Capstone Project Report

# Introduction

Inventory management is a critical function in modern distribution centers (DCs). It ensures the correct quantity of goods is available, enabling smooth operations across the supply chain. Traditionally, inventory counting relied on manual processes, which are labor-intensive, error-prone, and not scalable for high-throughput fulfillment centers. With advancements in automation and computer vision, inventory tracking using image-based methods offers a scalable and efficient solution. During high-demand periods (e.g., global pandemic scenarios), where manual labor is limited, the ability to automate such processes has proven essential to maintaining supply chain integrity. This project focuses on utilizing machine learning and computer vision to automate item counting in storage bins. The goal is to replace manual inventory checks with a robust model that processes bin images and predicts object counts reliably

# Data Preprocessing

Dataset: Amazon Bin Image Dataset Source: Open-source dataset available through AWS resources. Description: - Over 500,000 labeled images of bins from a live Amazon Fulfillment Center. - Each image has an associated JSON metadata file with: - ASINs (Amazon product codes) - Object quantity - Size, weight, and dimension attributes.

The dataset is structured into subdirectories where each subdirectory corresponds to a class.

A screenshot of a computer

AI-generated content may be incorrect.

A graph of blue rectangular bars

AI-generated content may be incorrect.

While not severely imbalanced, there is a noticeable difference between the smallest class (~1200 images) and the largest class (~2600 images). The largest class has more than double the images of the smallest.

The `ImageFolder` utility from torchvision was used to load and manage the data.  
  
The following transformations were applied:  
- Random horizontal flip and resized crop (for training)  
- Resize and center crop (for validation and testing)  
- Normalization using ImageNet mean and standard deviation  
  
Train, validation, and test datasets were split in a 70:15:15 ratio using `random\_split`.

# 3. Algorithms and Techniques

The chosen model is EfficientNet-B3, a convolutional neural network architecture known for its efficiency and performance across image classification tasks. EfficientNet scales depth, width, and resolution using a compound scaling method.  
  
We utilized transfer learning by freezing the pretrained layers of EfficientNet and replacing the final classifier layer to fit the number of classes (5). This approach provides a good trade-off between accuracy and training time.  
  
Loss function: CrossEntropyLoss  
Optimizer: Adam with a learning rate that was tuned using SageMaker HyperparameterTuner.

# 4. Implementation

The implementation consists of the following steps:  
1. Data loading and preprocessing using torchvision datasets and transforms.  
2. Model initialization using EfficientNet-B3 with customized classification layers.  
3. Training logic that tracks the best model using validation loss.  
4. Evaluation on the test set for final performance metrics.  
5. Deployment using SageMaker endpoint and `predictor\_cls` to infer results via API.

# 5. Model Evaluation

The model's performance was evaluated using the following metrics:  
- Accuracy  
- Loss  
  
Model performance:  
- Best validation loss: 24.34  
- Final test accuracy: Approximately 4.9%, indicating potential underfitting.  
  
To improve performance, further tuning of the learning rate, unfreezing layers for fine-tuning, or adding data augmentation may be necessary.

# 6. Discussion

Challenges faced:  
- Limited performance gain using frozen layers—fine-tuning may help.  
- Difficulty in interpreting training logs due to low accuracy.  
  
Model monitoring rules like overfitting, vanishing gradients, and poor initialization were employed via SageMaker Debugger. Further monitoring and retraining are required to adapt to real-world data shifts.

# 7. Conclusion

EfficientNet-B3 was deployed on SageMaker for an image classification task. While the pipeline is complete and functional, results suggest further improvements are necessary to reach a production-ready performance level.  
  
Future work includes:  
- Further fine-tuning of model layers  
- Using a larger dataset or performing data augmentation  
- Exploring ensemble techniques or alternative architectures

8. Model Architecture Overview  
  
The model architecture is based on EfficientNet-B3, a convolutional neural network (CNN) known for balancing high accuracy and computational efficiency. The implementation utilizes transfer learning with the following structure:  
  
- Base Layers: The EfficientNet-B3 base layers pretrained on ImageNet are used and frozen during training to leverage learned features.  
- Classifier Head: Custom layers added on top include:  
 - A linear layer reducing feature dimensions from 1536 to 512.  
 - ReLU activation function.  
 - Dropout layer to prevent overfitting.  
 - Final linear layer mapping to 5 output classes.  
  
This architecture offers a strong feature extraction capability and adapts well to the inventory classification task with limited data.

9. Cost Analysis  
  
The model training was conducted on Amazon SageMaker using the following configuration:  
  
A screenshot of a phone

AI-generated content may be incorrect.  
  
Additional cost may incur from hyperparameter tuning jobs, S3 storage, or prolonged endpoint deployment. Debugger and profiler usage may also add minimal additional overhead.