

Project Report

Project Overview

Distribution centers play a crucial role of modern supply chain and logistics operations. These centers are responsible for receiving, storing, and shipping products to retail outlets, customers, or other distribution hubs. Inventory Monitoring is one of the main challenge distribution center faces during these operations which is essential to ensure that products are available for shipment in a timely manner, and that the right quantities are dispatched for customer orders. Inventory Monitoring involves tasks like sorting, tracking, and packaging products into different bins, boxes, or pallets. Performing these tasks manually is time consuming and error prone. We can automate most of tasks using robots, automated systems or ML agents. A critical challenge in such environments is ensuring accurate inventory tracking and verifying that the correct number of items are packed and shipped. This becomes even more complicated when bins contain multiple different objects. Manual counting of items within each bin is inefficient and error-prone, especially when dealing with large volumes of goods. We will be using machine learning as a solution for the manual counting of items in the bin

Problem Statement

One of the key tasks of inventory monitoring in distribution centers is the accurate counting of objects within bins. Bins may contain a variety of items, and having a precise count of each item is essential for tasks such as inventory tracking, quality control, and ensuring the accuracy of shipping consignments. An error in the number of items could lead to overstocking or understocking of products, delays in order fulfillment, or even customer dissatisfaction.

For example, when preparing a shipment, it is critical that the correct number of products are included in each bin, as discrepancies could lead to incomplete orders or incorrect shipments. Manually counting items inside a bin is labor-intensive and prone to errors, especially when bins contain hundreds or thousands of small items.

To address this issue, a solution is needed that can automatically count the number of objects in each bin during the inventory monitoring distribution center. An automated system that can detect and count the objects within bins will help reduce human error, improve accuracy and performance during inventory monitoring in distribution centers.

Project Report

Solution

Machine Learning can provide solution to this problem of counting the objects in bin. By training a machine learning model on images or sensor data of bins, it is possible to develop a system that can accurately count the number of objects in each bin. The model can be integrated with the automation systems, allowing it to make real-time inventory updates which increases the performance in distribution centers ultimately leading to customer satisfaction.

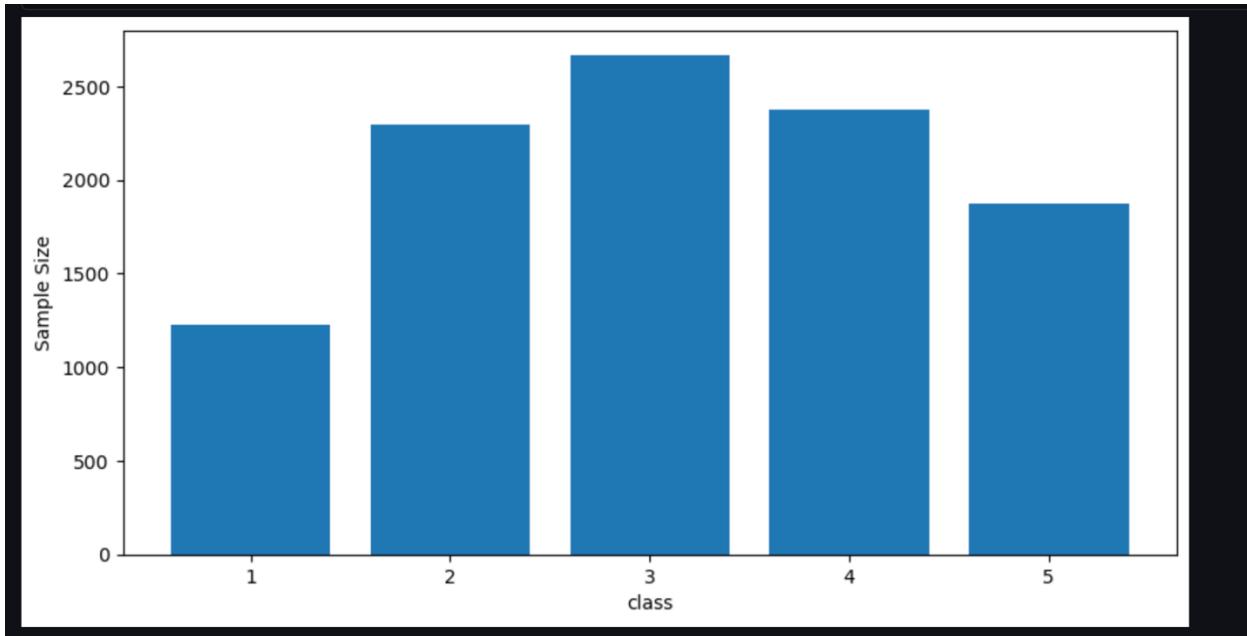
In this project, the goal is to build an object counting model using machine learning techniques, which can process images of bins, analyze the contents, and accurately count the number of items within. The machine learning model will be trained using labeled data that is images of bins with the corresponding object counts and deployed in the distribution center to assist with automated inventory tracking and shipment verification.

Data Exploration and Visualization

Amazon is one of leading supply chain and logistics company and has lot of distribution centers. For this project, we will use the Amazon Bin Image Dataset which consists of 500,000 images of bins, each containing one or more objects. These images were captured in various scenarios to reflect real-world conditions that might be encountered in distribution centers. Each image in the dataset contains an assortment of items, ranging from small objects to larger, more complex items. The variety of objects and bin configurations within the dataset is important, as it allows the model to generalize across different types of bins and object arrangements. Since there is lot of data I will be taking a smaller set of the entire data. Below is the bin count and the number of images containing that bin count.

```
{'1': 1228, '2': 2299, '3': 2666, '4': 2373, '5': 1875}
```

Project Report

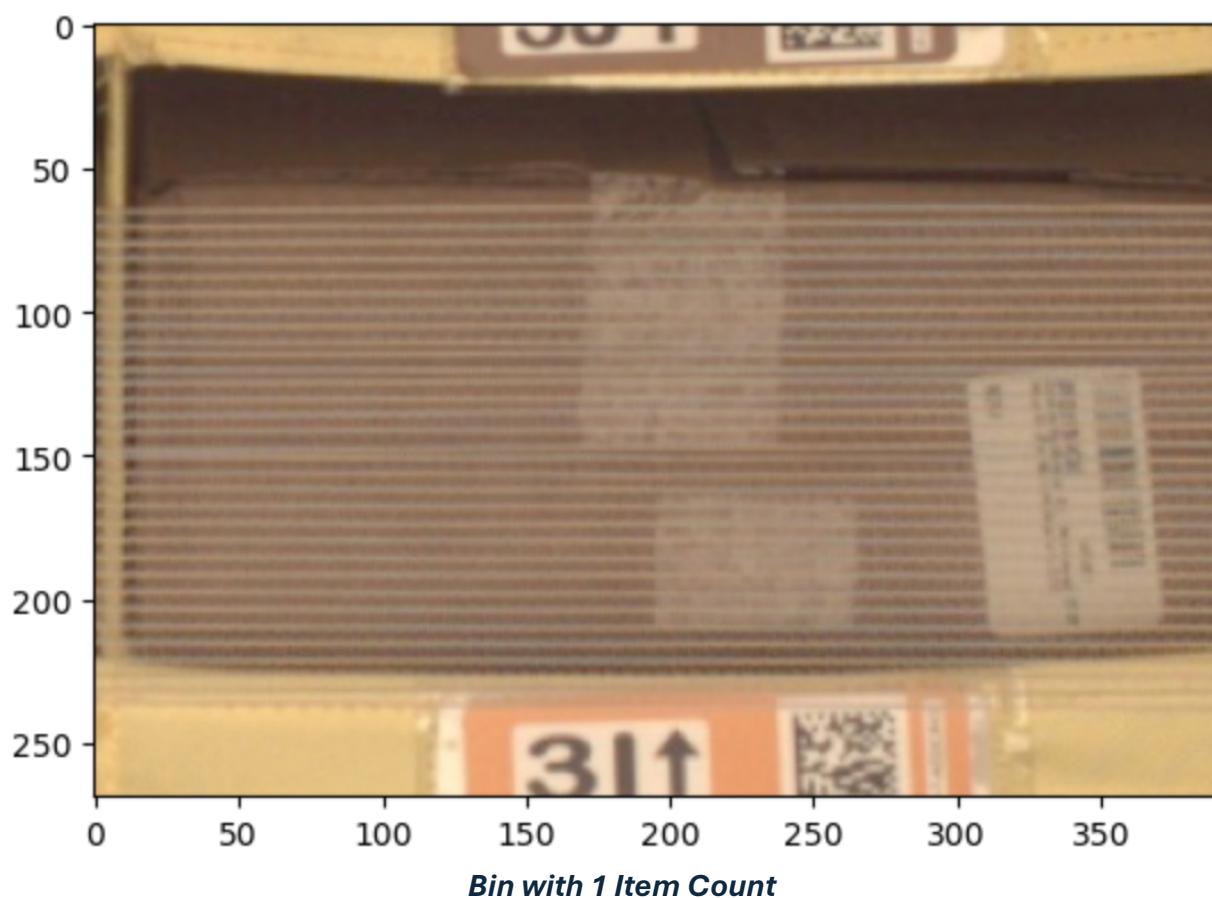


We have this file `file_list.json` (https://github.com/udacity/nd009t-capstone-starter/blob/master/starter/file_list.json) which contains the metadata images of all these images. We will use this meta data to download the images from the s3 bucket `aft-vbi-pds`

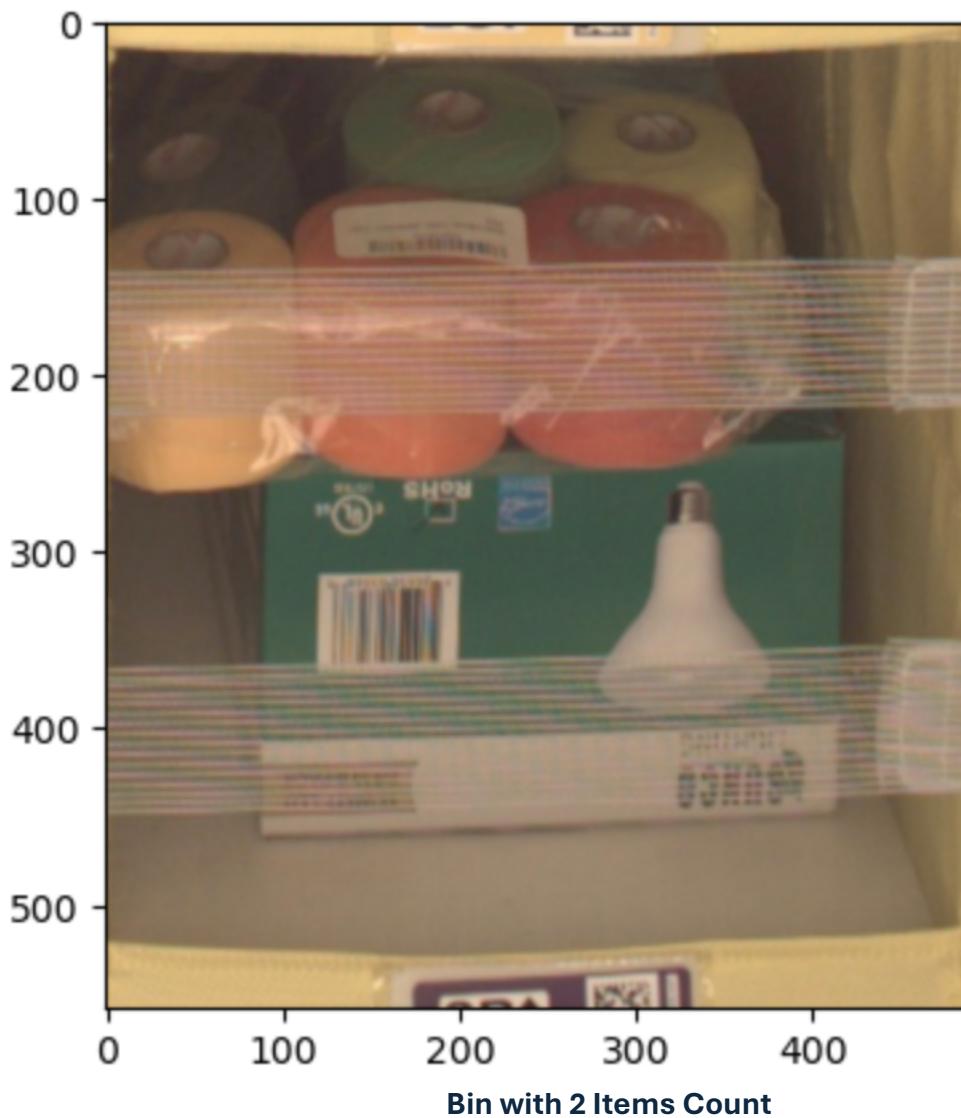
More info can be found here: <https://registry.opendata.aws/amazon-bin-imagery/>

Some of the images of the bins with different Item count

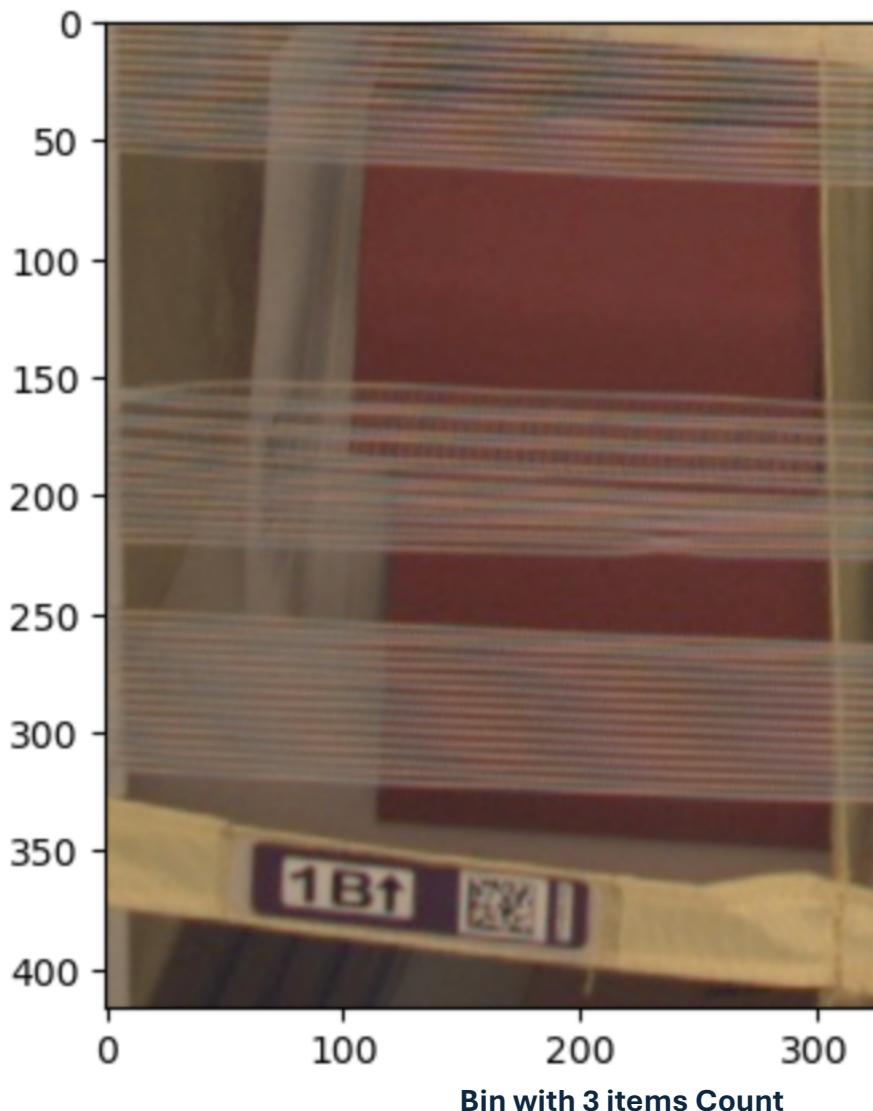
Project Report



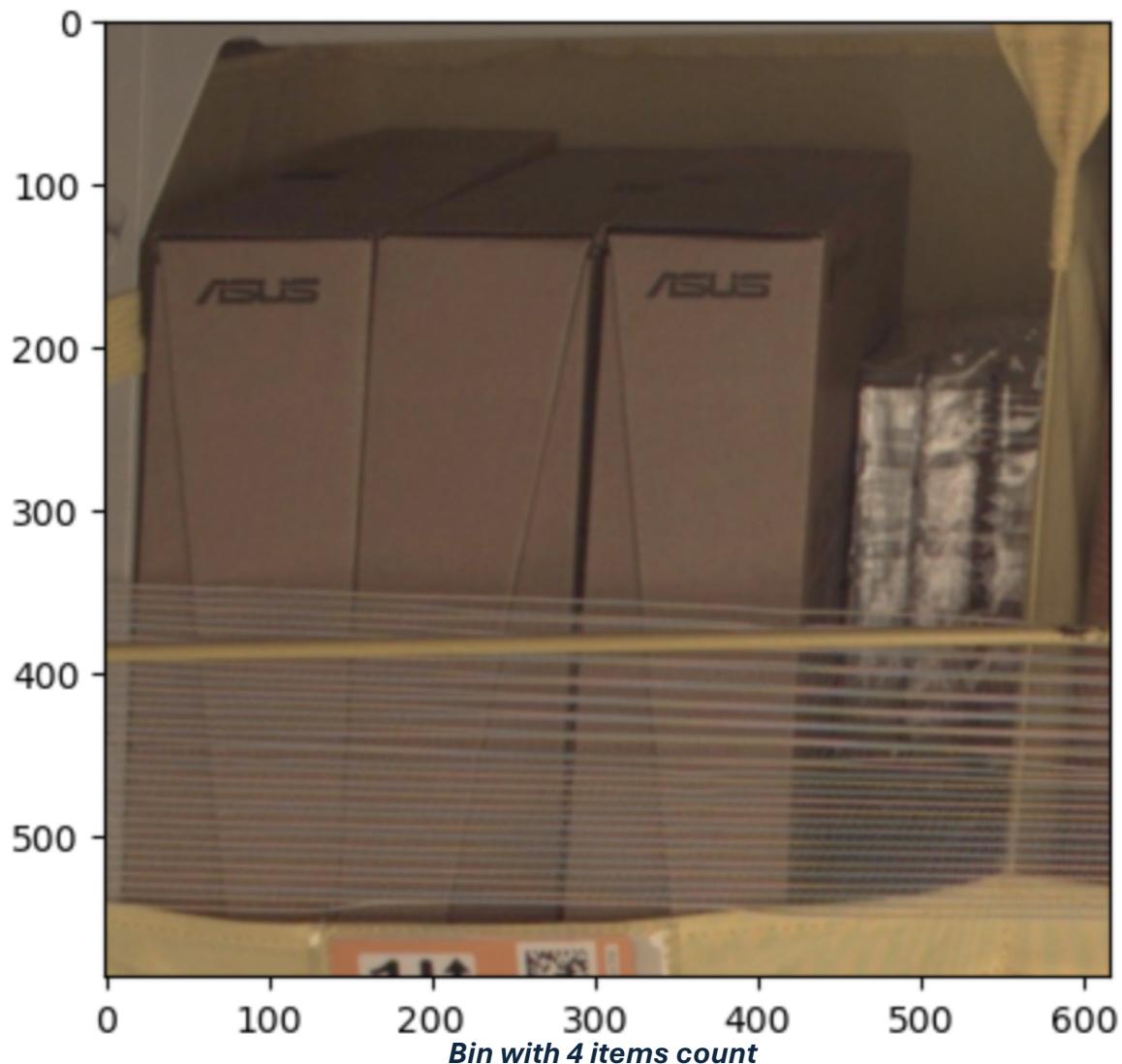
Project Report



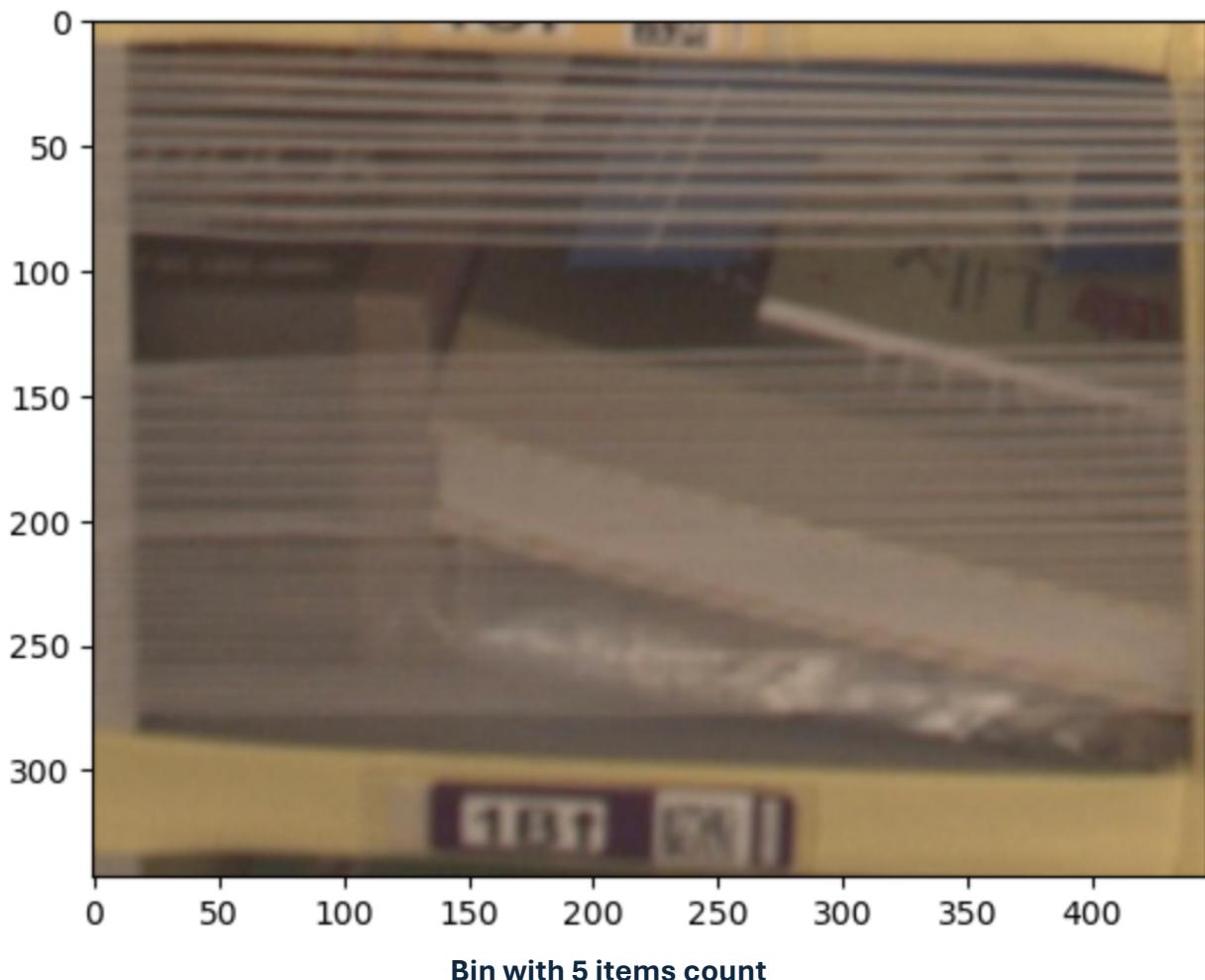
Project Report



Project Report



Project Report



The subset of data from amazon bin data set is further split into Train, Test and validation dataset. The crux here is we cannot randomly split the 20 percent entire subset to test data and this may result in choosing the entire 20 percent from one of the class of the images that contains the bin with specific number of items. This will result in bias while training and incorrect evaluation during test. We have the same issue while splitting the validation dataset. To overcome this Split is done on each class(bin images with specific item count) and aggregated to the final test and validation set.

Project Report

```
data moving for class 1
took random samples to test for size 245
took random samples to valid for size 196
Moving files to test_folder...
Moving files to valid_folder...
data moving for class 2
took random samples to test for size 459
took random samples to valid for size 368
Moving files to test_folder...
Moving files to valid_folder...
data moving for class 3
took random samples to test for size 533
took random samples to valid for size 426
Moving files to test_folder...
Moving files to valid_folder...
data moving for class 4
took random samples to test for size 474
took random samples to valid for size 379
Moving files to test_folder...
Moving files to valid_folder...
data moving for class 5
took random samples to test for size 375
took random samples to valid for size 300
Moving files to test_folder...
Moving files to valid_folder...
```

From the above logs we can see that split for train and validation data set is done on each class to prevent overfitting and underfitting of the model. So the final trainset will have 245 images which have bins with 1 item, 459 images which have bins with 2 items, 533 images which have bins with 3 items, 479 images which have bins with 4 items, 375 images which have bins with 5 items. That is 20 percent from each class of the total data set we considered. We uploaded the split data into s3 which can be used for model training.

Project Report

Objects (3) [Info](#) [More](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in more

[Find objects by prefix](#)

<input type="checkbox"/>	Name	Type	Last modified
<input type="checkbox"/>	test_data/	Folder	-
<input type="checkbox"/>	train_data/	Folder	-
<input type="checkbox"/>	valid_data/	Folder	-

Where each of the folder will have all the classes and the images that belong to them inside it

test_data/

[Objects](#) [Properties](#)

Objects (5) [Info](#) [Co](#)

Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3](#) more

[Find objects by prefix](#)

<input type="checkbox"/>	Name	Type
<input type="checkbox"/>	1/	Folder
<input type="checkbox"/>	2/	Folder
<input type="checkbox"/>	3/	Folder
<input type="checkbox"/>	4/	Folder
<input type="checkbox"/>	5/	Folder

Project Report

Benchmark Model:

Image Classification Based On CNN will be taken as Base Model. It is CNN model for image classification with good results.

Elngar AA, Arafa M, Fathy A, Moustafa B, Mahmoud O, Shaban M, Fawzy N. Image classification based on CNN: a survey. Journal of Cybersecurity and Information Management. 2021 Apr;6(1):18-50.

Model

To address the challenge of counting objects in bins within a distribution center, I am building an efficient machine learning solution based on a pretrained neural network. Instead of training a model from scratch, which can be computationally expensive and time-consuming, I will leverage the power of transfer learning. By fine-tuning a pretrained model, I can adapt an already trained neural network to our specific problem of object counting, saving significant development time while achieving high accuracy.

The core of this solution involves modifying the architecture of a pretrained neural network by adjusting the layers to better suit the object counting task. Specifically, I will focus on replacing or fine-tuning the final layers of the network to ensure it can output the correct number of objects for each bin. This approach will allow the model to learn not just to recognize individual items, but to estimate the total count of objects present in the bin based on visual features. I will also be focusing on to find the right hyper parameters for the model.

Using this strategy, the model will take input images or data related to the bins, process the information through the modified neural network, and output the predicted object count. By leveraging AWS Sage Maker, I will streamline the model training, validation, and deployment process, ensuring an end-to-end solution that is scalable, reliable, and ready for integration into real-world distribution center operations.

This solution demonstrates a practical application of transfer learning and neural network fine-tuning, while also showcasing how pretrained models can be adapted and optimized to solve domain-specific problems in a more efficient and effective manner.

Project Report

Environment & Platform:

- AWS Sage maker – to develop, train, tune and deploy
- AWS S3 – for storage

Algorithms:

- Torch
- Residual Network - convolutional neural network (Base Model)

Hyper parameter Tuning:

Hyper parameter Tuning plays a crucial role in model development as it helps maximize the performance of your machine learning model by finding the best configuration of hyperparameters. This is done using the HyperparameterTuner of Sage maker. Three parameters that were tunes are learning Rate, batch size and epochs.

```
hyperparameter_ranges = {
    "lr": ContinuousParameter(0.001, 0.1),
    "batch_size": CategoricalParameter([16, 32, 64, 128]),
    "epochs": CategoricalParameter([5, 10])
}
```

Below are the four training jobs that were run using the hyperparameter tuning

	batch_size	epochs	lr	TrainingJobName	TrainingJobStatus	FinalObjectiveValue	TrainingStartTime	TrainingEndTime	TrainingElapsedSeconds
2	"128"	"5"	0.058185	pytorch-training-241114-1958-002-dc2fa01c	Completed	1.596853	2024-11-14 19:59:44+00:00	2024-11-14 20:12:02+00:00	738.0
3	"64"	"10"	0.088724	pytorch-training-241114-1958-001-390d8933	Completed	1.583377	2024-11-14 19:59:43+00:00	2024-11-14 20:18:18+00:00	1115.0
0	"128"	"5"	0.051661	pytorch-training-241114-1958-004-c1653c5a	Completed	1.537071	2024-11-14 20:18:30+00:00	2024-11-14 20:25:50+00:00	440.0
1	"16"	"5"	0.012927	pytorch-training-241114-1958-003-d5a7fe3c	Completed	1.528551	2024-11-14 20:13:54+00:00	2024-11-14 20:21:15+00:00	441.0

Using the hyperparameters of the best model we train the final model that will be deployed.

Project Report

Training jobs Info						
		Creation time	Duration	Job status	Warm pool status	Time left
<input type="radio"/>	pytorch-training-2024-11-14-21-53-57-771	11/14/2024, 4:53:58 PM	-	InProgress	-	-
<input type="radio"/>	pytorch-training-2024-11-14-20-52-15-676	11/14/2024, 3:52:16 PM	13 minutes	Completed	-	-
<input type="radio"/>	pytorch-training-241114-1958-004-c1653c5a	11/14/2024, 3:18:26 PM	7 minutes	Completed	Terminated	-
<input type="radio"/>	pytorch-training-241114-1958-003-d5a7fe3c	11/14/2024, 3:13:50 PM	7 minutes	Completed	Terminated	-
<input type="radio"/>	pytorch-training-241114-1958-002-dc2fa01c	11/14/2024, 2:59:00 PM	13 minutes	Completed	Reused	-
<input type="radio"/>	pytorch-training-241114-1958-001-390d8933	11/14/2024, 2:58:58 PM	19 minutes	Completed	Reused	-
<input type="radio"/>	pytorch-training-241114-0133-004-c2bd44e5	11/13/2024, 8:54:26 PM	14 minutes	Completed	Terminated	-
<input type="radio"/>	pytorch-training-241114-0133-003-773552fa	11/13/2024, 8:54:24 PM	13 minutes	Completed	Terminated	-
<input type="radio"/>	pytorch-training-241114-0133-002-b74cd6fe	11/13/2024, 8:33:25 PM	19 minutes	Completed	Reused	-
<input type="radio"/>	pytorch-training-241114-0133-001-78e8bf2	11/13/2024, 8:33:23 PM	19 minutes	Completed	Reused	-

Evaluation Metrics

As this is a Classification Model, we will use Accuracy and Root Mean Squared Error to evaluate this model.

Accuracy=Total number of predictionsNumber of correct predictions

Formula for RMSE:

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Where:

- n is the number of observations (or test samples).
- y_i is the true count for the i^{th} observation.
- \hat{y}_i is the predicted count for the i^{th} observation.

Calculated the Accuracy for the model evaluation. We can see it logs

Project Report

The accuracy when compared to benchmark model is far less but can be improved with increasing the training data. The base model is just for recognizing an object where this model counts the items in the image. So it little complicated considered to base model.

```
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:15:27,798, pathname: train.py, message: for epoch= 0 load= train, loss and accuracy are epoch_loss=1.9233093203959202, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:15:42,717, pathname: train.py, message: for epoch= 0 load= valid, loss and accuracy are epoch_loss=1.5441920822290978, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:16:41,176, pathname: train.py, message: for epoch= 1 load= train, loss and accuracy are epoch_loss=1.5985223522310354, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:16:56,055, pathname: train.py, message: for epoch= 1 load= valid, loss and accuracy are epoch_loss=1.5667759590051786, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:17:54,171, pathname: train.py, message: for epoch= 2 load= train, loss and accuracy are epoch_loss=1.5881166801300037, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:18:09,221, pathname: train.py, message: for epoch= 2 load= valid, loss and accuracy are epoch_loss=1.6259217233697836, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:19:08,074, pathname: train.py, message: for epoch= 3 load= train, loss and accuracy are epoch_loss=1.550403799304018, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:19:23,128, pathname: train.py, message: for epoch= 3 load= valid, loss and accuracy are epoch_loss=1.5583122449409754, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:20:22,635, pathname: train.py, message: for epoch= 4 load= train, loss and accuracy are epoch_loss=1.5318043778897117, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:20:37,924, pathname: train.py, message: for epoch= 4 load= valid, loss and accuracy are epoch_loss=1.5521682082999602, ...
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:20:37,924, pathname: train.py, message: training done successfully
▶ 2024-11-14... 'levelName:'INFO, 'ts':2024-11-14 20:20:57,513, pathname: train.py, message: Testing Accuracy: 28.906999041227227, Testing Loss: 1.528551061338569
```

Deployment

The trained model with the best hyperparameters is deployed and created an endpoint which can used for the prediction.

The screenshot shows the Amazon SageMaker Endpoints page. At the top, there's a navigation bar with 'Amazon SageMaker' and a back arrow, followed by 'Endpoints'. Below the navigation is a search bar labeled 'Search endpoints'. To the right of the search bar are four buttons: a refresh icon, 'Update endpoint', 'Actions' (with a dropdown arrow), and a prominent orange 'Create endpoint' button. Underneath these buttons is a small icon with a question mark and a circular arrow. The main area is a table titled 'Endpoints'. The table has columns: 'Name', 'ARN', 'Creation time', 'Status', and 'Last updated'. There is one row in the table, representing the endpoint 'pytorch-inference-2024-11-14-21-28-54-741'. The ARN column shows the full ARN: 'arn:aws:sagemaker:us-east-1:523060554049:endpoint/pytorch-inference-2024-11-14-21-28-54-741'. The Creation time is '11/14/2024, 4:28:55 PM'. The Status is 'InService' with a green circular icon. The Last updated column shows '11/14/2024, 4:32:43 PM'. At the bottom of the table, there are navigation arrows for '1' and icons for 'Edit' and 'Delete'.

During a inference call the input object need some transformation to be used as a model input so the inference call has this transformation step embedded
transforms.Compose([
transforms.Resize((224, 224)),

Project Report

```
transforms.ToTensor(),  
])
```

Design

