and the optimizer used was `Adam` with a learning rate of 0.001. The results were as follows:

- training loss on final epoch: 0.091
- training accuracy on final epoch: 0.966

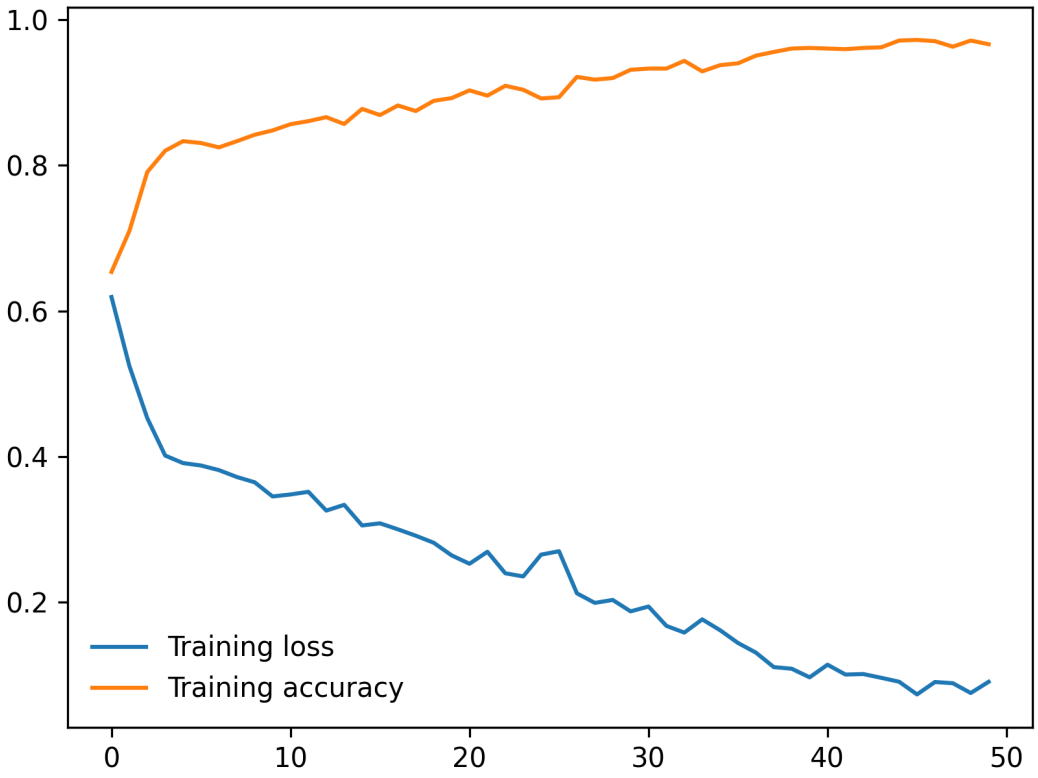


Figure 4: Training Loss and Accuracy for Model 2

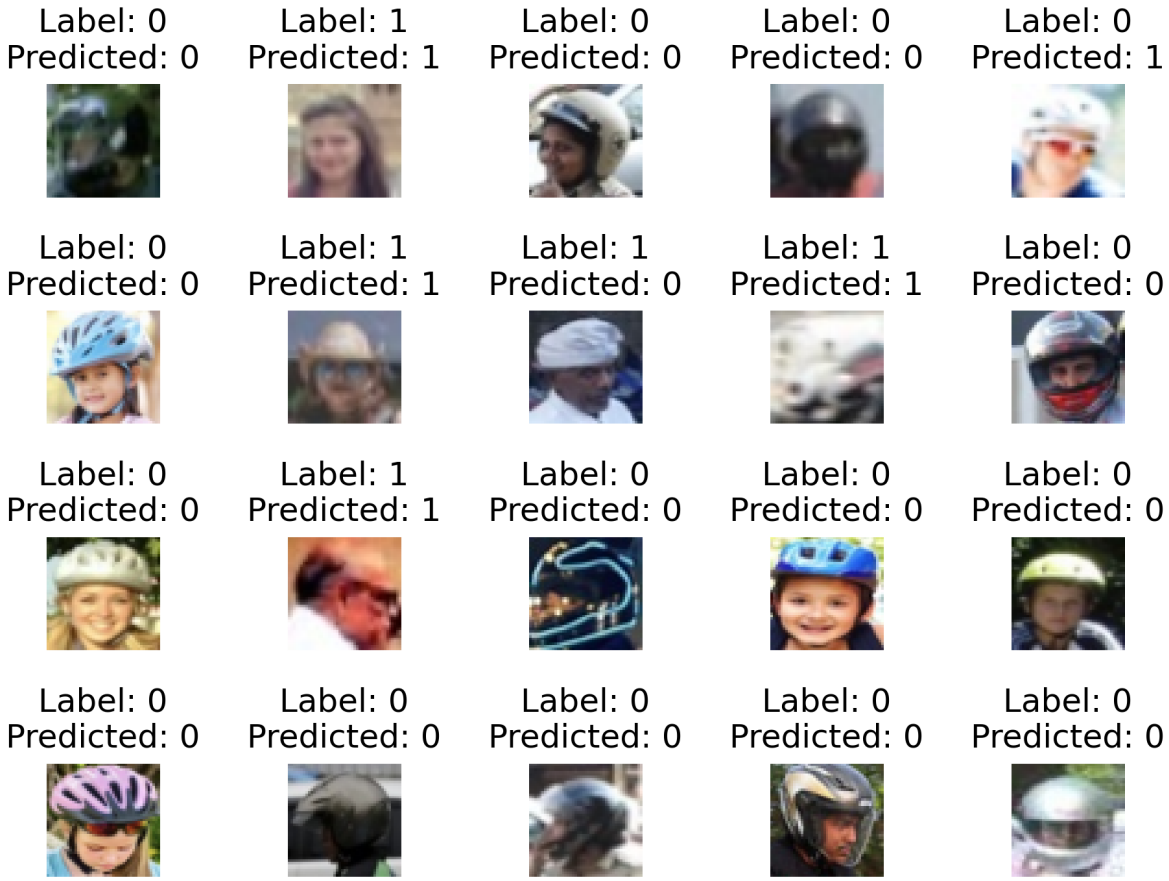


Figure 5: 20 Random Sample Predictions from Model 2

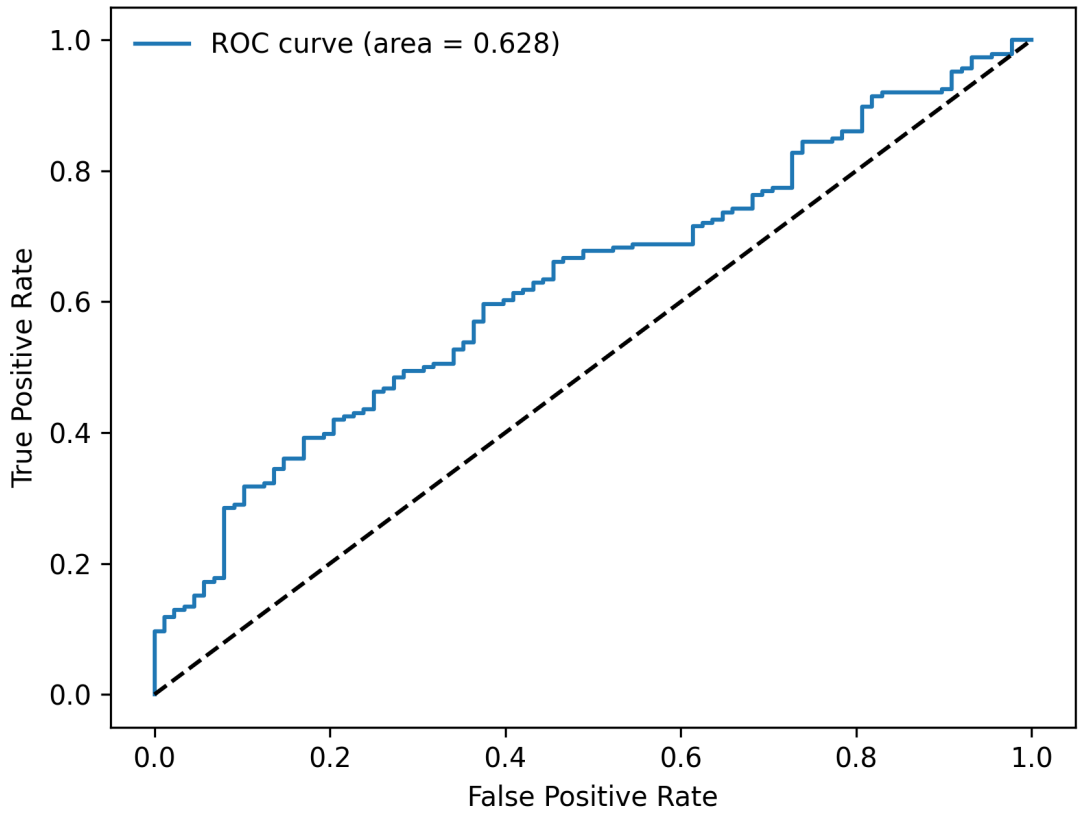


Figure 6: ROC Curve for Model 2

The ROC-AUC Score was thus determined to be 0.628, along with a test classification accuracy of 0.894. The model was then saved to `models/model_2.pt`.

3.1.3 Model 3

Even though Model 2 had a higher classification accuracy, it had a lower ROC-AUC score. We decided to train another model with 3 convolutional layers, but using a different loss criterion `CrossEntropyLoss` instead of `NLLLoss`. The Kernels had the following parameters:

- Kernel 1: 3x3, 16 output channels
- Kernel 2: 3x3, 32 output channels
- Kernel 3: 3x3, 64 output channels
- dropout probability: 0.4

The fully connected layers had 128 and 2 output channels respectively. The dropout probability was set to 0.4. The model was trained for 50 epochs with a batch size of 32. The Loss Function used was `CrossEntropyLoss` and the optimizer used was `Adam` with a learning rate of 0.001. The results were as follows:

- training loss on final epoch: 0.076
- training accuracy on final epoch: 0.973

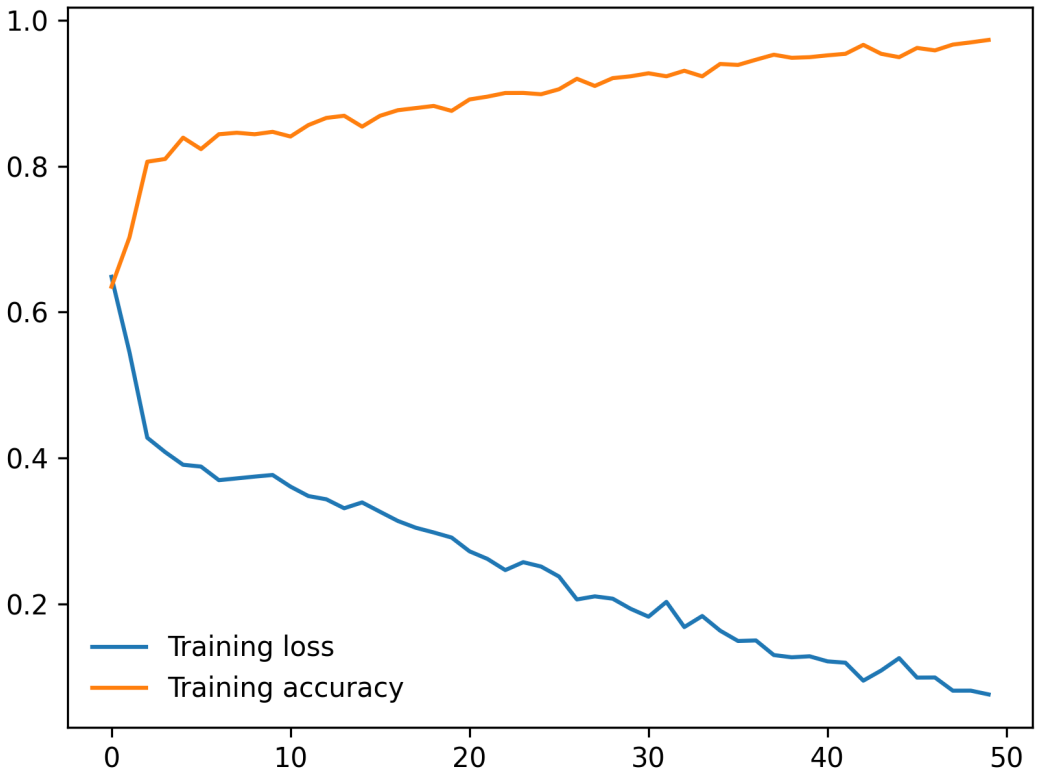


Figure 7: Training Loss and Accuracy for Model 3

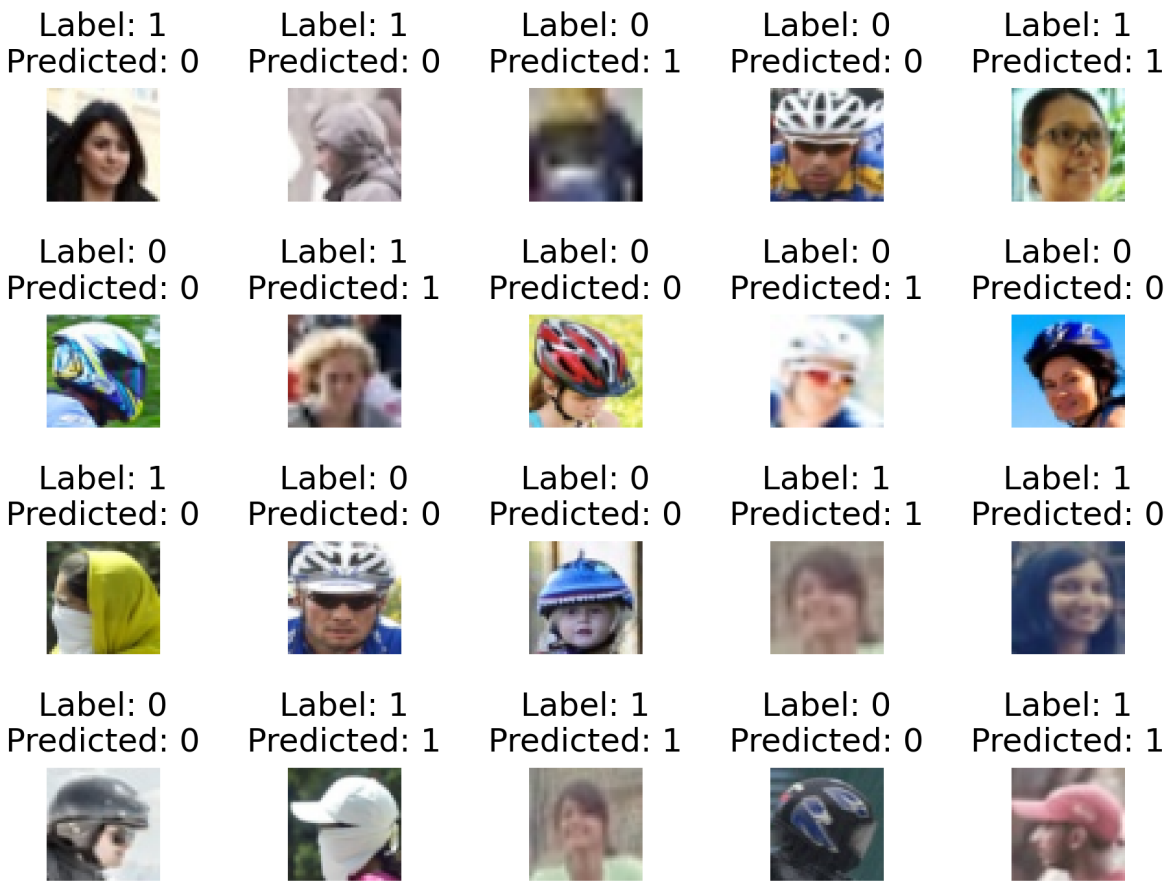


Figure 8: 20 Random Sample Predictions from Model 3

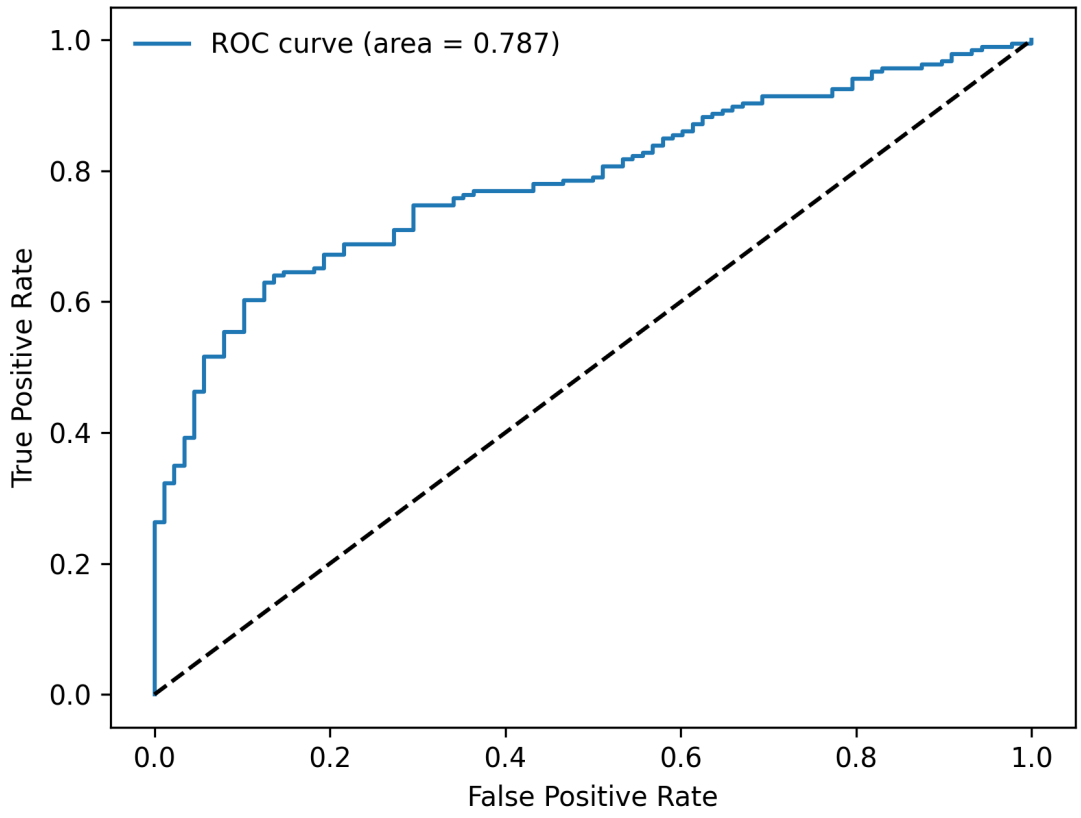


Figure 9: ROC Curve for Model 3

The ROC-AUC Score was thus determined to be 0.787, along with a test classification accuracy of 0.893. The model was then saved to `models/model_3.pt`.

3.1.4 Model 4

Since we were able to improve the performance of model 2 by changing the loss criterion, we decided to attempt the same with model 1. We trained a CNN with 2 convolutional layers and loss criterion `CrossEntropyLoss`. The Kernels had the following parameters:

- Kernel 1: 3x3, 16 output channels
- Kernel 2: 3x3, 32 output channels
- dropout probability: 0.4

The fully connected layers had 128 and 2 output channels respectively. The dropout probability was set to 0.4. The model was trained for 50 epochs with a batch size of 32. The Loss Function used was `CrossEntropyLoss` and the optimizer used was `Adam` with a learning rate of 0.001. The results were as follows:

- training loss on final epoch: 0.067
- training accuracy on final epoch: 0.978

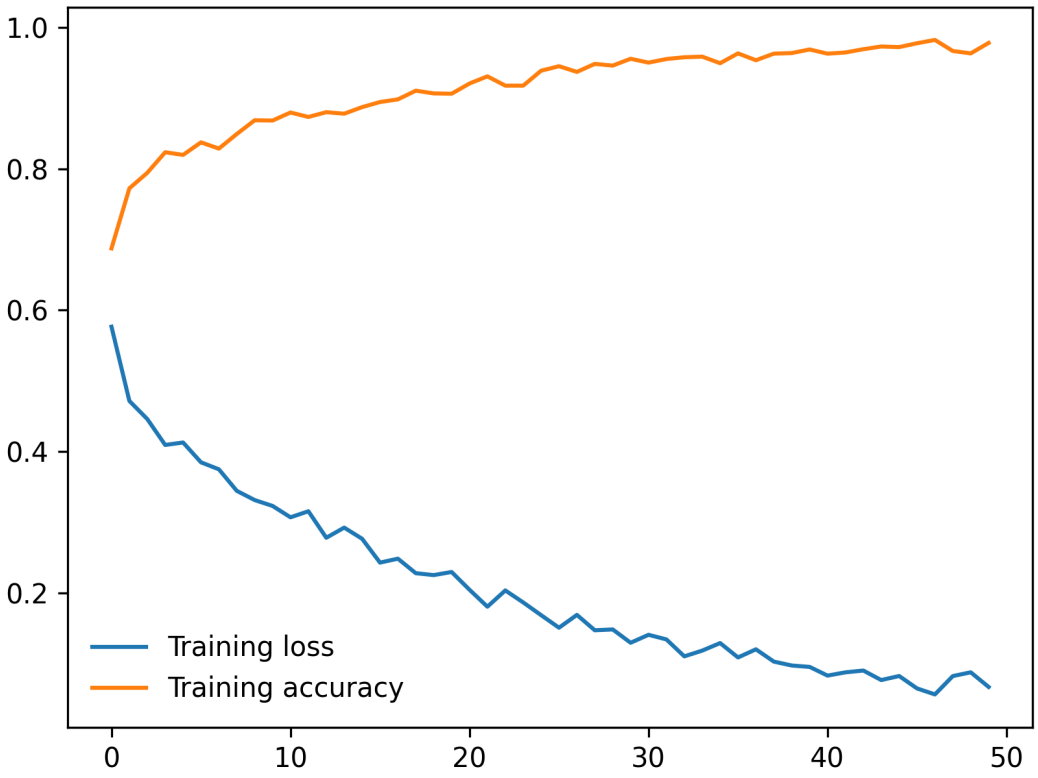


Figure 10: Training Loss and Accuracy for Model 4

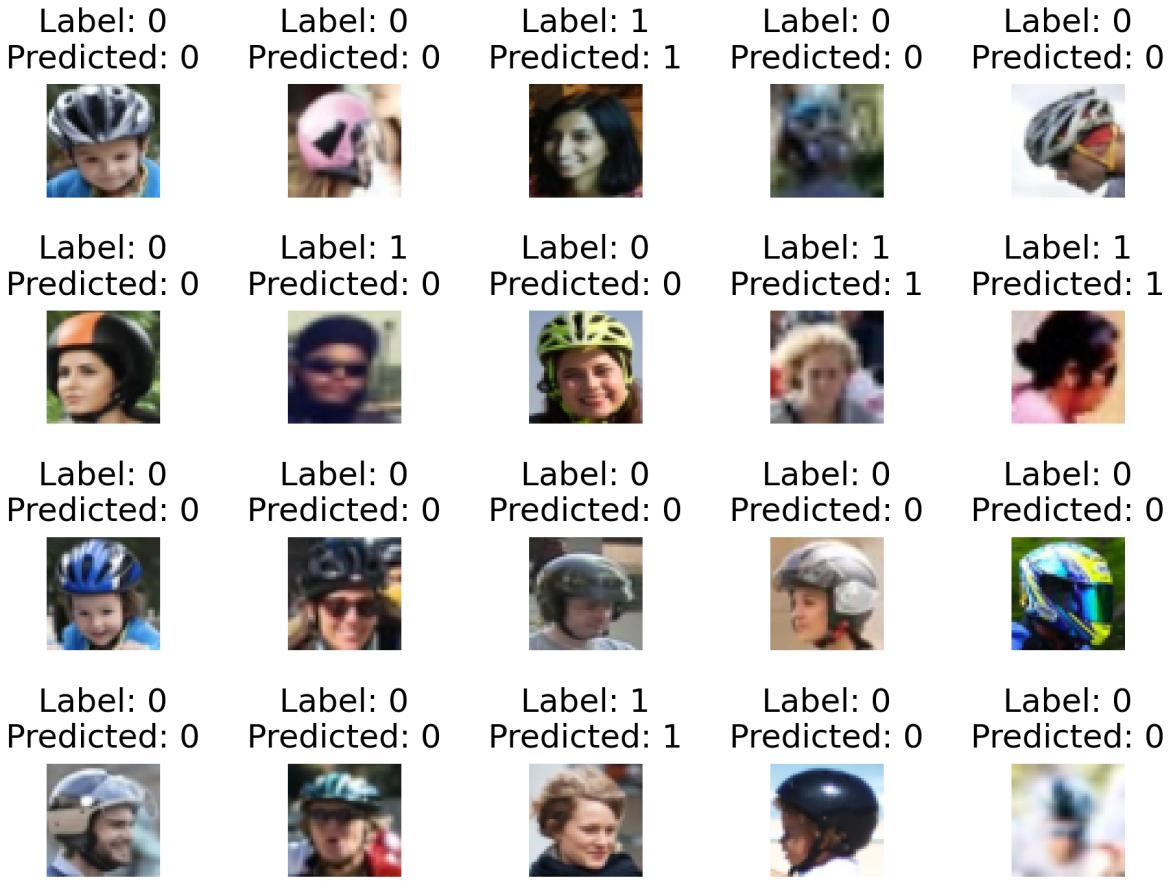


Figure 11: 20 Random Sample Predictions from Model 4

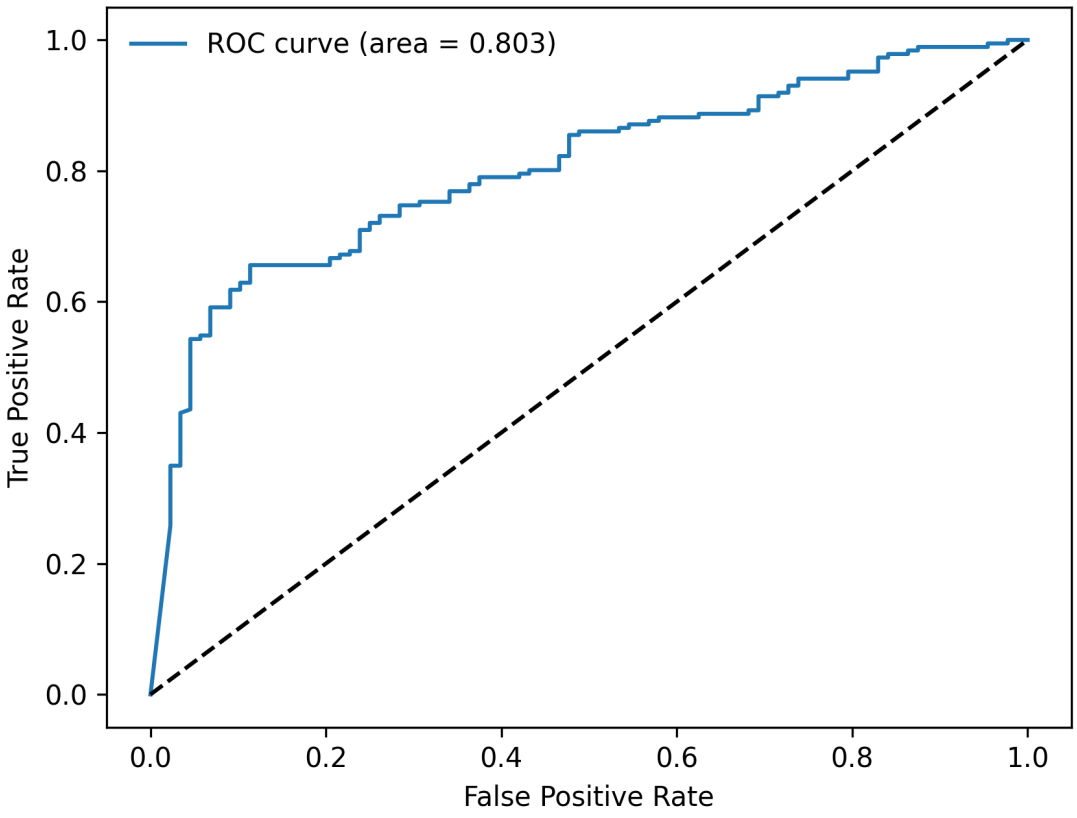


Figure 12: ROC Curve for Model 4

The ROC-AUC Score was thus determined to be 0.803, along with a test classification accuracy of 0.880. The model was then saved to `models/model_4.pt`.

4. Results

After several experiments, we determined that Model 1 performed the best with a ROC-AUC score of 0.812. The architecture of Model 1 is shown below:

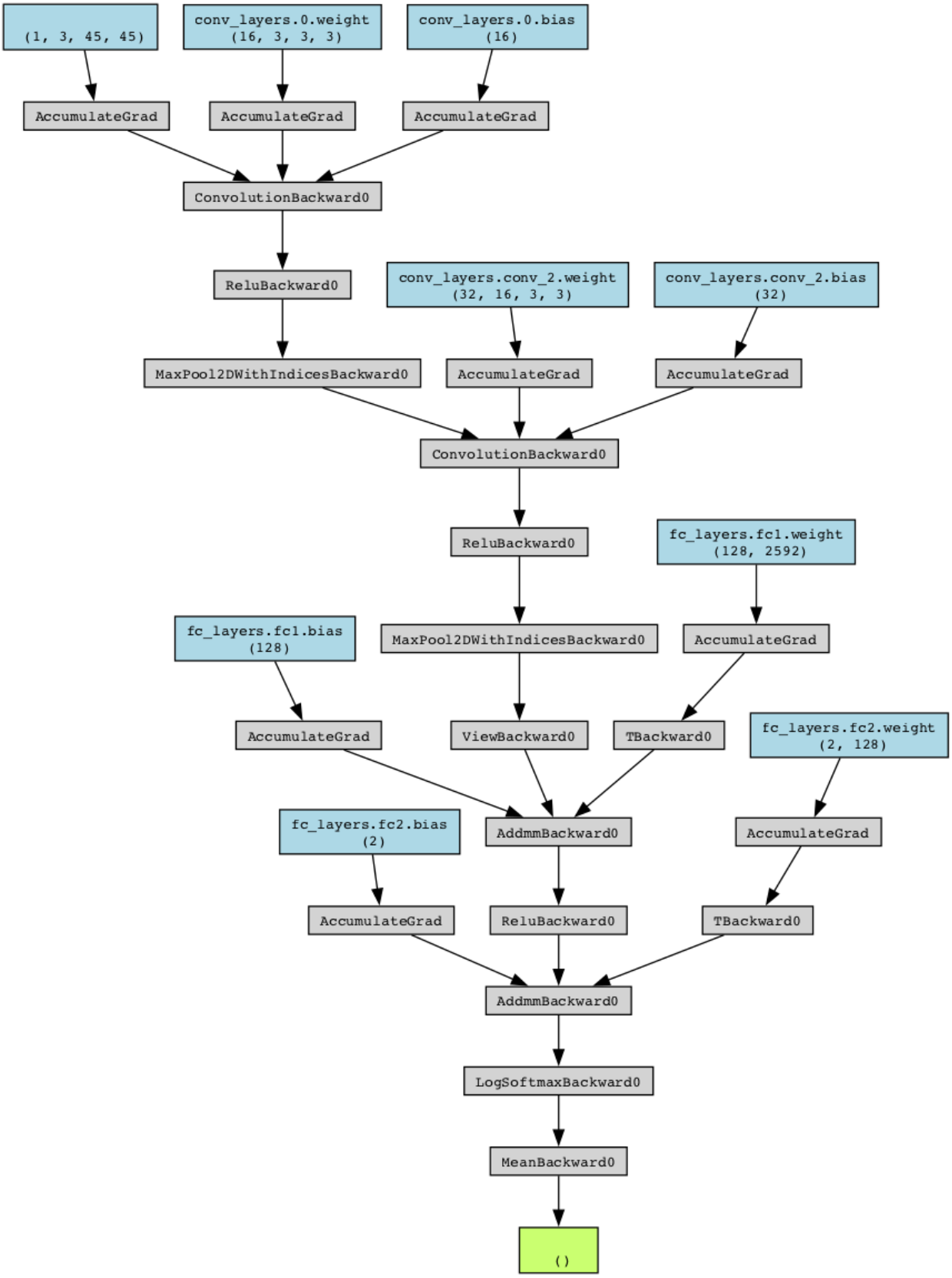
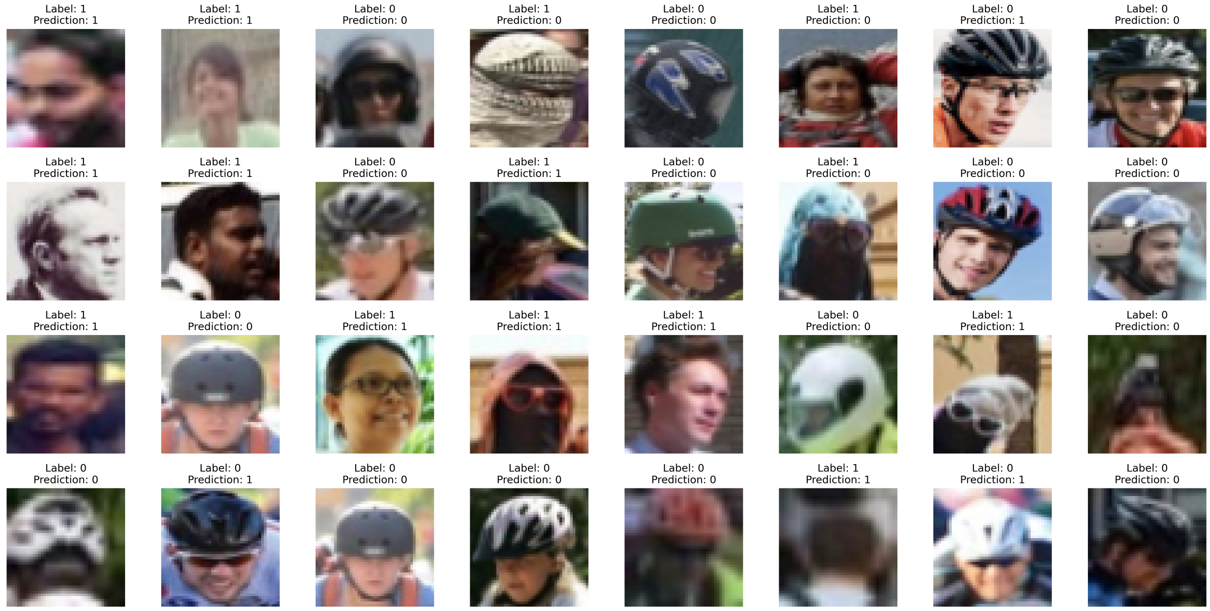
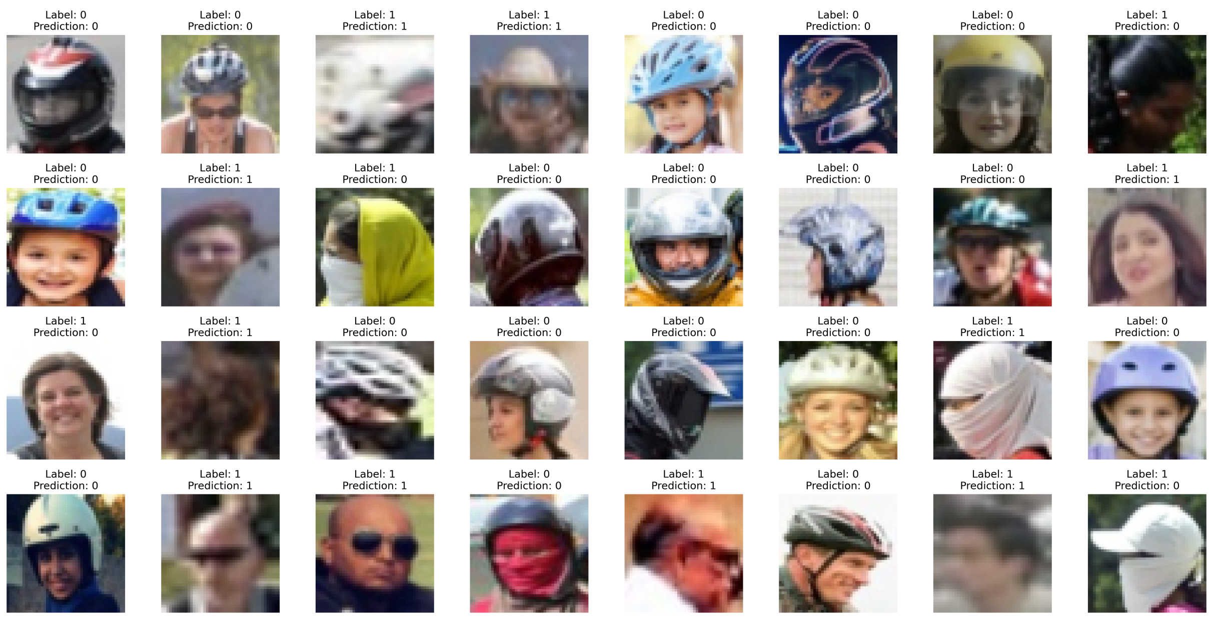


Figure 13: Architecture of Model 1

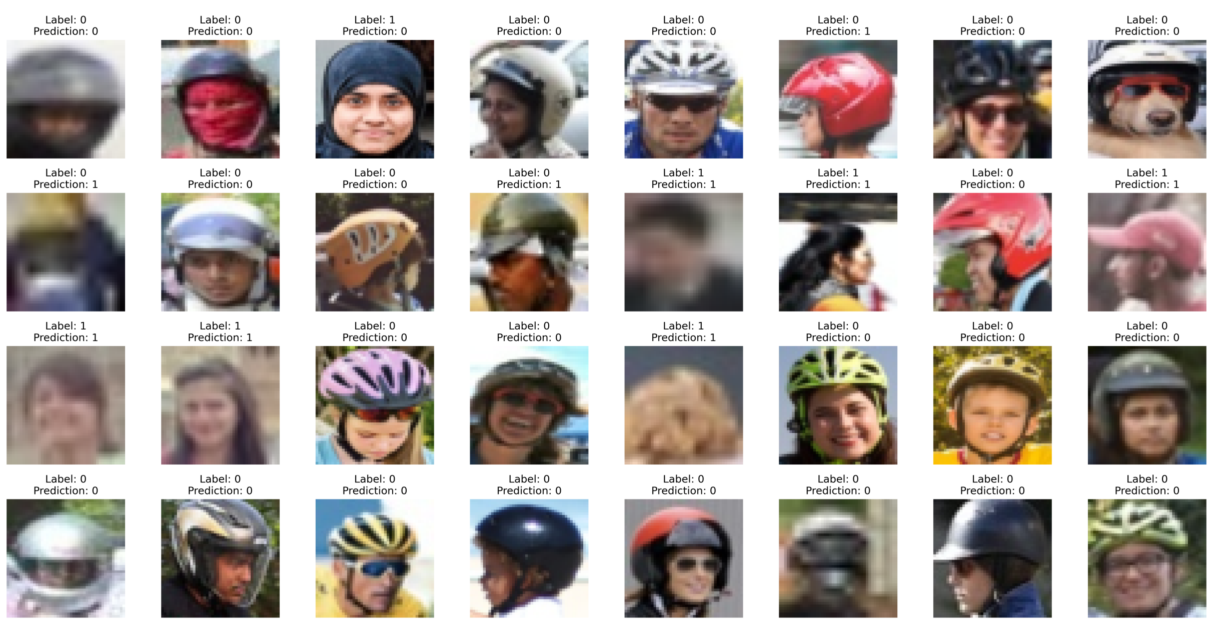
The results for each of the cropped images along with the true and predicted labels are shown below:



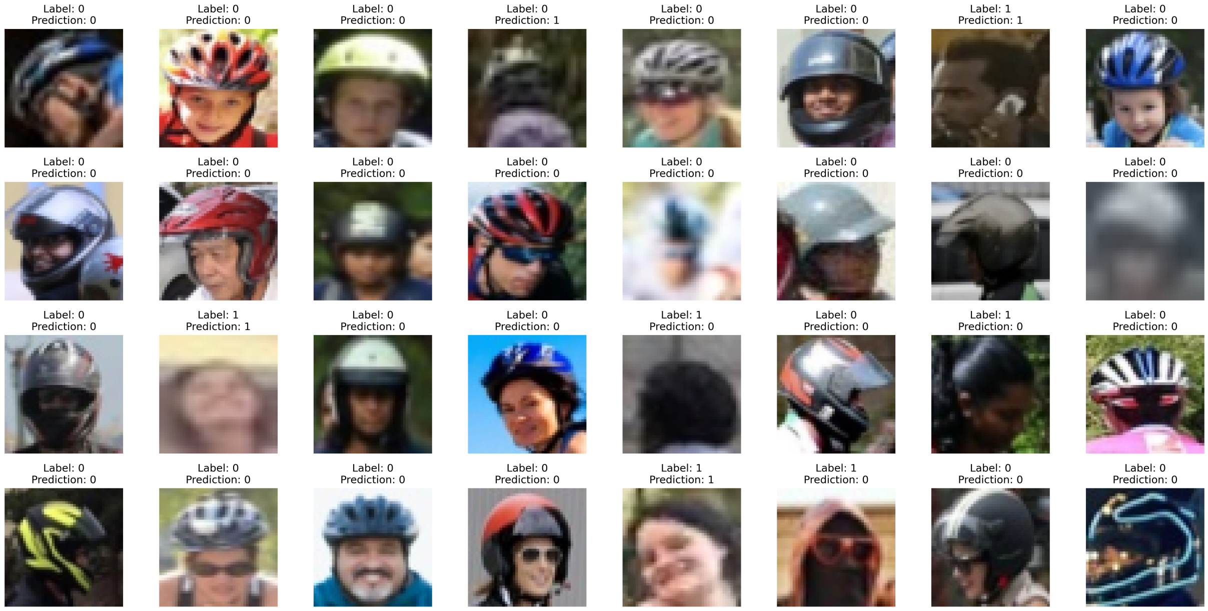
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 0
Label: 0
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 1
Label: 0
Prediction: 0

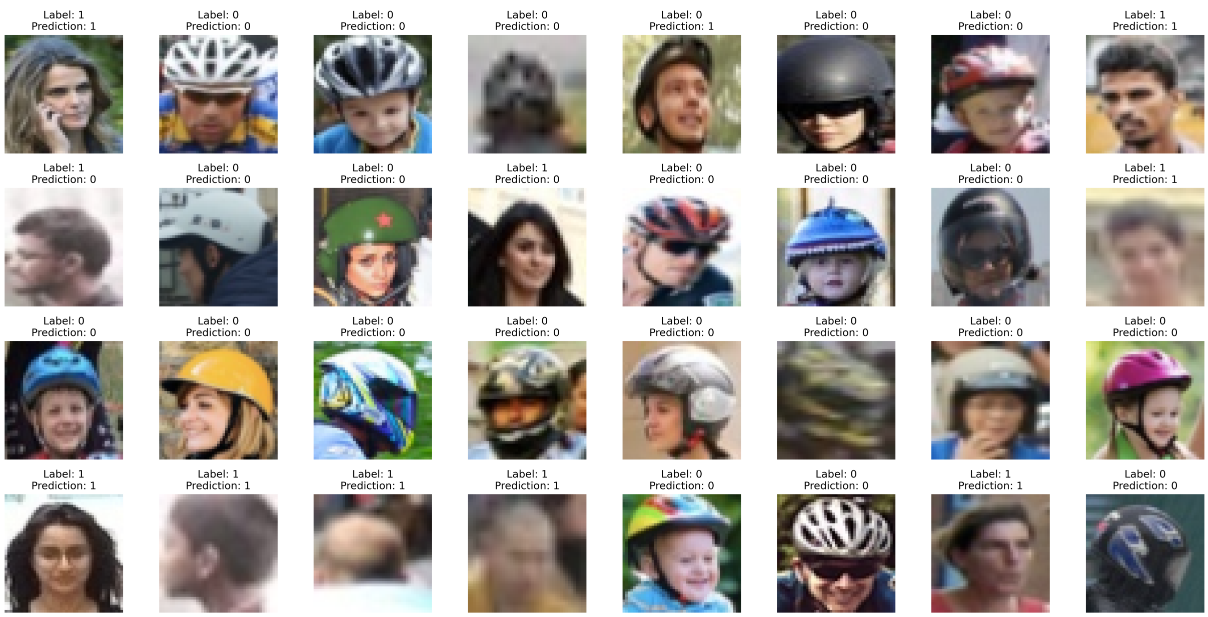


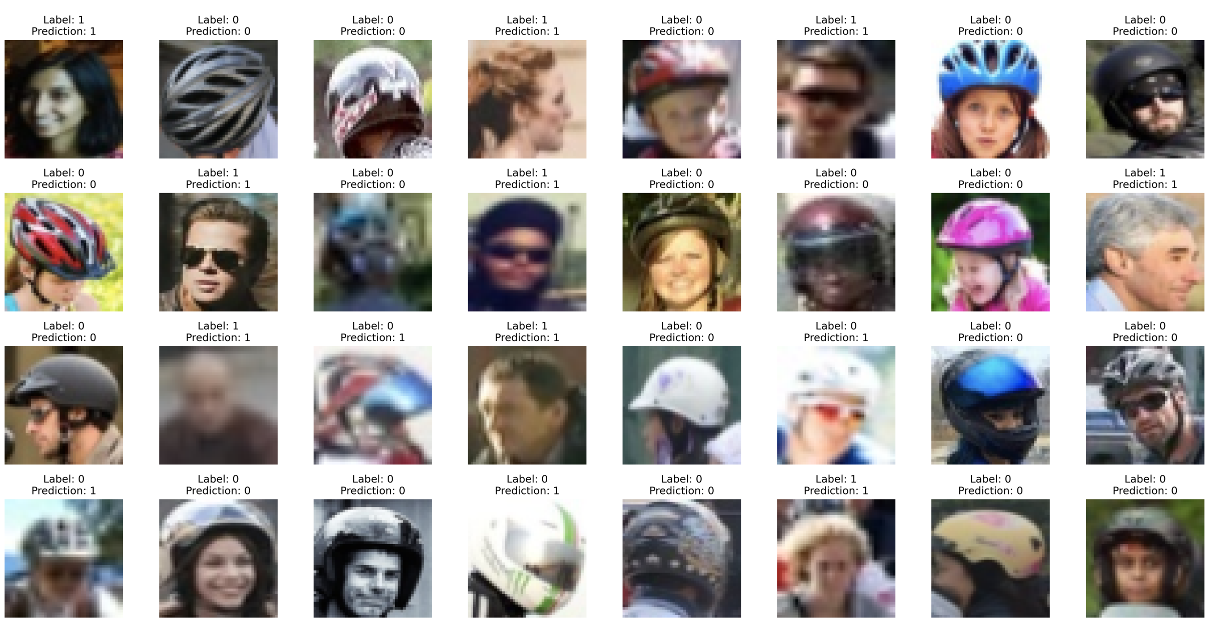
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 0

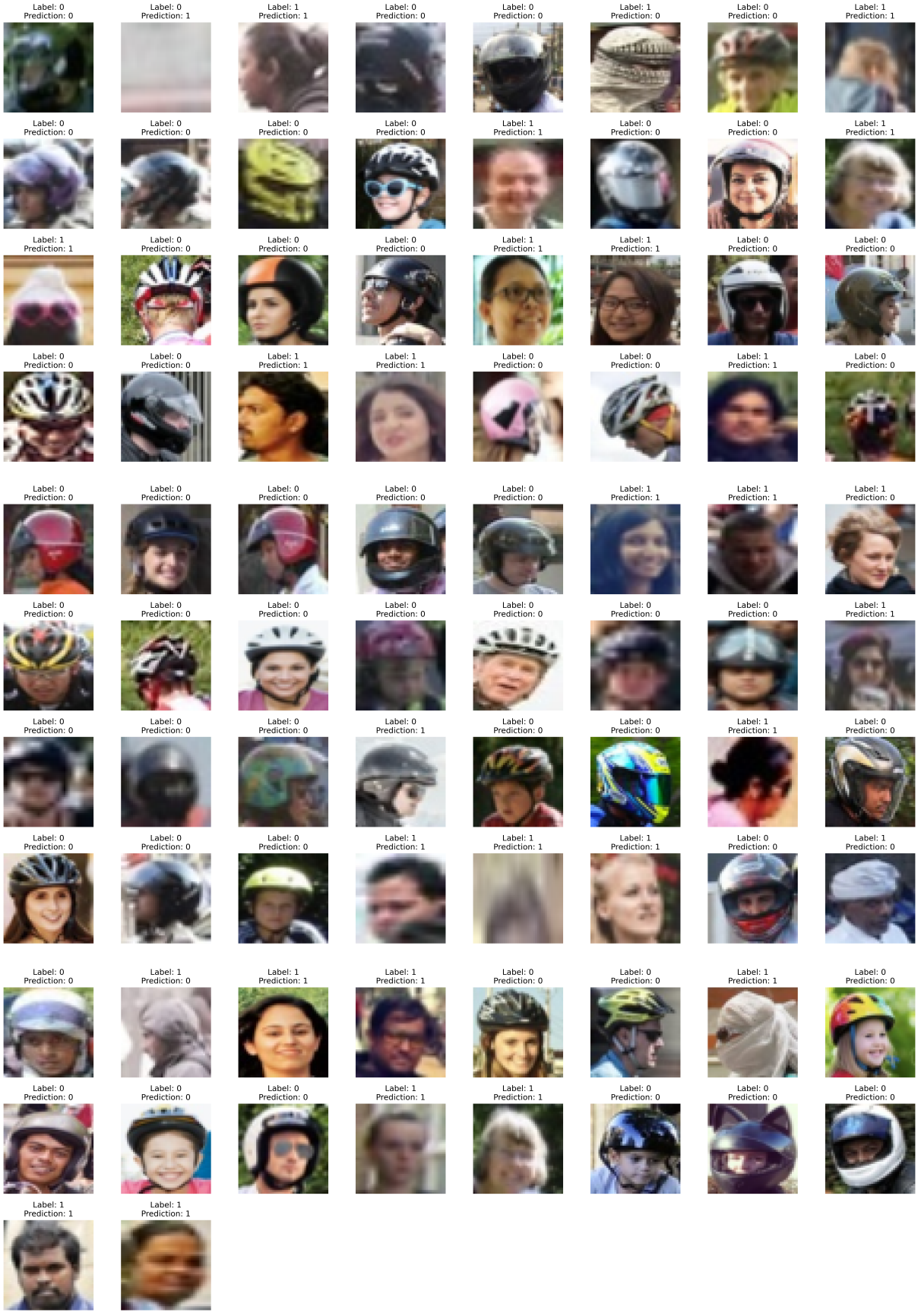


Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 1
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 1
Prediction: 1
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0
Label: 0
Prediction: 0


Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 1 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0


Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 1 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 1 Prediction: 1
Label: 1 Prediction: 1
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0


Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 1
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0
Label: 0 Prediction: 1
Label: 0 Prediction: 0
Label: 1 Prediction: 1
Label: 0 Prediction: 0
Label: 0 Prediction: 0



Figures 14-22: Predictions for each cropped image in the randomlly selected test set.

From an analysis of the predictions above as well as the results noted in Section 3.1.1. We have been able to achieve a ROC-AUC score of 0.812 and a test classification accuracy of 0.884. We can also see from the images above that the model is generally good at accurately labeling any images with a person not wearing a helmet except when the person in the image is wearing some form of head covering. This needs to be improved, however, it is difficult to do so without a larger dataset.

The biggest hurdle in the implementation of this project was the small dataset size which significantly limited the accuracy we could hope to achieve.