AI600 – Assignment 1
Deep Learning

Muhammad Ali Siddique
25280022

February 18, 2026

GitHub Repository

`https://github.com/alisidd-00/25280022_DEEP_LEARNING_ASSIGNMENT_1`

Contents

1 Question 1

1.1 Part A: Data Preprocessing and Exploratory Analysis

1.1.1 Missing Values

Rows containing missing values were dropped. The number of missing entries was small, so dropping them preserved dataset consistency without introducing assumptions through filling them with mean/median/mode etc. This approach avoids bias and keeps preprocessing simple and appropriate for the dataset size.

1.1.2 Class Distribution

The target variable is clearly imbalanced. Moderate (class 1) is the majority class with roughly 20k samples, while Luxury (class 3) is the smallest with about 2.3k samples. The largest to smallest ratio is approximately 8.7. Without mitigation strategies such as class weighting or resampling, a classifier may most likely become biased toward the majority class.

Figure 1: Distribution of price classes

1.1.3 Categorical Encoding

Categorical variables `neighbourhood_group` and `room_type` were encoded using one-hot encoding. This method avoids introducing artificial ordinal relationships between categories and is standard practice for neural network inputs.

1.1.4 Normalization

Numerical features were standardized using z-score normalization (subtract mean, divide by standard deviation). This ensures features are on comparable scales so no single feature dominates due to magnitude.

1.1.5 Feature Relationships

Analysis of numerical features showed:

- **minimum_nights**: Weak separation between classes.
- **number_of_reviews**: Moderate class tends to have the highest average.
- **availability_365**: Clear upward trend across price classes.
- **amenity_score**: Strong monotonic increase from Budget to Luxury, indicating strong predictive power.

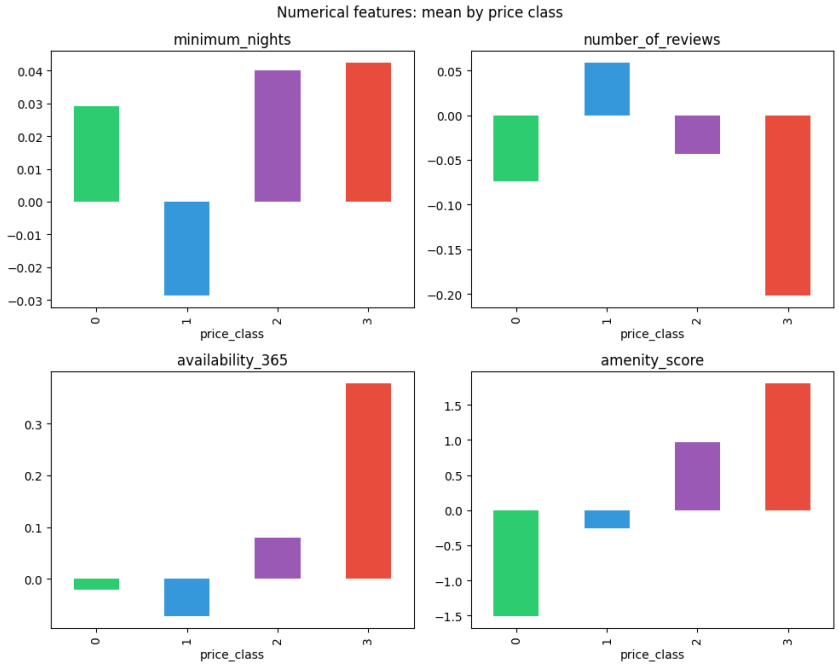

Figure 2: Feature relationships with price class

1.1.6 Correlation Matrix

The Pearson correlation matrix showed:

- Amenity score has strong positive correlation with price class.
- Other numerical features show weak correlations.
- No strong multicollinearity between features.

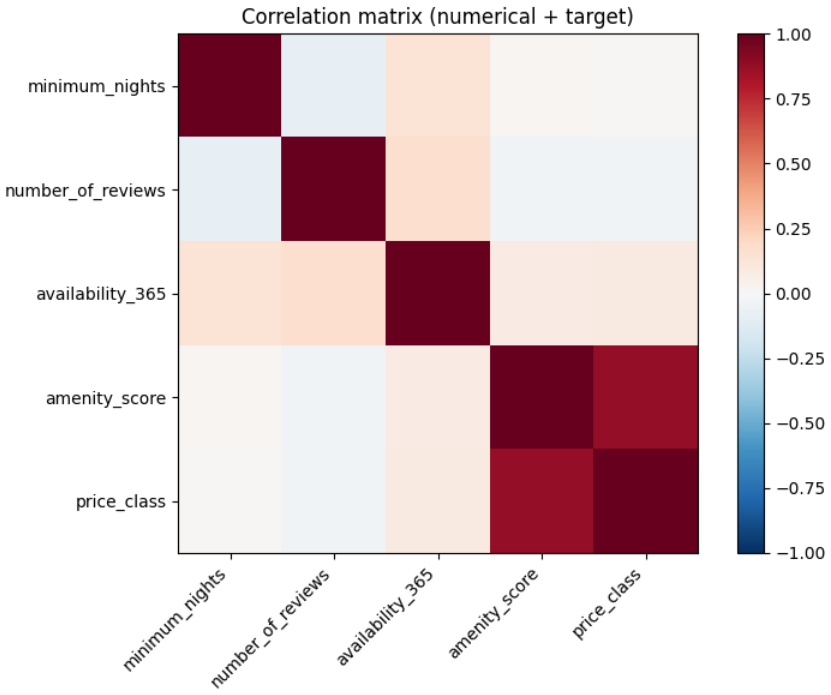

Figure 3: Correlation matrix

1.1.7 Conclusions from EDA

Most influential features:

- Amenity score (strongest predictor)
- Room type
- Neighbourhood group
- Availability (moderate signal)

Amenity score appears unusually dominant, with correlation around 0.88 and clear separation across classes.

1.2 Part B(a): Two-Layer Perceptron from Scratch

A two-hidden-layer MLP was implemented using NumPy with explicit forward propagation, cross-entropy loss, and backward propagation. Training was performed using batch gradient descent for 200 iterations.

1.2.1 Training Results

Final accuracies:

- Sigmoid: train ≈ 0.5659, validation ≈ 0.5566
- ReLU: train ≈ 0.82, validation ≈ 0.82

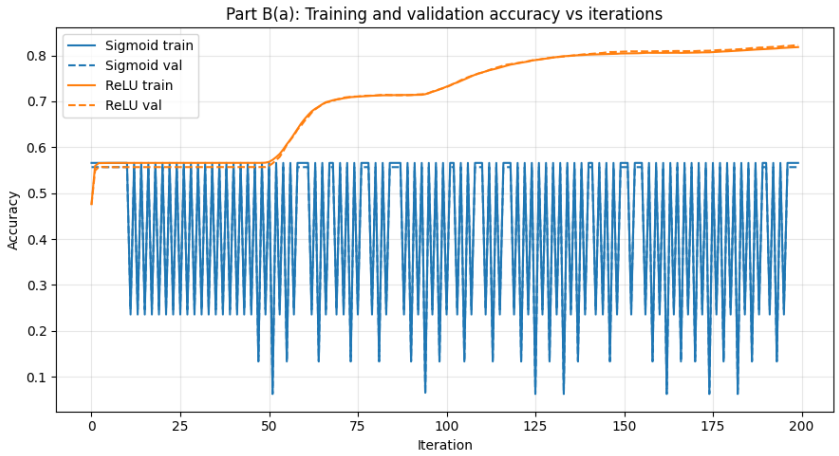

Figure 4: Training and validation accuracy vs iterations

1.2.2 Activation Function Comparison

Sigmoid showed unstable learning and saturation effects leading to poor convergence. ReLU produced smooth convergence and stable gradients, achieving significantly higher accuracy.

1.3 Part B(b): Gradient Magnitude Comparison

Observations:

- Sigmoid first layer gradients were extremely small, demonstrating vanishing gradients.
- Sigmoid second layer gradients were unstable early and diminished over time.
- ReLU gradients remained stable and nonzero in both layers.

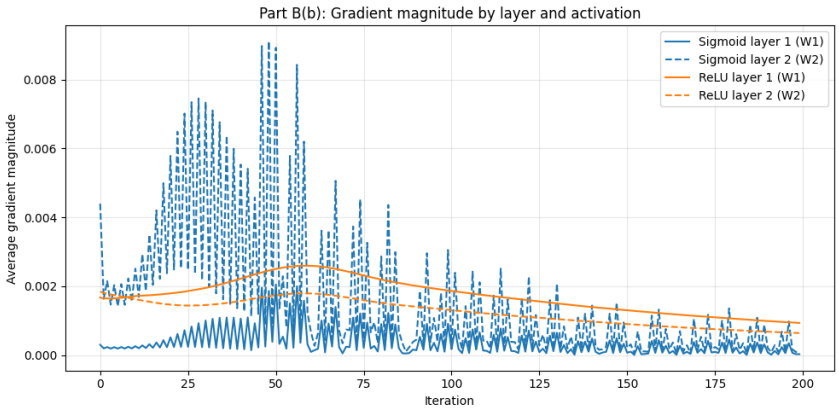

Figure 5: Gradient magnitude comparison

1.4 Part C(a): Gradient-Based Feature Attribution (Analytical)

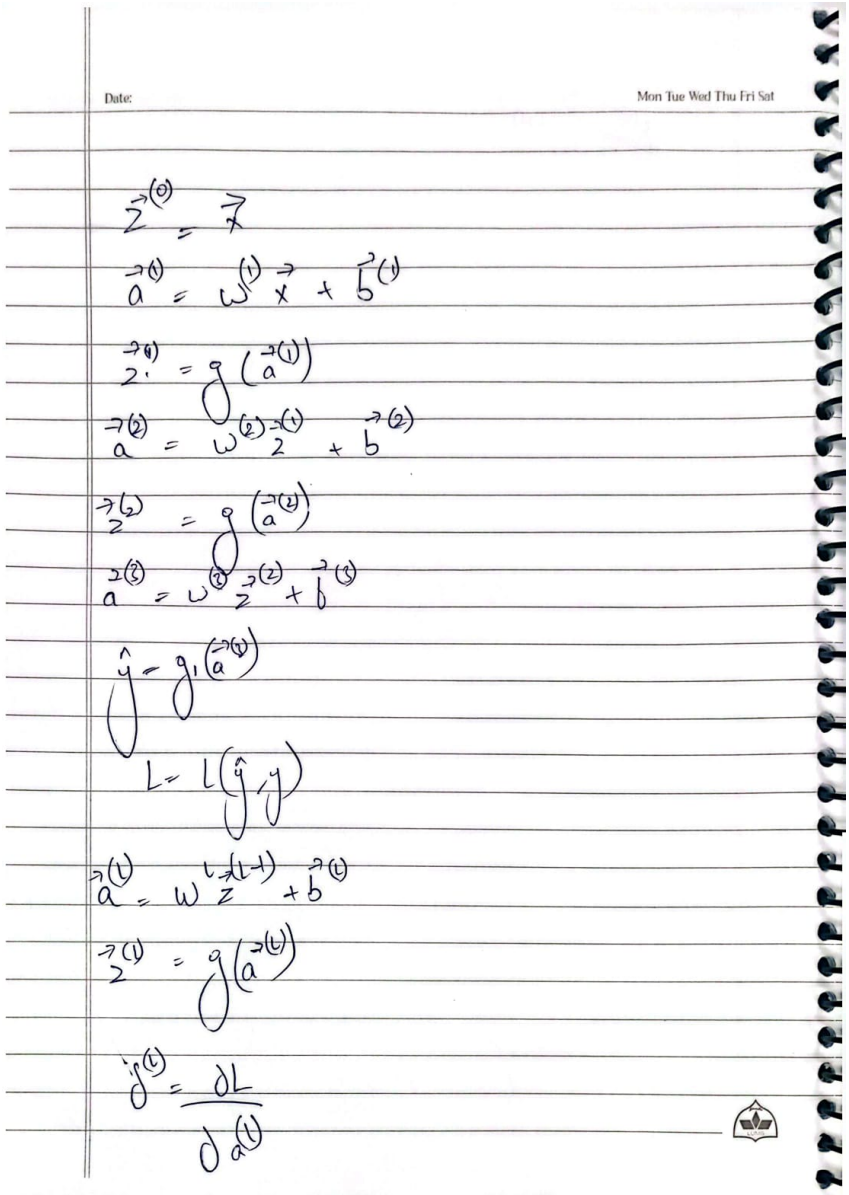

Figure 6: Handwritten Solution

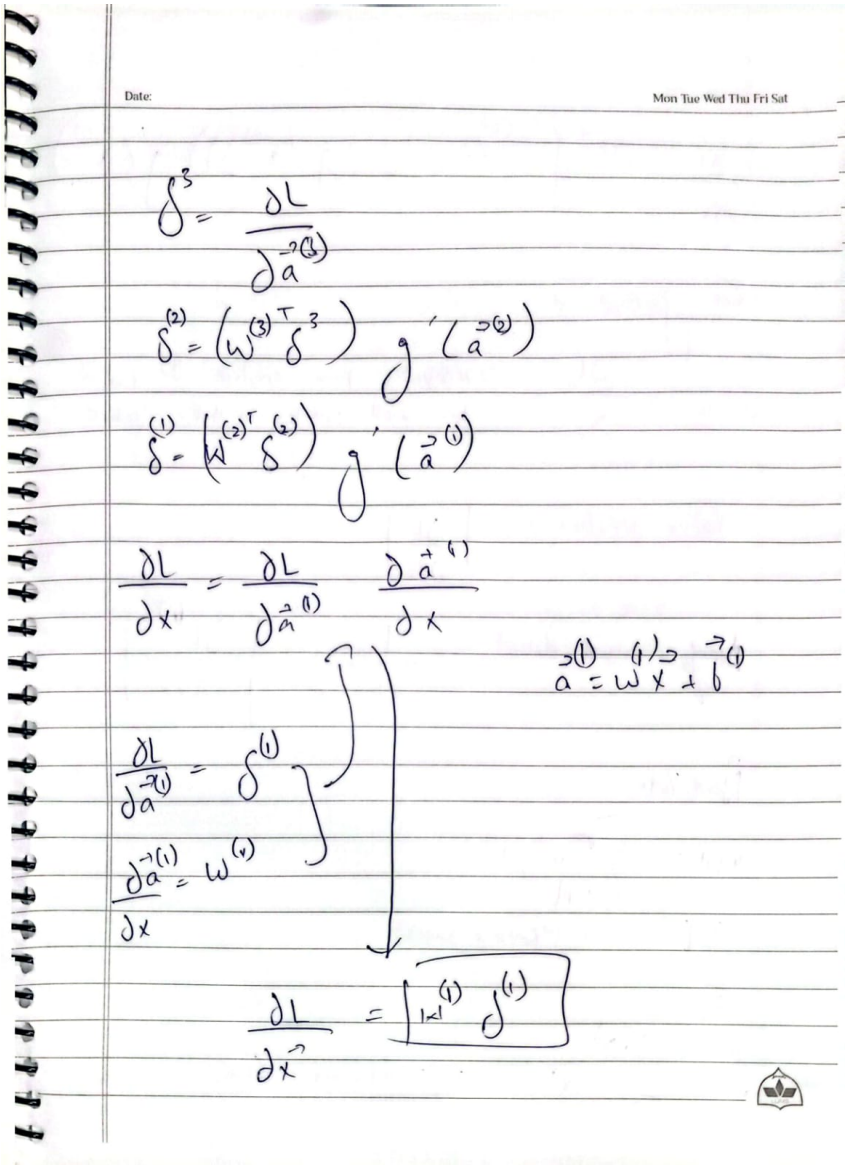

Figure 7: Handwritten Solution

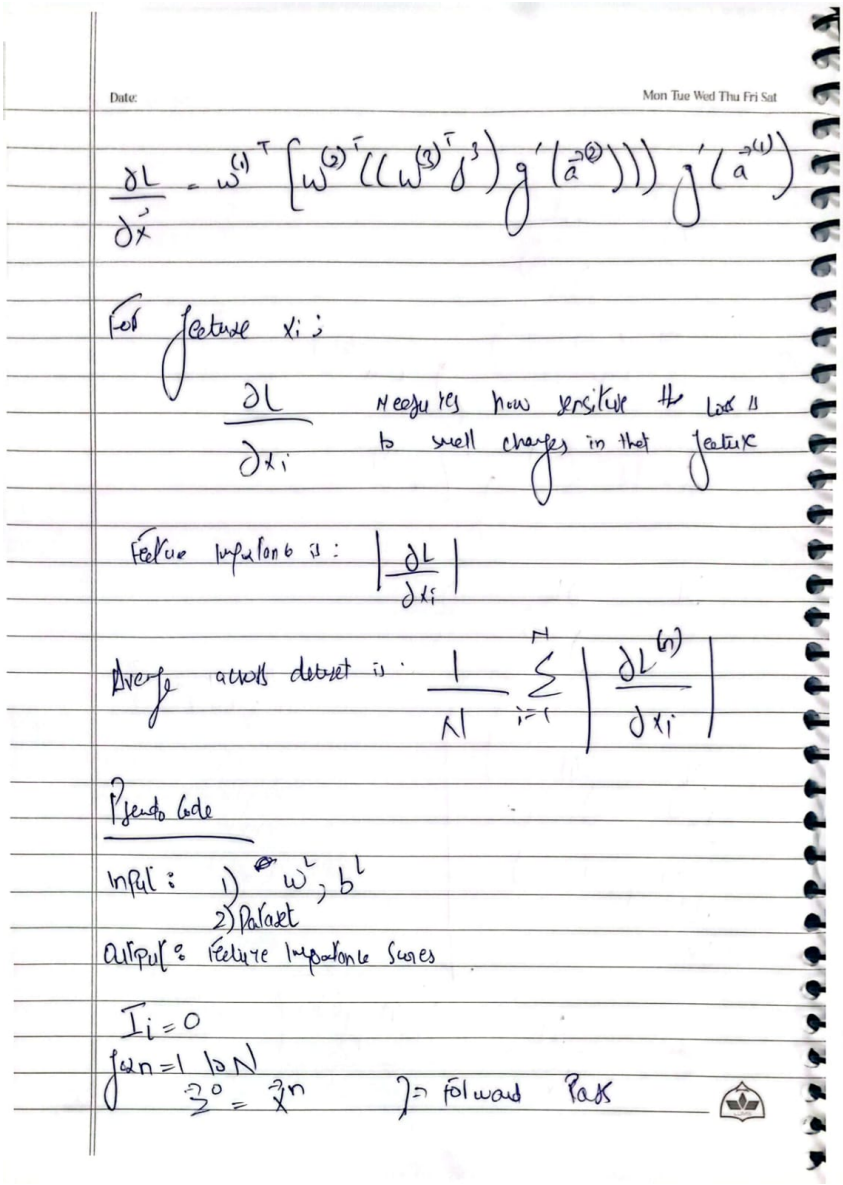

Figure 8: Handwritten Solution

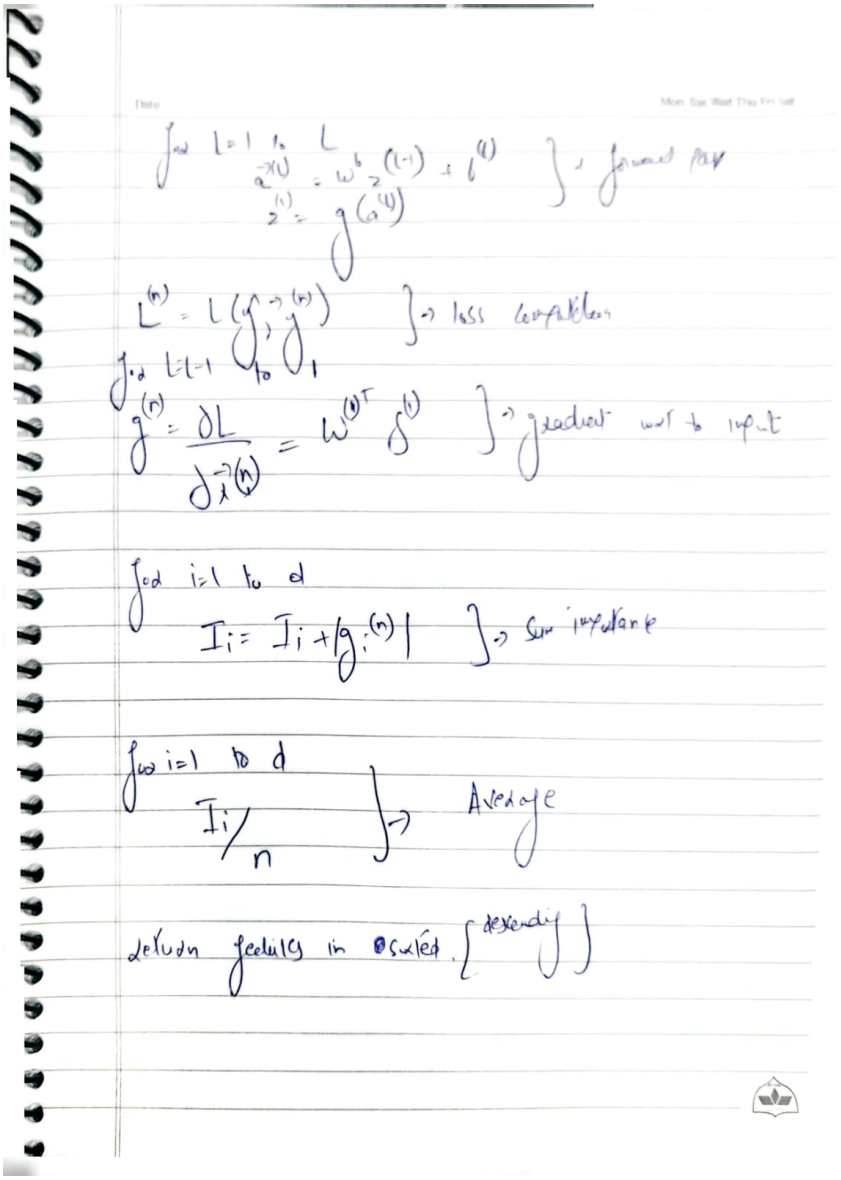

Figure 9: Handwritten Solution

1.5 Part C(b): Feature Attribution

Gradient-based feature attribution was implemented by computing the gradient of the winning class output with respect to input features and averaging magnitudes across correctly classified samples.

Results showed:

- Amenity score was the most influential feature.
- Room type and neighbourhood group followed.
- Other numerical features contributed less.

These findings match conclusions from Part A, indicating the model learned meaningful patterns.

Pseudocode

```
Input: trained model, data X
Initialize feature_importance = zeros(D)

For each sample x in X:
    p = model.forward(x)
    c = argmax(p)
    grad_x = gradient(output_c, x)
    feature_importance += abs(grad_x)

feature_importance /= number_of_samples
ranked_features = sort_descending(feature_importance)
```

1.6 Part D: Test Evaluation and Generalization

1.6.1 Reported Performance

The trained ReLU MLP achieved:

- Training accuracy ≈ 0.82
- Validation accuracy ≈ 0.82
- Test accuracy ≈ 0.39

Training and validation accuracies are very similar, indicating that the model fits the training distribution well and does not exhibit severe overfitting within the training dataset. However, the large drop in performance on the test set indicates a significant generalization gap.

1.6.2 Analysis of the Performance Gap

Training and validation sets were obtained from the same dataset (train.csv) using a random split, so both follow the same data distribution. The test set (test.csv) is a separate held-out dataset, and the large drop in accuracy suggests that its feature distribution or feature–label relationships differ from those seen during training.

1.6.3 Root Cause Using Evidence from Part A and Part C

Evidence from earlier analysis helps explain this behavior:

1. Evidence from EDA (Part A) Exploratory analysis showed:

- Strong class imbalance in the dataset.
- Amenity score had extremely strong correlation with the target (around 0.88).
- Other features provided weaker signals.

This indicates that the model can rely heavily on a small number of dominant features rather than learning robust multi-feature patterns.

2. Evidence from Feature Attribution (Part C) Gradient-based attribution showed:

- Amenity score was by far the most influential feature.
- Room type and neighbourhood group were secondary contributors.
- Other numerical features had relatively low importance.

This confirms that the model's predictions are highly dependent on a small subset of features.

1.6.4 Why the Model Fails on the Test Set

Because the model relies strongly on a few dominant features, any change in:

- distribution of amenity scores,
- relationship between amenity score and price class,
- distribution of neighbourhoods or room types,

can significantly degrade performance. If the test dataset differs slightly in these relationships, the learned decision boundaries may no longer be valid, resulting in poor accuracy.

Thus, the generalization failure is primarily due to:

- reliance on a small number of dominant features,
- class imbalance,
- possible distribution shift between train and test datasets.

1.6.5 Mitigation Strategies

Several approaches could improve generalization:

- Regularization (e.g., L2, L1) to reduce reliance on dominant features.
- Class weighting or resampling to reduce bias toward majority classes.
- Collecting or incorporating more diverse training data to better match the test distribution.
- Using cross validation rather than a single train/validation split for more robust model selection.

These strategies would encourage the model to learn more stable and transferable feature representations, improving performance on unseen data.

2 Question 2

2.1 Handwritten Derivations

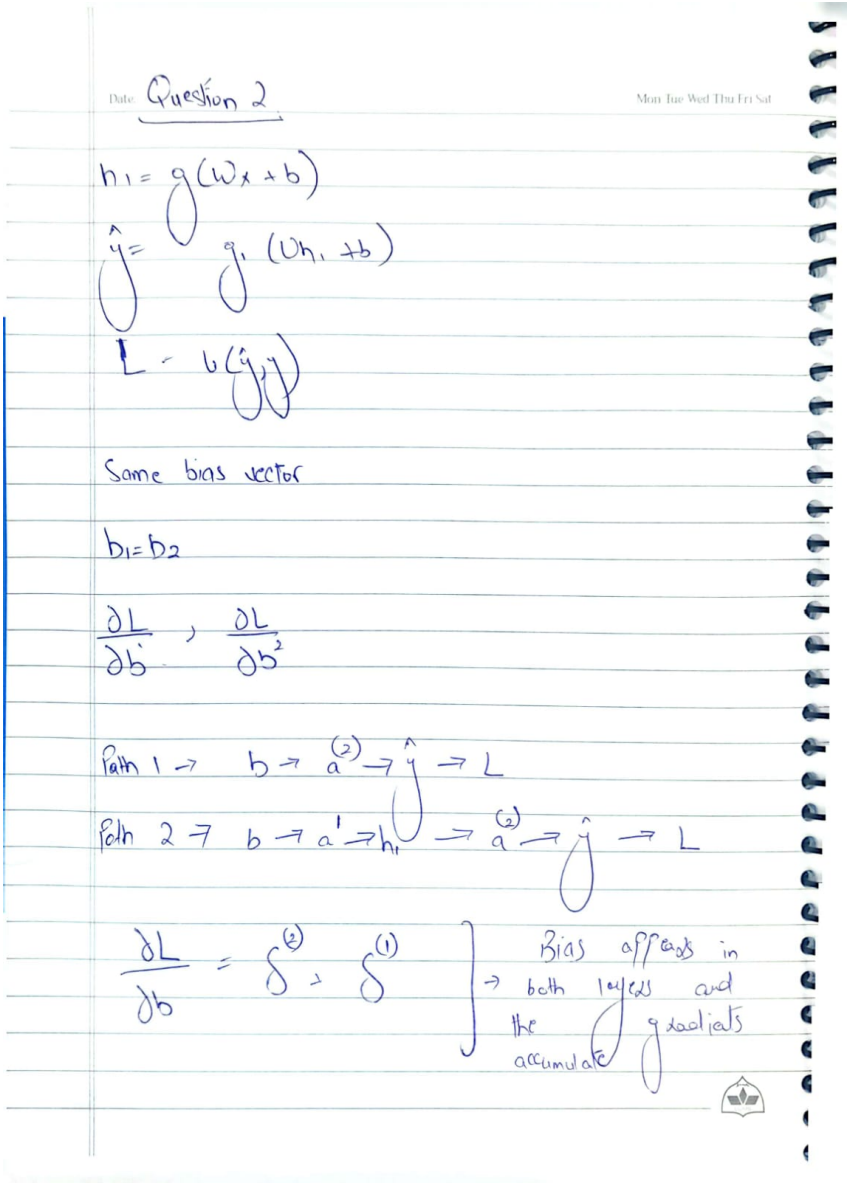

Figure 10: Handwritten Solution

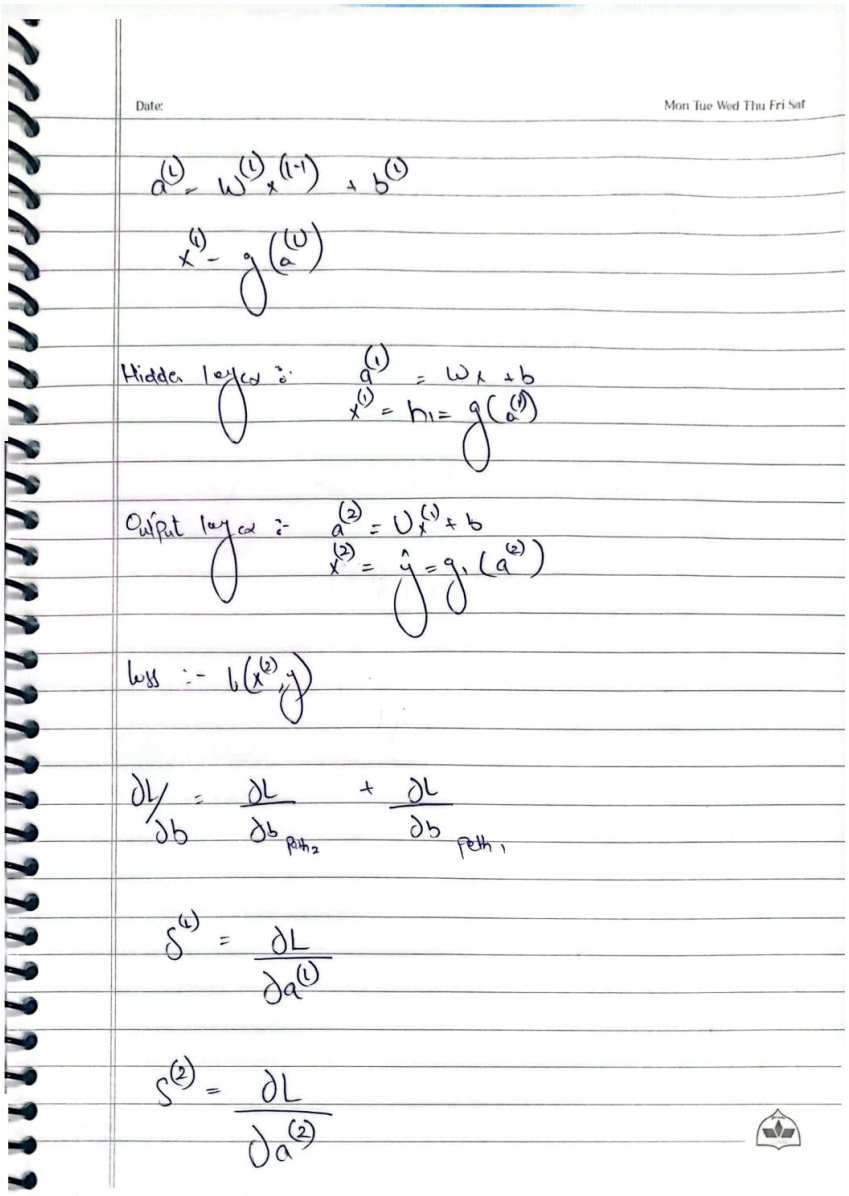

Figure 11: Handwritten Solution

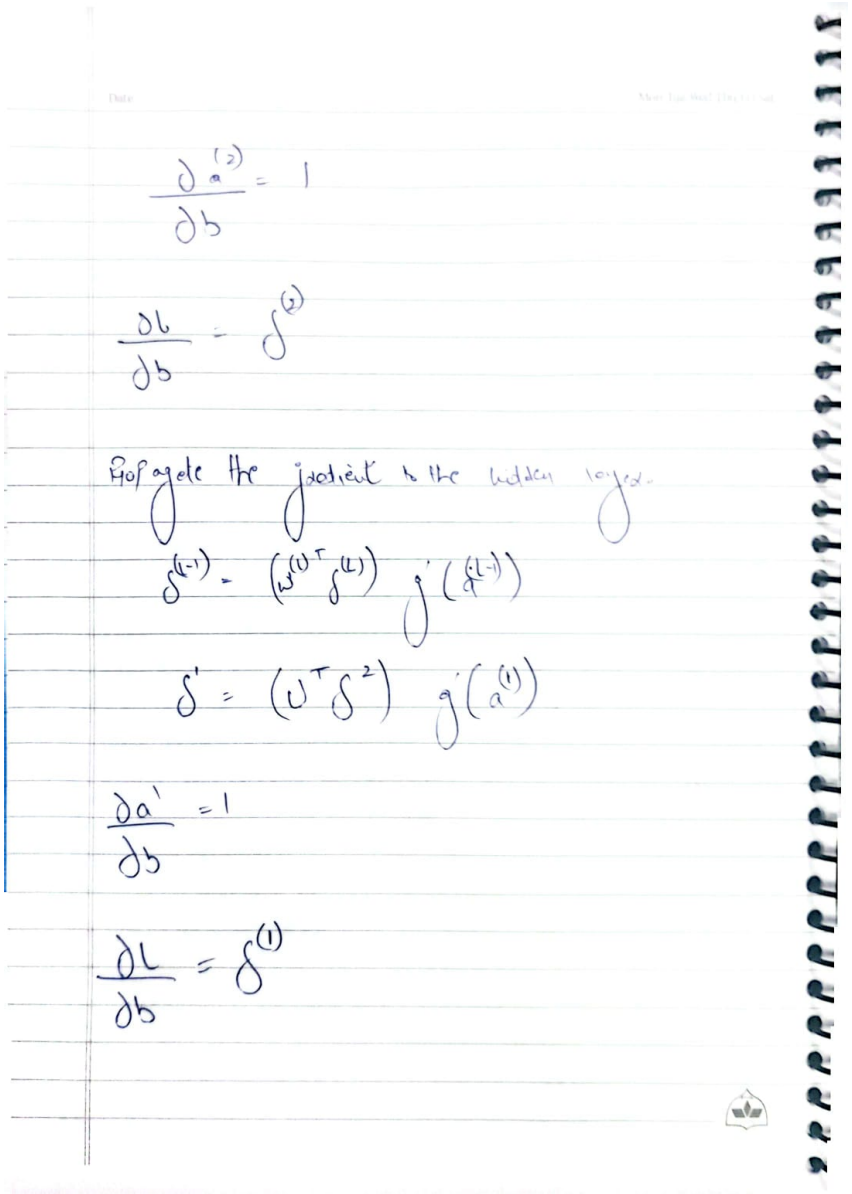

Figure 12: Handwritten Solution

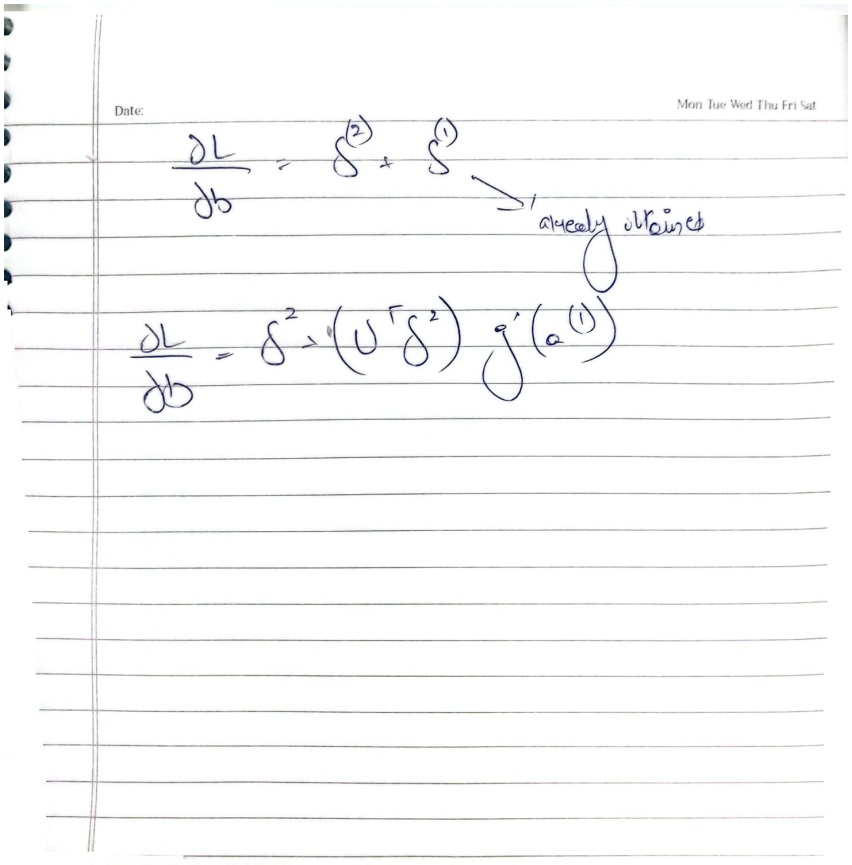

Figure 13: Handwritten Solution

2.2 Optimization and Convergence Effect

We compare two models:

- Shared-bias model: the same bias parameter is used in both layers.
- Independent-bias model: each layer has its own bias.

Bias sharing reduces the number of parameters and introduces parameter coupling between layers. The gradient of the shared bias accumulates contributions from multiple layers, which can increase gradient magnitude and make optimization less well conditioned. At the same time, sharing biases reduces flexibility, since each layer cannot independently shift its activation distribution. In contrast, independent biases allow each layer to adapt separately, improving conditioning and typically leading to more stable convergence in gradient based optimization. Therefore, independent biases are generally expected to provide faster and more stable convergence, while shared biases may act as a mild regularizer but at the cost of representational flexibility.

3 Use of Generative AI Tools

Prompts Used

I used ChatGPT and Claude to assist with:

- Providing syntax for data manipulation , setting up of two layer perceptron.
- Giving me a sample latex output which I then modified based on my work
- Checking whether interpretations of results (EDA findings, gradient attribution, and generalization gap) were logically consistent.

Example prompt:

> "Based on my Jupyter notebook outputs and analysis cells, can you double check my responses in the output cells and text cells of my jupyter notebook, especially Part D of Question 1?"

I used AI tools for writing syntaxes that are used, modified them according to the needs of the assignment. I hadn't worked on latex before, so I generated a draft template and filled it in accordingly. The work is solely mine but with the advent of AI tools, their importance can't be neglected as they speed up time and and provide efficiency in redundant tasks such as looking up syntaxes etc.