Project Documentation: Amazon Bin Image Quantity Prediction

Team Members
Syed Sufyaan – SE22UECM046
Ayaz Ahmed Ansari – SE22UARI024
Aryan Pawar – SE22UARI195
Hamza Babukhan – SE22UARI208

1. Problem Statement

This project focuses on developing an end-to-end deep learning system capable of estimating the number of objects present inside a warehouse storage bin. The system is built using the Amazon Bin Image Dataset, which includes real-world bin photographs and their corresponding metadata. The goal of the model is to predict the total item count directly from the input image, thereby automating one of the fundamental operations in warehouse inventory validation.

The solution mimics the workflow used in real fulfillment centers, where automated systems assist in checking whether the expected number of items are present in each bin. To achieve this, the project includes dataset preparation, model design, training, evaluation, and deployment through a user-friendly interface.

2. Dataset Description

The dataset contains 10,000 bin images sampled from the larger Amazon Bin Image Dataset. Each image has a corresponding metadata JSON file, which stores details such as the item identifiers (ASIN), quantity, dimensions, and other product attributes.

To train the model, the dataset was divided into:
• 80% Training Data (8000 images)
• 10% Validation Data (1000 images)
• 10% Test Data (1000 images)

The images vary significantly in terms of lighting, bin size, and degree of object occlusion. Many bins contain multiple objects overlapping or partially hidden. Understanding the structure of these images and metadata was an important step before preparing the regression model pipeline.

3. Model Architecture

A custom ResNet-style regression architecture was designed for predicting the quantity of items inside a bin. The model includes:

- Four residual blocks for deep feature extraction
- Convolutional layers with batch normalization and ReLU activation
- Global Average Pooling to reduce spatial dimensions while retaining feature importance
- A fully connected regression head producing a single numeric output

The input to the model is a 416×416 RGB image, resized and normalized. The output is a floating-point quantity, which can later be rounded for evaluation and display. The final model consists of 11.34 million parameters. Residual connections were used to maintain gradient flow in deeper layers and improve training stability.

4. Training Details

The model was trained using the following configuration:

Loss Function: Mean Squared Error (MSE)
Chosen for its suitability in regression tasks where the cost of larger errors grows quadratically.

Optimizer: AdamW
This optimizer combines Adam's adaptive learning with decoupled weight decay, improving generalization and convergence in deep networks.

Learning Rate Scheduling: ReduceLROnPlateau
This scheduler monitors validation MAE and reduces the learning rate when improvement stalls, enabling the model to fine-tune deeper during later epochs.

Mixed Precision Training:
Implemented using gradient scaling to reduce memory usage and accelerate training while maintaining numerical stability.

Training was performed over 25 epochs, and the progression of loss and MAE was monitored to ensure consistent learning behavior and early detection of overfitting or divergence.

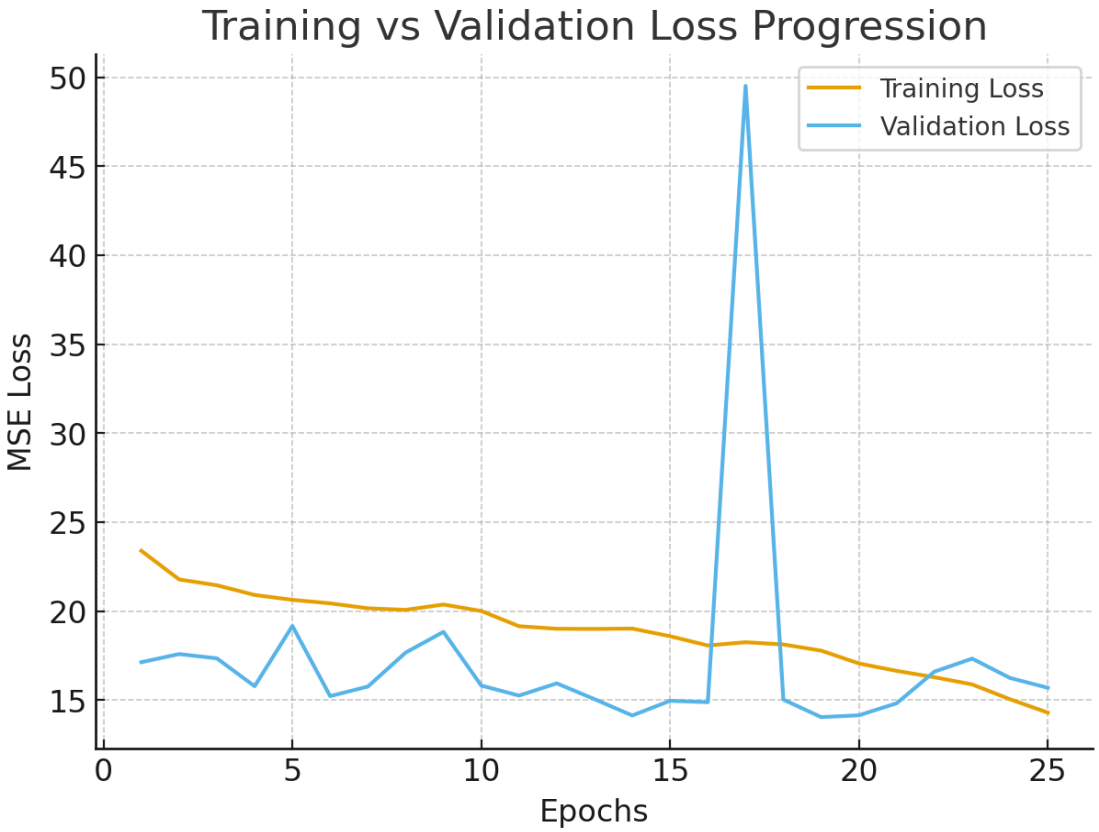
5. Performance Scores

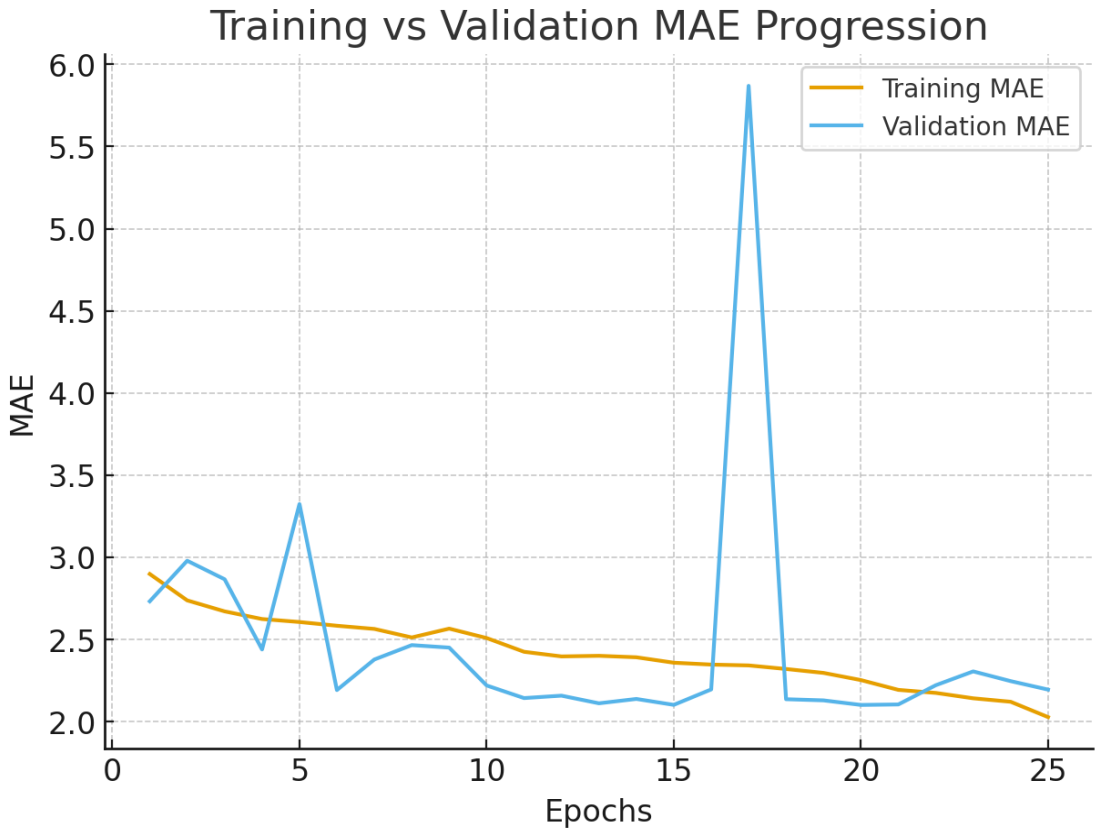
After completing 25 epochs of training, the model achieved the following results:

- **Best Validation MAE:** 2.10
- **Best Validation MSE:** ~15.2
- **Final Train MAE:** ~2.03

The training and validation curves provided visual insights into the model's behavior across epochs.

The closeness between training and validation scores indicates stable learning, showing that the model was able to generalize effectively on unseen validation images.




Training vs Validation MAE Progression
Training MAE
Validation MAE
MAE
Epochs
6.0
5.5
5.0
4.5
4.0
3.5
3.0
2.5
2.0
0
5
10
15
20
25

6. Model Deployment

The trained model was integrated into a Streamlit-based web application. The application supports:

- Batch image upload functionality
- Real-time model inference for quantity prediction
- Display of prediction values along with confidence estimation mechanisms

The model weights are stored in cloud storage and loaded dynamically when the application initializes. This architecture ensures portability, as the app can be deployed on local machines, cloud servers, or GPU-enabled environments.

7. Key Outcomes of the Project

Throughout the development lifecycle, the following outcomes were achieved:

- Construction of a scalable dataset handling pipeline
- Implementation of a custom deep learning regression architecture
- Successful monitoring of training through performance curves
- Deployment of a functional user-facing demo for real-time predictions

8. Future Work

Several extensions can further enhance the system:

- Integration of object detection architectures to detect and classify items inside bins
- Combining metadata features (item dimensions, weight) into multimodal models
- Implementing model monitoring and retraining triggers as part of an MLOps pipeline
- Adding barcode or ASIN recognition to validate individual product identity alongside quantity

9. References

The project references the dataset descriptions, objectives, and guidelines provided in the course assignment document, which outlines the expectations for dataset usage, model experimentation, UI development, and deployment design.