Binary Classification of Handwritten Digits Using Perceptron Models

Alhim A. Vera[1] and Ali A. Minai[2]

[1,2]Department of Electrical and Computer Engineering and Computer Science
University of Cincinnati, Cincinnati, OH 45221, USA
[1]veragoaa@mail.uc.edu
[2]ali.minai@uc.edu

November 1, 2024

Abstract

This project explores the application of single-layer perceptrons for binary classification on handwritten digits using the MNIST dataset. In the first problem, we trained a perceptron to distinguish between binary outcomes, optimizing the learning rate and bias w_0 to reach rapid convergence. The second problem extended this approach by training ten separate perceptrons to classify each digit from 0 to 9, addressing class imbalance through oversampling. Results showed that simpler digits, such as 0 and 1, achieved lower error rates and higher accuracy, while more complex digits, like 8 and 9, demonstrated higher error rates due to visual similarity with other digits. The study highlights that single-layer perceptrons can effectively classify distinct digits but face challenges with complex shapes, suggesting potential improvements with multi-layer perceptrons for better generalization on visually intricate digits, code available: https://github.com/AdonaiVera/neuro-eval-lab

Keywords: Perceptron, Binary Classification, MNIST Dataset, Handwritten Digit Recognition

1 Problem 1

In this project, we explored the implementation and evaluation of a single-layer perceptron model for binary classification, focusing on distinguishing between handwritten digits. The problem is framed as a supervised learning task, utilizing a subset of the MNIST dataset, which contains grayscale images of handwritten digits. Our objective was to train a perceptron to classify images into two classes, representing binary outcomes, and then evaluate its performance on unseen data.

1.1 System Specification

We initially set the number of epochs to 100, incorporating a stopping criterion to terminate training early if the error fraction fell below 0.01. This allowed for flexibility in case the model required multiple passes. The learning rate (η) was carefully tuned: we began with a small value of 0.001, but it did not yield optimal results in the evaluation metrics. Through iterative adjustments, a final learning rate of 0.01 was selected, which produced the best performance metrics. With this configuration, the model achieved adequate results within just two epochs, as the problem complexity was low and allowed rapid convergence.

1.2 Results

To optimize the bias w_0, we initially tested increments of 0.5, but observed minimal variation in the metrics. This led us to adjust w_0 in steps of 10 to better visualize the impact. Ultimately, the ROC curve confirmed that the initial calculated w_0 provided the best performance, balancing precision and recall effectively.

The results below demonstrate the performance and evaluation metrics of the perceptron model across training, bias optimization, and classification on the challenge set.

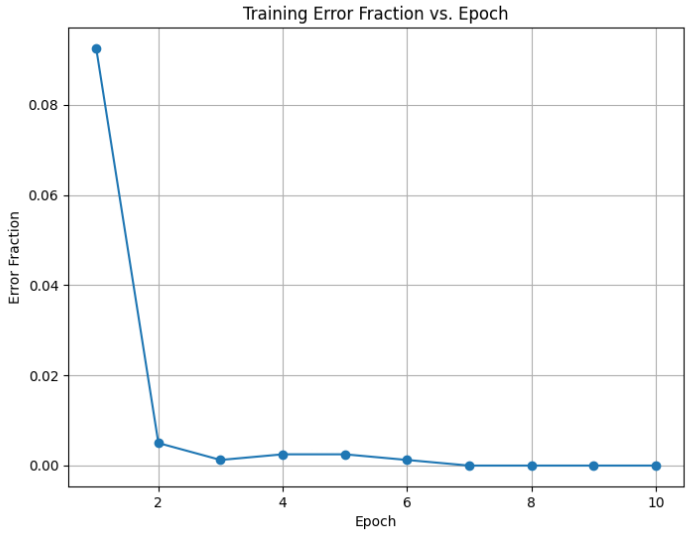

Figure 1: Training Error Fraction vs. Epoch. The error fraction on the training set across epochs indicates rapid convergence, with the model reaching minimal error in only a few epochs.

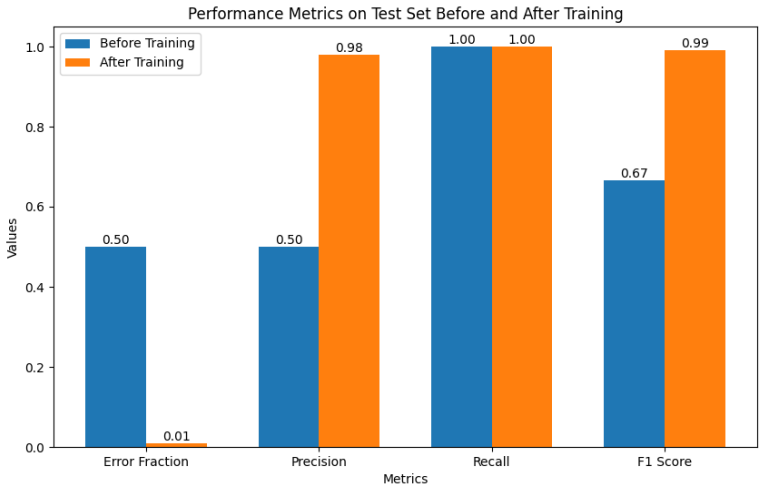

Figure 2: Performance Metrics Before and After Training. A comparison of error fraction, precision, recall, and F1 score on the test set before and after training, highlighting the improvement in model performance.

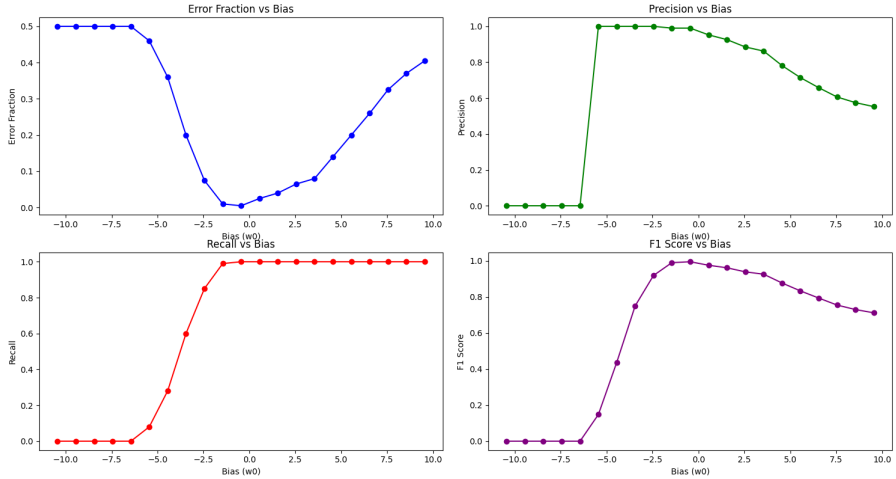

Figure 3: Performance Metrics vs. Bias (w_0). Error fraction, precision, recall, and F1 score are displayed for different bias values. Through tuning, the optimal w_0 was found to balance precision and recall, enhancing F1 score.

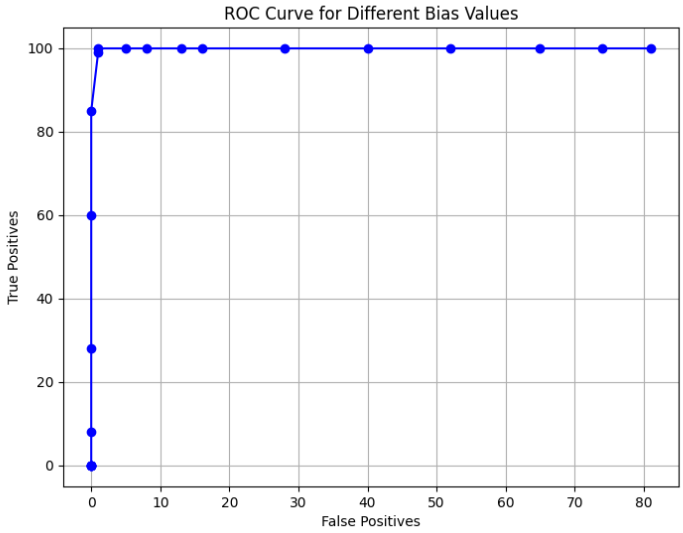

Figure 4: ROC Curve for Different Bias Values. True positives vs. false positives for varying bias values, helping to identify the optimal w_0 based on ROC analysis.

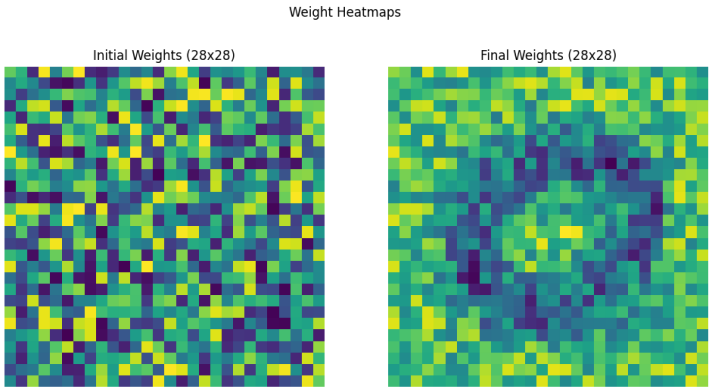

Figure 5: Initial and Final Weight Heatmaps. Visual comparison of the perceptron's weights before and after training, showing how the weights were adjusted through training to improve classification performance.

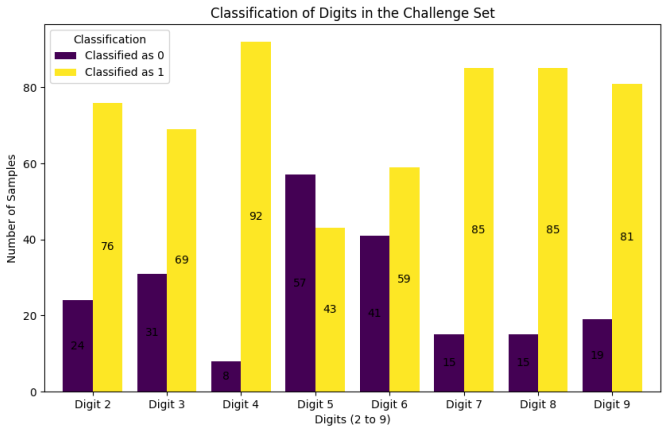

Figure 6: Classification of Digits in the Challenge Set. This plot illustrates the number of samples for each digit (2-9) classified as 0 or 1, demonstrating the model's generalization capabilities on digits outside the training set.

1.3 Analysis of Results

The results reveal that the perceptron model effectively classifies digits from the challenge set with some notable trends, as shown in Figure 6. Digits such as 4, 7, 8, and 9 have a high likelihood of being classified as 1, with over 80 samples in each case classified as 1. This could be due to structural features in these digits that resemble patterns learned from training samples labeled as 1. While digits like 2, 3, 5, and 6 are still predominantly classified as 1, they have a relatively higher proportion of samples classified as 0 compared to other digits. For example, digits 5 and 6 show approximately half of the samples classified as 0, indicating that the model struggles to make confident classifications for these digits. The perceptron model, being a single-layer network, is limited in capturing complex, non-linear patterns, leading to certain digits being more easily misclassified. Overall, this analysis underscores the model's limitations in generalization beyond binary classification, as it relies on linear boundaries that do not adequately separate the patterns of digits outside the training set.

2 Problem 2

In this second part of the homework, we implemented a binary classification framework using separate perceptron models to identify each digit from 0 to 9 within the MNIST dataset. Each perceptron was tasked with distinguishing a single digit, outputting 1 when recognizing its target digit and 0 for all others. This approach introduces class imbalance, as only 10% of the training samples are labeled 1 for each perceptron, while the remaining 90% are labeled 0. To address this imbalance, we experimented with both oversampling and undersampling strategies to ensure each perceptron received sufficient training examples for the minority class. Our goal was to train each perceptron model effectively, achieving low error rates while balancing precision, recall, and F1 scores across all classes.

2.1 System Specification

Each perceptron was initialized with random weights and trained individually to recognize its target digit within a binary classification framework. Due to the inherent class imbalance in each perceptron's training set (10% target digit and 90% non-target digits), we applied an oversampling strategy to increase the representation of the minority class in each epoch. After experimenting with various learning rates, a rate of 0.01 was found to provide stable and efficient convergence across all perceptrons. To establish the optimal number of epochs, we trained the perceptron for the digit 9—identified as the most complex to classify—until an acceptable error level was reached. After this evaluation, 8 epochs were determined to yield the best results, achieving a balanced accuracy error below 0.05. This epoch count and configuration were applied consistently across all perceptrons to ensure uniformity in training.

2.2 Results

In this section, we present the performance of each perceptron model in classifying the digits 0 through 9. Each perceptron was trained with 8 epochs, as determined optimal through experimentation with the perceptron for digit 9. The figures below show the Balanced Accuracy Error, Precision, Recall, and F1 scores before and after training, as well as the error reduction across epochs.

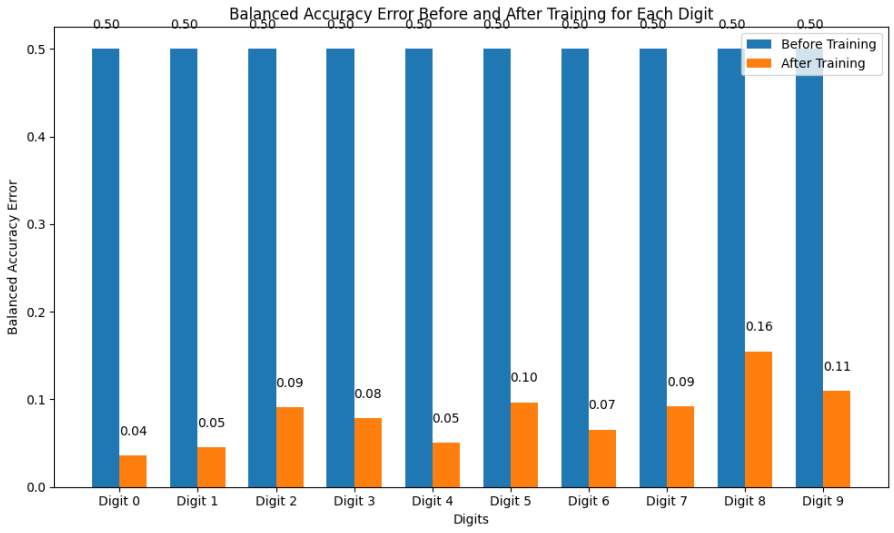

Figure 7: Balanced Accuracy Error Before and After Training for Each Digit. The decrease in balanced accuracy error demonstrates the effective learning achieved by each perceptron across the 8 epochs.

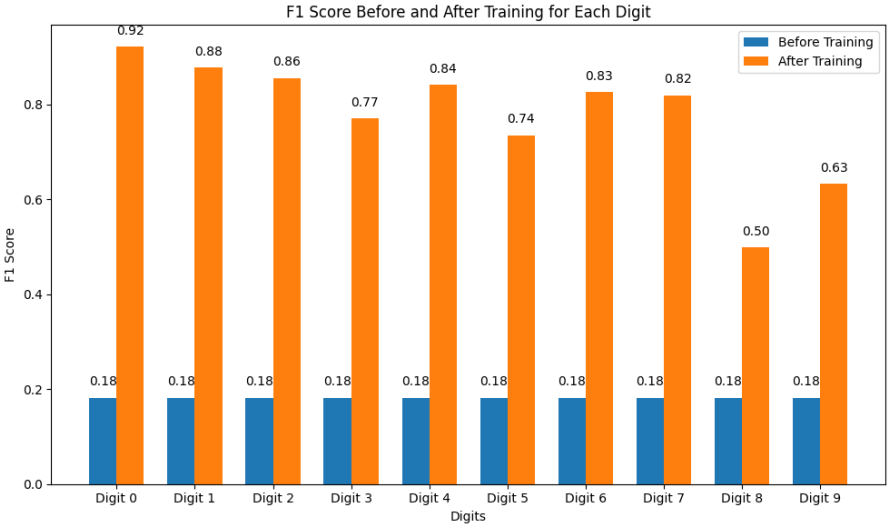

Figure 8: F1 Score Before and After Training for Each Digit. The substantial improvement in F1 scores indicates better performance in handling the unbalanced class distribution for each perceptron.

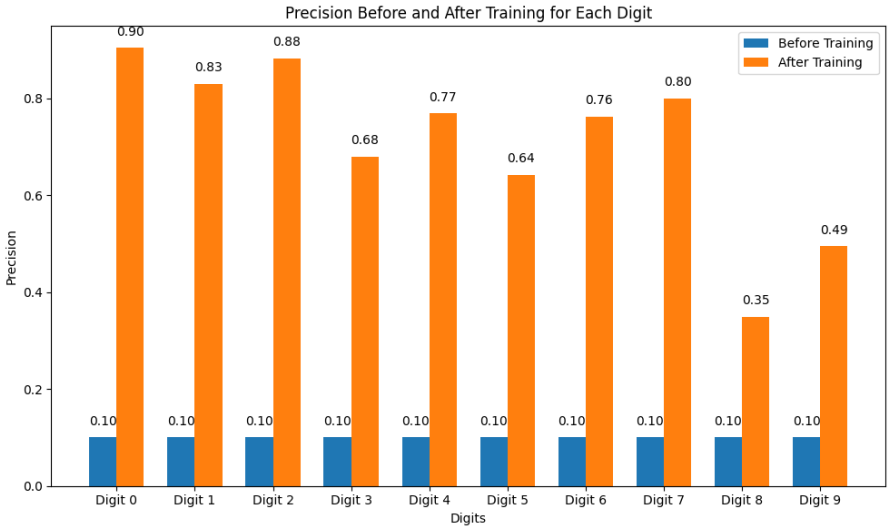

Figure 9: Precision Before and After Training for Each Digit. After training, precision values improved for each digit, indicating a reduction in false positives.

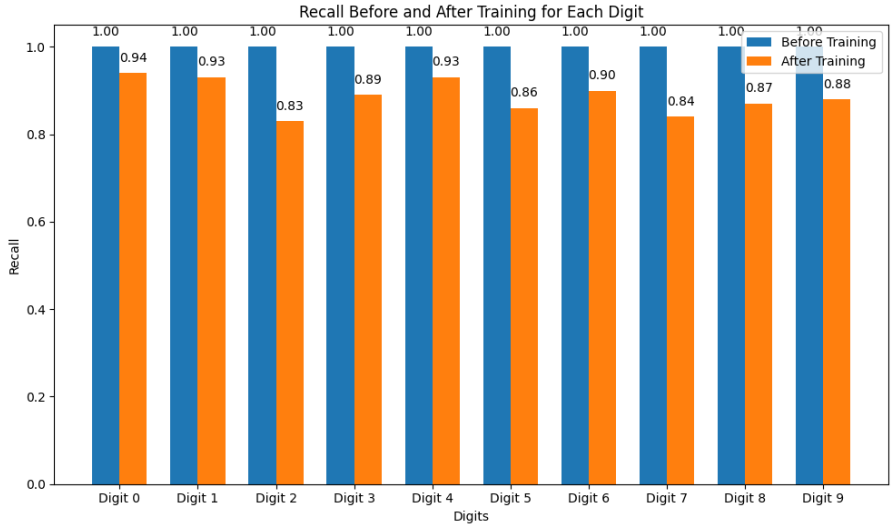

Figure 10: Recall Before and After Training for Each Digit. The recall improvement post-training demonstrates a reduction in false negatives across the models.

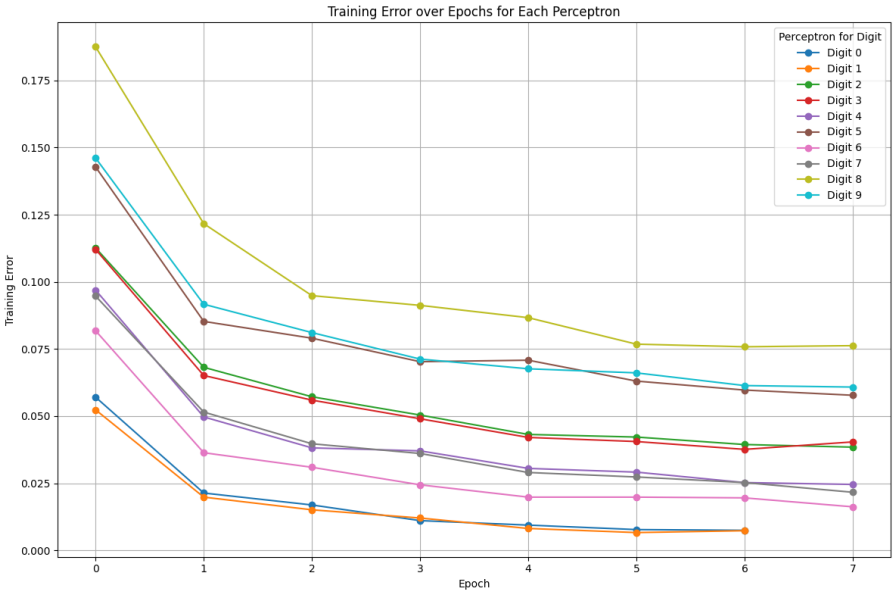

Figure 11: Training Error Reduction Across Epochs for Each Perceptron. This plot shows the error fraction at each epoch, illustrating the learning process for each digit-specific perceptron.

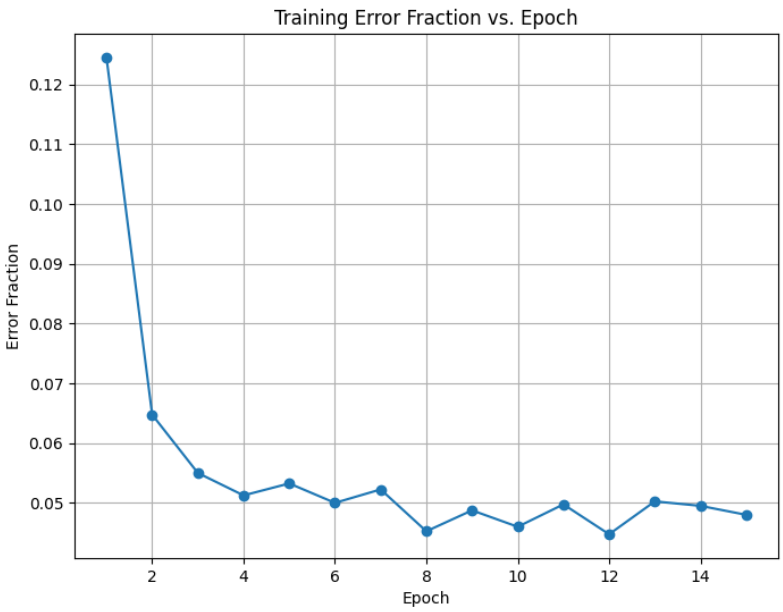

Figure 12: Training Error Fraction Over Epochs for Digit 9 Perceptron. This separate figure for the digit 9 perceptron shows the gradual reduction in error, validating the choice of 8 epochs for optimal performance.

2.3 Analysis of Results

The results indicate that the perceptrons performed with varying degrees of success across the different digits. In general, digits that are visually distinct and less complex, such as 0, 1, and 6, achieved lower error rates and higher accuracy metrics after training. This suggests that the perceptron models were better able to distinguish these digits due to their simpler and more unique shapes, which likely reduced confusion with other classes.

Conversely, digits like 8 and 9 showed relatively higher error rates even after training, with balanced accuracy errors that decreased gradually but remained above those of simpler digits. The perceptrons for these digits converged more slowly and required more epochs to achieve significant error reduction, as seen in the training error reduction plot. This outcome is likely due to the inherent visual similarity of these digits to other numbers (for example, 8 shares similar circular features with 3, and 9 resembles 4 in certain positions), making it harder for the perceptron models to create distinct boundaries between these classes.

The balanced accuracy error plot (Figure 7) highlights the substantial improvement for each perceptron, with initial errors starting around 0.50 (as expected for an untrained binary classifier with random initialization) and converging to values between 0.02 and 0.16. Digits like 2, 5, 8, and 9 exhibited the highest final errors, likely due to their shape complexity and resemblance to other digits, which increases the difficulty of distinguishing them in a binary classification setup.

The F1 Score plot (Figure 8) reflects this trend, showing that digits such as 0 and 1 have the highest F1 scores post-training, while digits like 8 and 9 scored lower. The precision (Figure 9) and recall (Figure 10) metrics further reinforce this pattern, indicating that while recall remained relatively high across digits, precision was particularly challenging to optimize for more complex digits.

The training error over epochs for each digit (Figure 11) illustrates that all perceptrons showed a large amount of learning, with a consistent decline in error across epochs. Although the error rates did not converge to zero for more complex digits, the general reduction demonstrates effective learning. Overall, the perceptron models performed well for simpler digits but struggled with visually complex or similar digits due to the limitations of linear decision boundaries in single-layer perceptions.