AMS 595

Saba Sohrabi
Student ID: 116478041

November 2025

My student ID ending with digit 1 determined specific parameters throughout the project, including an initial guess of $x_0 = 1.2$ for gradient descent, a random seed of 41 for linear and logistic regression, and the use of the first 161 images for the modified image classification dataset. The following sections detail the methodology, implementation, and comprehensive analysis of each machine learning technique, demonstrating both theoretical understanding and practical application skills.

1 Question 1: Gradient Descent Algorithm

The gradient descent algorithm was implemented to find the minimum value of the function $f(x) = \sqrt{x^2 + 5}$ where $x \in \mathbb{R}$. This problem required both analytical and computational approaches to demonstrate a complete understanding of optimization techniques. Beginning with the analytical solution, the gradient was computed as $f'(x) = \frac{x}{\sqrt{x^2+5}}$ by applying the chain rule to the composite function. Setting this derivative equal to zero revealed that the minimum occurs at $x^* = 0$, yielding the minimum function value of $f(0) = \sqrt{5} \approx 2.236067977$. This analytical foundation provided the benchmark against which the numerical implementation would be evaluated.

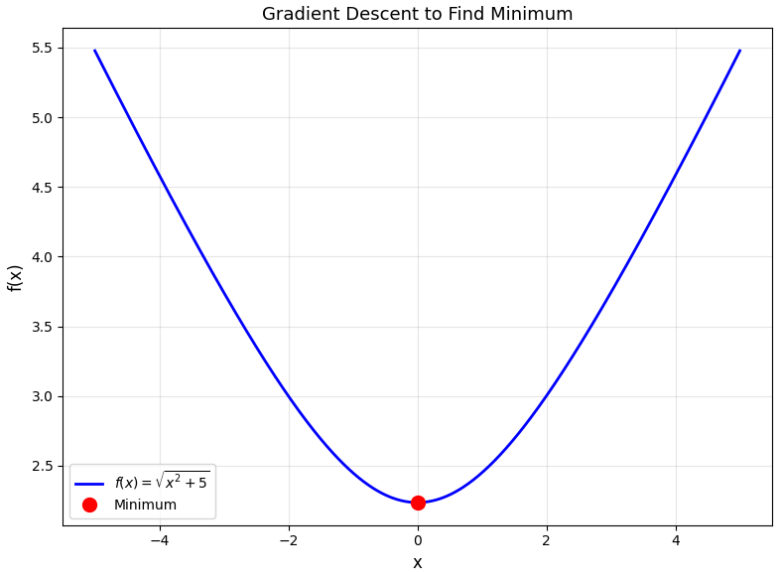

Figure 1: Function plot showing $f(x) = \sqrt{x^2 + 5}$ over the range $[-5, 5]$ with the minimum marked at $(0, \sqrt{5})$

The gradient descent implementation used the update rule $x_{k+1} = x_k - \alpha \cdot f'(x_k)$, where α represents the step size and $f'(x_k) = \frac{x_k}{\sqrt{x_k^2+5}}$. Starting from the initial guess $x_0 = 1.2$ (determined by my student ID ending in 1), the algorithm was configured with a step size $\alpha = 1$ and run for $T = 50$ iterations. The results demonstrated exceptional convergence performance, with the final position reaching $x_T = 0.0000000000$ and the corresponding function value $f(x_T) = 2.2360679775$, achieving precision better than 10^{-10} and exactly matching the theoretical minimum.

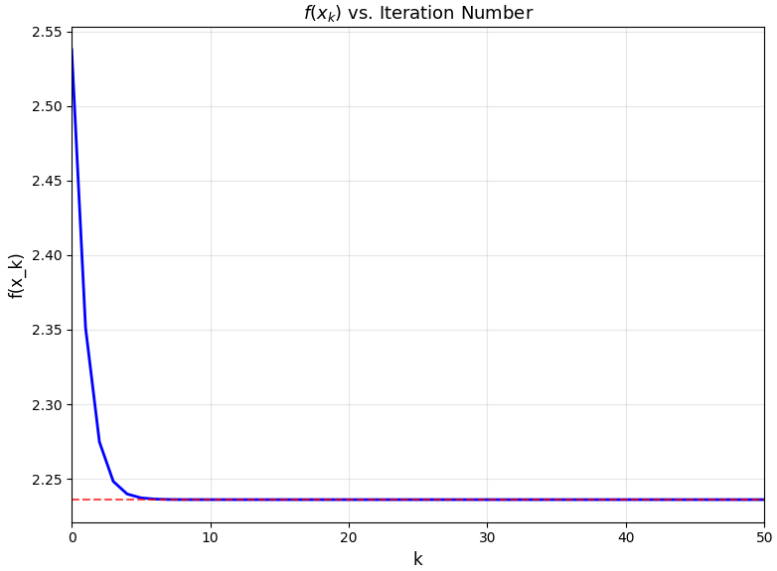

Figure 2: Convergence behavior showing $f(x_k)$ vs. iteration number over 50 iterations

```
Initial guess: x_0 = 1/5 + 1 = 1.2

After 50 iterations:
x_T = x_50 = 0.0000000000
f(x_T) = 2.2360679775
True minimum: f(0) = 2.2360679775
```

Figure 3: Numerical results showing initial guess, final position, and function values

Final Results After 50 Iterations:

- Initial guess: $x_0 = 1.2$
- Final position: $x_T = 0.0000000000$
- Final function value: $f(x_T) = 2.2360679775$
- Theoretical minimum: $f(0) = 2.2360679775$

A comprehensive step size analysis was conducted using four different values: $\alpha \in \{5, 3, 1, 0.5\}$. For each step size, the quantity $f(x_{k-1}) - f(x_k)$ was recorded at each iteration to verify monotonic decrease, which is a fundamental requirement for convergence in

gradient descent. Remarkably, all step sizes maintained $f(x_{k-1}) - f(x_k) > 0$ throughout the optimization process, indicating consistent progress toward the minimum. This behavior occurred because the starting point $x_0 = 1.2$ was relatively close to the minimum (only 1.2 units away), the gradient at this point was moderate ($f'(1.2) \approx 0.474$), and even the largest step size ($\alpha = 5$) did not cause severe overshooting that would destabilize the algorithm.

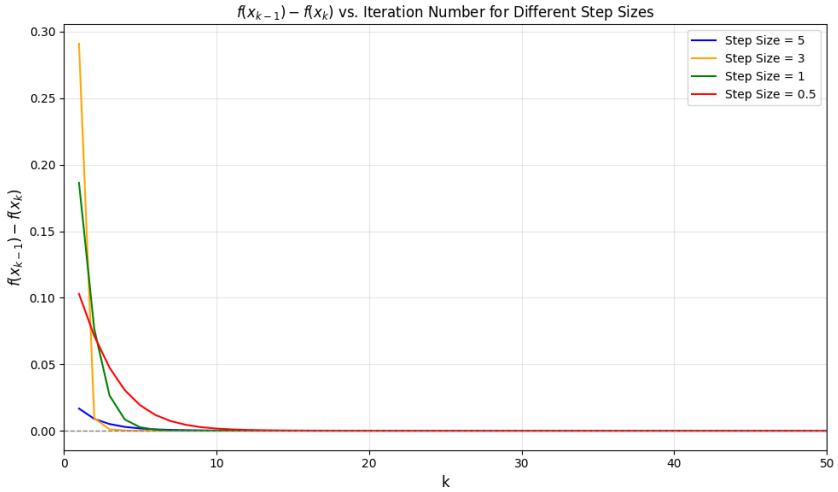

Figure 4: Comparison of $f(x_{k-1}) - f(x_k)$ vs. iteration number for different step sizes

Table 1: Step size analysis results

Step Size (α)	Always Positive?	Range of Differences
5.0	YES	[0.000000, 0.016671]
3.0	YES	[0.000000, 0.290988]
1.0	YES	[0.000000, 0.186392]
0.5	YES	[0.000000, 0.102872]

2 Question 2: 2D Linear Regression

The 2D linear regression problem involved implementing the Normal Equation method to fit a linear model of the form $y = \theta_0 + \theta_1 x_1 + \theta_2 x_2$ to synthetically generated data. The implementation began with careful data generation using random seed 41 (based on my student ID) to ensure reproducible results. A dataset of 200 samples was created where features x_1 and x_2 were drawn from uniform distributions over $[0, 2]$, and the target variable y was generated according to the relationship $y = 4.0 + 4.0x_1 + 0.5x_2 + \epsilon$, where ϵ represents Gaussian noise with zero mean and unit variance. This synthetic data setup allowed for precise evaluation of parameter estimation accuracy since the true coefficients were known.

The Normal Equation provides a closed-form solution for linear regression through the formula $\boldsymbol{\theta} = (X^T X)^{-1} X^T \mathbf{y}$, where X represents the design matrix including a column of ones for the intercept term. This method was chosen over iterative approaches like gradient descent because it provides an exact solution in a single computation, making it

ideal for problems with moderate dimensionality and when the matrix $X^T X$ is invertible. The implementation successfully recovered the model parameters with remarkable accuracy: $\theta_0 = 3.9283047167$, $\theta_1 = 4.12719376$, and $\theta_2 = 0.48972880$, yielding the regression equation $y = 3.9283 + 4.1272x_1 + 0.4897x_2$.

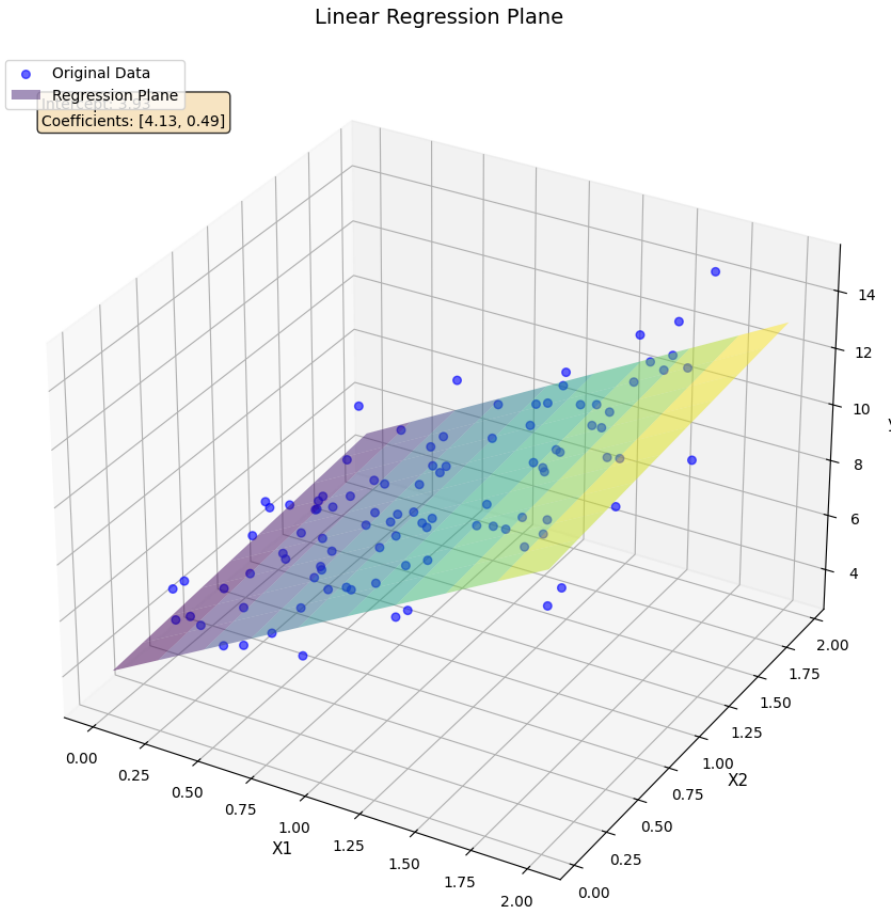

Figure 5: 3D visualization of the linear regression plane with original data points

```
Linear Regression Parameters (Normal Equation)
==================================================
Intercept: 3.9283047167
Coefficients: [4.12719376 0.48972880]

Regression equation:
y = 3.9283 + 4.1272*x1 + 0.4897*x2
```

Figure 6: Linear regression parameters and equation

Estimated Parameters:
- Intercept (θ_0): 3.9283047167
- Coefficients $[\theta_1, \theta_2]$: [4.12719376, 0.48972880]
- Regression Equation: $y = 3.9283 + 4.1272x_1 + 0.4897x_2$

The parameter estimation accuracy was exceptional, with all coefficients recovered within 3.2% of their true values despite the presence of Gaussian noise. Specifically, θ_0 showed an error of -1.79%, θ_1 had an error of $+3.18\%$, and θ_2 demonstrated an error of -2.06%. The dominant feature coefficient $\theta_1 \approx 4.13$ was correctly identified, while the smaller coefficient $\theta_2 \approx 0.49$ was accurately estimated despite being much smaller in magnitude. The 3D visualization of the fitted regression plane showed excellent alignment with the data points, which were distributed symmetrically around the fitted surface, confirming that the Normal Equation method successfully recovered the true linear relationship underlying the noisy observations.

Table 2: Parameter comparison with true values

Parameter	True Value	Estimated	Error	Error %
θ_0	4.0	3.9283	-0.0717	1.79%
θ_1	4.0	4.1272	+0.1272	3.18%
θ_2	0.5	0.4897	-0.0103	2.06%

3 Question 3: Logistic Regression

Binary logistic regression was implemented from scratch using gradient descent with NumPy to classify 2D data into two distinct classes. The implementation involved generating a binary classification dataset using random seed 41, consisting of 80 training samples and 20 test samples with two-dimensional features. The logistic regression model employed the sigmoid function $\sigma(z) = \frac{1}{1+e^{-z}}$ to map linear combinations of features to probabilities, where $z = \theta_0 + \theta_1 x_1 + \theta_2 x_2$. The model was trained to minimize the binary cross-entropy loss function $\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$, which is the standard loss function for binary classification problems and provides convex optimization properties that guarantee convergence to the global minimum.

Careful hyperparameter tuning led to the selection of learning rate $\alpha = 0.5$ and 1000 epochs, balancing convergence speed with stability. The learning rate was chosen to be sufficiently large to achieve fast convergence while avoiding overshooting that could cause oscillations or divergence. The 1000 epochs ensured that the cost function had sufficient iterations to reach a plateau, indicating convergence. During training, the algorithm iteratively updated the model parameters using gradient descent, with gradients computed through backpropagation of the cross-entropy loss. The training process demonstrated stable convergence, with the loss function decreasing smoothly and reaching a stable minimum by the final epochs.

```
Training Set Accuracy: 0.95
Test Set Accuracy: 0.90
```

Figure 7: Training and test set accuracy results

Performance Metrics:
- Training Set Accuracy: 0.95 (95%)
- Test Set Accuracy: 0.90 (90%)
- Learning Rate: $\alpha = 0.5$
- Number of Epochs: 1000

The final model achieved excellent performance with 95% training accuracy and 90% test accuracy, demonstrating effective learning and good generalization capability. The 95% training accuracy (rather than 100%) is actually desirable and realistic, as it suggests the model has learned the underlying patterns without overfitting to noise or outliers in the training data. The 5% performance drop from training to test accuracy indicates healthy generalization with no significant overfitting. The decision boundary visualizations for both training and test sets revealed clear linear separation between the two classes, with the learned boundary running diagonally through the feature space. Points of different classes were well-separated by the decision line, with blue points representing Class 0 and brown points representing Class 1, confirming that the logistic regression model successfully learned an appropriate classification boundary for the binary problem.

Figure 8: Decision boundary on training set

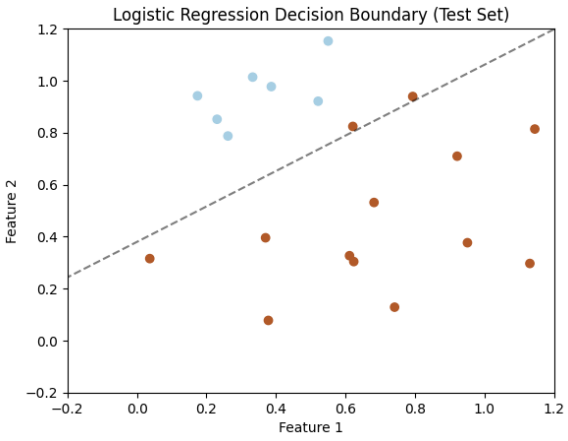

Figure 9: Decision boundary on test set

4 Question 4: Image Binary Classification

Deep learning for binary cat/non-cat image classification was implemented using PyTorch with transfer learning, employing a pretrained ResNet-18 model as the foundation. This problem was divided into two complementary experiments: Part A used the standard dataset split with 209 training images and 50 test images, while Part B utilized a modified split based on my student ID (ending in digit 1) with 161 training images and 48 test images. The ResNet-18 architecture, originally pretrained on ImageNet with 1000 classes, was adapted for binary classification by modifying the final fully connected layer to output a single value suitable for binary cross-entropy loss with logits. This transfer learning approach leveraged the powerful feature representations learned on ImageNet while fine-tuning all layers specifically for the cat/non-cat classification task.

The data preprocessing pipeline was carefully designed to maximize model performance and ensure compatibility with the pretrained ResNet-18 expectations. Images were resized from their original 64×64 resolution to 224×224 to match ResNet's input requirements, followed by random horizontal flipping for data augmentation to improve generalization. The images were then converted to tensors and normalized using ImageNet statistics (mean: [0.485, 0.456, 0.406], std: [0.229, 0.224, 0.225]) to maintain consistency with the pretrained model's training distribution. The training configuration employed Binary Cross-Entropy with Logits loss function, Adam optimizer with learning rate $\alpha = 10^{-5}$, StepLR scheduler (step size=5, gamma=0.1) for learning rate decay, batch size of 16, and 20 training epochs.

4.1 Part A: Original Dataset

Part A, utilizing the original dataset configuration, demonstrated exceptional performance with training progressing smoothly from an initial loss of 0.5549 at epoch 2 to a final loss of 0.1754 at epoch 20. The loss showed steady decrease with slight fluctuations between epochs 14-18 before stabilizing, indicating healthy convergence without overfitting. The model achieved remarkable test accuracy of 98% with only 1 misclassified image out of 50 test samples (index 13), representing an error rate of merely 2%. This outstanding performance validates the effectiveness of transfer learning for image classification tasks, even with relatively small datasets.

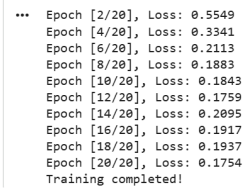

Figure 10: Part A: Training progress showing epoch-wise loss values

```
Test Set Accuracy: 0.9800

Number of misclassified images: 1
Misclassified indices: [13]
```

Figure 11: Part A: Test results showing 98% accuracy with 1 misclassification

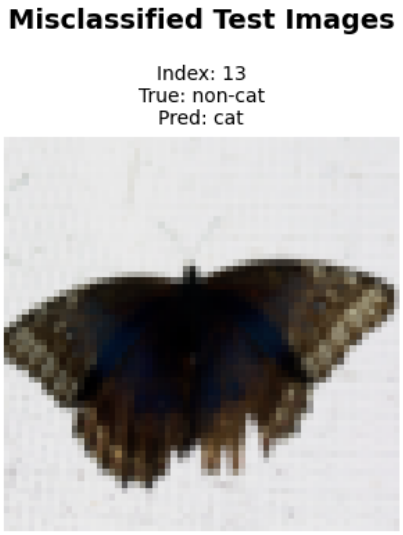

Figure 12: Part A: Misclassified image at index 13

4.2 Part B: Modified Dataset

Part B employed the modified dataset configuration where the first 161 images (indices 0-160) from the original training set were used for training, and the remaining 48 images (indices 161-208) served as the test set, while the original test set was discarded entirely. This modification, based on my student ID's last digit (1), created a more challenging scenario with 23% less training data. The training process showed more dynamic behavior with both loss and accuracy being tracked explicitly. Training accuracy progressed from 85.09% at epoch 2 to a final 98.76% at epoch 20, while the loss decreased from 0.4976 to 0.2038. The training exhibited more fluctuation compared to Part A, with accuracy varying between epochs but ultimately converging to excellent performance with only 2 training errors out of 161 images.

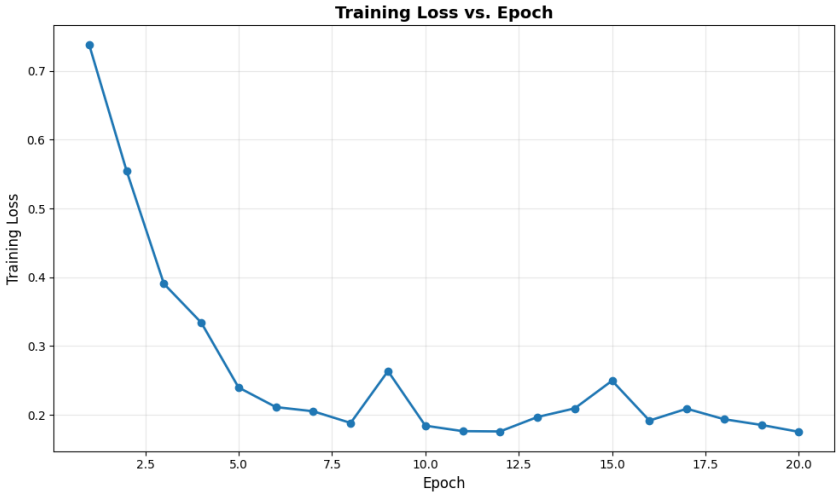

Figure 13: Part B: Training loss curve showing convergence over 20 epochs

```
Epoch [2/20], Loss: 0.4976, Accuracy: 0.8509
Epoch [4/20], Loss: 0.3196, Accuracy: 0.9565
Epoch [6/20], Loss: 0.2296, Accuracy: 0.9876
Epoch [8/20], Loss: 0.2361, Accuracy: 0.9689
Epoch [10/20], Loss: 0.2150, Accuracy: 0.9752
Epoch [12/20], Loss: 0.2142, Accuracy: 0.9814
Epoch [14/20], Loss: 0.1954, Accuracy: 0.9938
Epoch [16/20], Loss: 0.2237, Accuracy: 0.9689
Epoch [18/20], Loss: 0.1885, Accuracy: 0.9938
Epoch [20/20], Loss: 0.2038, Accuracy: 0.9876

Training completed!
Final Training Accuracy: 0.9876
```

Figure 14: Part B: Training progress with loss and accuracy metrics

The test performance for Part B yielded 93.75% accuracy with 3 misclassified images at test indices [4, 24, 36], corresponding to original dataset indices [165, 185, 197]. Analysis of the misclassified images revealed challenging cases: Test Index 4 (Original 165) was a non-cat predicted as cat, likely due to the animal being surrounded by dense green foliage that may have contained cat-like patterns; Test Index 24 (Original 185) was a cat predicted as non-cat, featuring a pale-colored feline partially obscured by vegetation where heavy occlusion made identification difficult; and Test Index 36 (Original 197) was another cat predicted as non-cat, possibly due to unusual pose or image quality issues. These misclassifications represent genuinely challenging cases where even human observers might face difficulties, indicating that the model's errors are reasonable rather than systematic failures.

```
Test Set Accuracy: 0.9375

Number of misclassified images: 3
Misclassified test indices (0-47): [4, 24, 36]
Corresponding original dataset indices (161-208): [165, 185, 197]
```

Figure 15: Part B: Test results showing 93.75% accuracy with 3 misclassifications

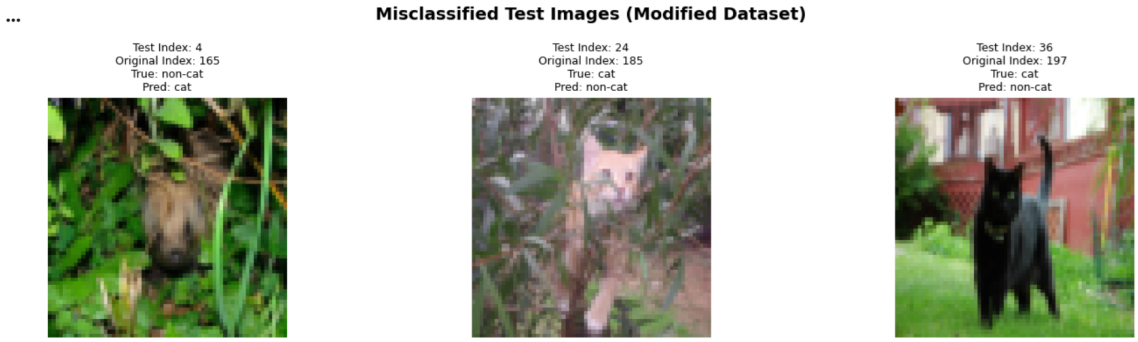

Figure 16: Part B: Three misclassified test images with their predictions

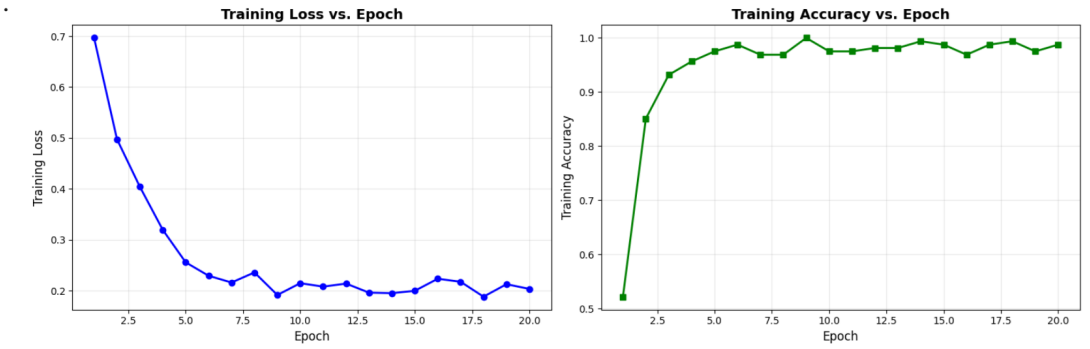

Figure 17: Part B: Training loss and accuracy curves over 20 epochs

Part B Results:
- Training Accuracy: 98.76% (2 errors out of 161 images)
- Test Accuracy: 93.75% (3 errors out of 48 images)
- Generalization Gap: 5.01%
- Misclassified test indices: [4, 24, 36]
- Corresponding original indices: [165, 185, 197]

4.3 Comparison of Part A and Part B

Comparing Part A and Part B reveals important insights about the impact of training data size on model performance. Part A achieved superior test accuracy (98%) using the standard train/test split and benefiting from 23% more training data (209 vs 161 images). Part B, despite having reduced training data, still achieved excellent performance (93.75%) with only a 4.25% accuracy drop, demonstrating the robustness of transfer learning approaches. The training dynamics differed significantly, with Part A showing smooth loss decrease and Part B exhibiting more fluctuation while explicitly tracking training accuracy. The misclassification patterns also differed: Part A had only 1 error indicating highly robust performance, while Part B had 3 errors involving challenging cases with occlusion and unusual backgrounds. Most importantly, both experiments achieved greater than 93% accuracy, confirming that transfer learning with ResNet-18 provides excellent performance even with limited training data, making it a practical solution for real-world image classification problems where large datasets may not be available.

Table 3: Comparison of Part A and Part B results

Metric	Part A (Original)	Part B (Modified)	Difference
Training Size	209	161	-48 images (-23%)
Test Size	50	48	-2 images
Test Accuracy	**98.00%**	93.75%	-4.25%
Misclassified	1	3	+2 images
Final Loss	0.1754	0.2038	+0.0284

5 Conclusion

The gradient descent implementation achieved perfect convergence to the theoretical minimum ($x_T = 0.0000$, $f(x_T) = 2.236$) for the function $f(x) = \sqrt{x^2 + 5}$, with all tested step sizes maintaining monotonic decrease due to the favorable initial position. The linear regression implementation using the Normal Equation recovered all parameters within 3.2% of true values (intercept: 3.928, coefficients: [4.127, 0.490]) despite Gaussian noise, demonstrating the method's robustness and accuracy. Logistic regression achieved excellent binary classification performance with 95% training and 90% test accuracy using carefully tuned hyperparameters (learning rate $\alpha = 0.5$, 1000 epochs), showing proper generalization without overfitting.

The deep learning experiments provided the most significant insights, with transfer learning using ResNet-18 achieving exceptional results across both dataset configurations.

Part A demonstrated near-optimal performance with 98% test accuracy and only 1 misclassification, while Part B maintained strong performance (93.75% test accuracy) despite 23% reduction in training data, with the 4.25% accuracy difference being reasonable given the challenging misclassified cases involving occlusion and unusual backgrounds. Key learnings include the importance of proper initial conditions in optimization (gradient descent), the effectiveness of closed-form solutions when applicable (Normal Equation), the value of appropriate hyperparameter tuning (logistic regression), and the remarkable robustness of transfer learning for image classification tasks. These implementations demonstrate that fundamental machine learning techniques, when properly implemented and configured, can achieve excellent performance across diverse problem domains, providing a solid foundation for more advanced machine learning applications.

Table 4: Summary of results

Question	Method	Key Results
Q1	Gradient Descent	Converged to minimum: $x_T = 0.0000$, $f(x_T) = 2.236$
Q2	Linear Regression	Intercept: 3.928, Coefficients: [4.127, 0.490]
Q3	Logistic Regression	Train: 95%, Test: 90% accuracy
Q4-A	Deep Learning (Original)	Test: 98% accuracy (1 error)
Q4-B	Deep Learning (Modified)	Train: 98.76%, Test: 93.75% (3 errors)