Capstone Proposal

Inventory Monitoring at Distribution Centers

Ahmed Al Qady

December 27, 2022

Project Overview

Huge warehouses like Amazon requires reliable automated systems that sort and distribute packages. These packages can hold multiple items, therefore an automated system that counts the items in each package is required. Such system needs to be reliable and be able to identify and count items correctly even if the input photo is not very clear. This capstone project is about developing such system.

Amazon Bin Image Dataset contains photos of packages with a metadata file that shows the number of items in the package. This dataset will be used to train a model that can count the number of items in each package. A system like this can be used to track inventory and make sure that delivery consignments have the correct number of items.

Problem Definition

The aim of this project is developing a machine learning system that counts number of items in a package based on a picture of the package's content so it can be used to track the warehouse's inventory and ensure that each package contains the correct number of items. The trained model need to have a known accuracy to be able to predict false positives.

Datasets & Inputs

This project will use only 10,000 images from Amazon Bin Image Dataset to keep the project size reasonable. Post-processing the dataset divides it into train, test and validation sets using ratios of 60%, 20%, 20% and the number of images per class will be similar.

Solution Statement

A machine learning model that counts number of items in a package based on an input image of the package's content.

Benchmark Model

ResNet-50 is a 50 layers deep convolution neural network, it has an image input size of 224x224, a pre-trained version of this network is available in AWS which was trained on more than a million images from the ImageNet dataset, therefore it learned rich feature representation for a wide range of images, so it will be used as a baseline model in this project.

Evaluation Metrics

Standard metrics like Cross Entropy Loss and precision (accuracy) will be used to asses the quality of the training process, these metrics measures and quantifies the solution and ensures its repeatability. Hyper parameter optimization will be used to ensure the training converges into expected results and that the dataset won't be over-fitted.

Project Analysis

Dataset Overview

Amazon Bin Image Dataset contains over 500,000 images along with metadata of packages with multiple items in it. To limit the training time and cost of training for this project, Udacity provided a Json file with about

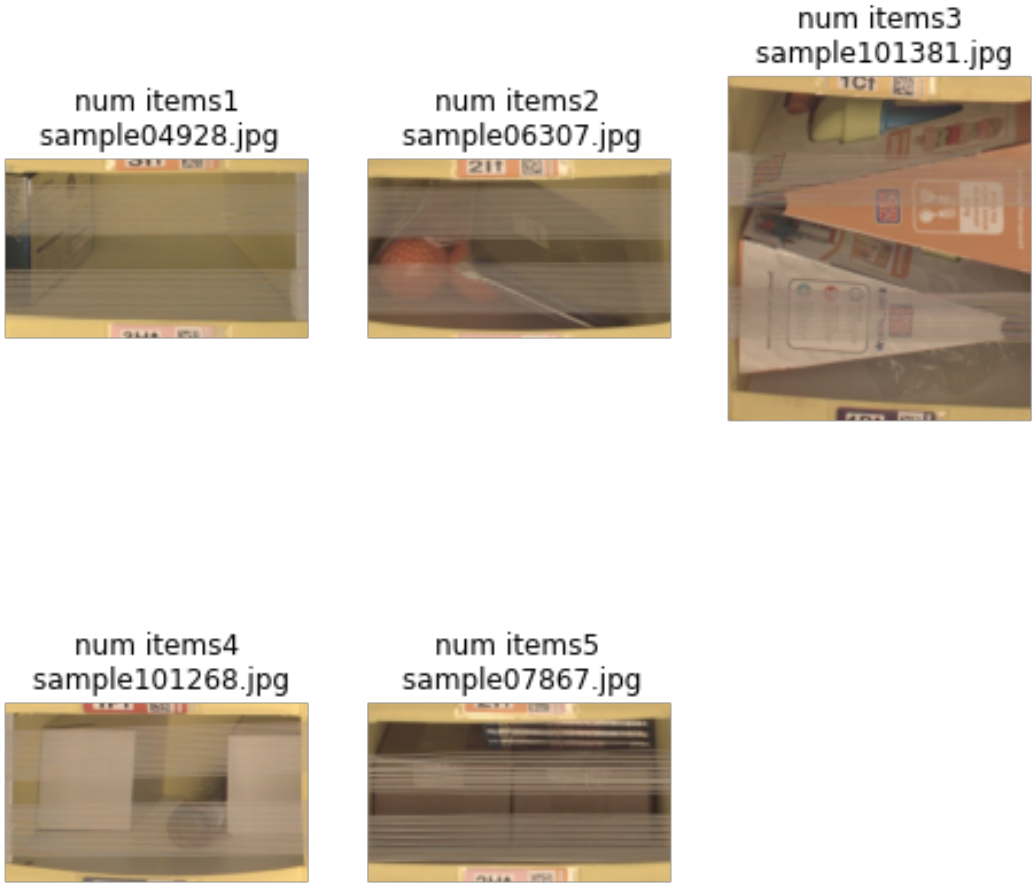
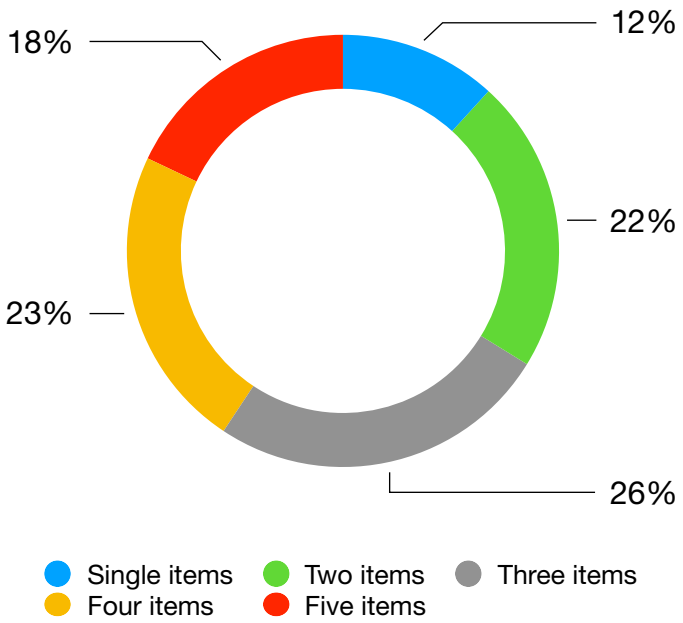


FIGURE 1: DATASET SAMPLE

FIGURE 2: CLASS DISTRIBUTION



10,441 images divided into 5 classes, each class has images of packages with (1,2,3,4,5) items in it.

Sample distribution of the dataset is as following:

- Single item: 1228 images.
- Two items: 2299 images.
- Three items: 2666 images.
- Four items: 2373 images.
- Five items: 1875 images.

Model Selection

This is an image classification project, so the best option is to use a pre-trained image classification network that is already able to process and classify image data. Resnet-50 was chosen as it is easily available in PyTorch models but, other image classification networks might work just as well.

Model Implementation

This project is developed in AWS using Sagemaker while minimizing the costs of the training process. The key steps of implementation are as following:

1. Data Preparation
   - Downloading data from a dataset
   - Preprocessing the data and dividing it into train, test and validation datasets
   - Uploading data to S3

2. Model Training
   - Hyper parameter tuning
   - Training the model and ensuring no training anomalies
   - Measuring KPI metrics

3. Model Deployment
   - Verifying model results
   - Assessment of the project quality, if not acceptable the training should be repeated

Data Preparation

For this project, the images dataset was divided into train, test and validation sets using the 60/20/20 rule. ResNet-50 accepts input images of 224x224 pixels in RGB format, so the dataset images were resized and vectorized to match the input of the network.

Data Preprocessing

Random image flipping was applied in the training process to ensure that mirrored images yield same results. Also random image cropping was applied to ensure differently scaled images yield same results, as the network should be able to count different sized items in the packages.

Images from Amazon Bin Image Dataset feature similar lighting conditions which results in a similar color space, so to reduce the overall cost of this project, normalization of the color space for the training images was not enabled.

Hyperparameters Tuning

hpo.py script executes a single epoch on 10% of the dataset to determine the best hyperparameters to be used when training the model.
- Learning rate was tuned for the range of (0.001, 0.1) and the optimal value found is 0.0019270438362405597
- Batch size was tuned for values of (32, 64, 128, 256, 512) and the optimal value found was 32

Name	Status	Objective metric value	Creation time	Training Duration
pytorch-training-221227-1249-010-91f3904d	Completed	100.3530044555664	Dec 27, 2022 12:49 UTC	8 minute(s)
pytorch-training-221227-1249-009-0c6ef03f	Completed	28.53260040283203	Dec 27, 2022 12:49 UTC	8 minute(s)
pytorch-training-221227-1249-008-0cbf78bd	Completed	1.753243088722229	Dec 27, 2022 12:49 UTC	8 minute(s)
pytorch-training-221227-1249-007-40df89d5	Completed	2.974428176879883	Dec 27, 2022 12:49 UTC	7 minute(s)
pytorch-training-221227-1249-006-e21535c3	Completed	56.932491302490234	Dec 27, 2022 12:49 UTC	7 minute(s)
pytorch-training-221227-1249-005-4aa09549	Completed	2.126978635787964	Dec 27, 2022 12:49 UTC	7 minute(s)
pytorch-training-221227-1249-004-8a7b5f0d	Completed	4.232886791229248	Dec 27, 2022 12:49 UTC	7 minute(s)
pytorch-training-221227-1249-003-fdfad4f9	Completed	1.6565487384796143	Dec 27, 2022 12:49 UTC	7 minute(s)
pytorch-training-221227-1249-002-11816428	Completed	1.640995979309082	Dec 27, 2022 12:49 UTC	7 minute(s)
pytorch-training-221227-1249-001-0361e0da	Completed	1.6112617254257202	Dec 27, 2022 12:49 UTC	7 minute(s)

FIGURE 3: HYPERPARAMETER TUNING JOBS

Model Training

Training procedure was initiated after determining the best hyperparameters for this project, file train.py provides the code used for training. The file is made to work from Sagemaker notebook and it can also be used as a standalone script. The training procedure finished in 1.36 hour using instance count of 10 and instance type of ml.m5.xlarge.

Name	Creation time	Duration	Job status	Warm pool status	Time left
pytorch-training-2022-12-27-13-04-39-930	Dec 27, 2022 13:04 UTC	an hour	Completed	-	-

FIGURE 4: MODEL TRAINING JOB

Training Results

Sagemaker Instance Training

Cross Entropy Loss Function is presented in the following figure.

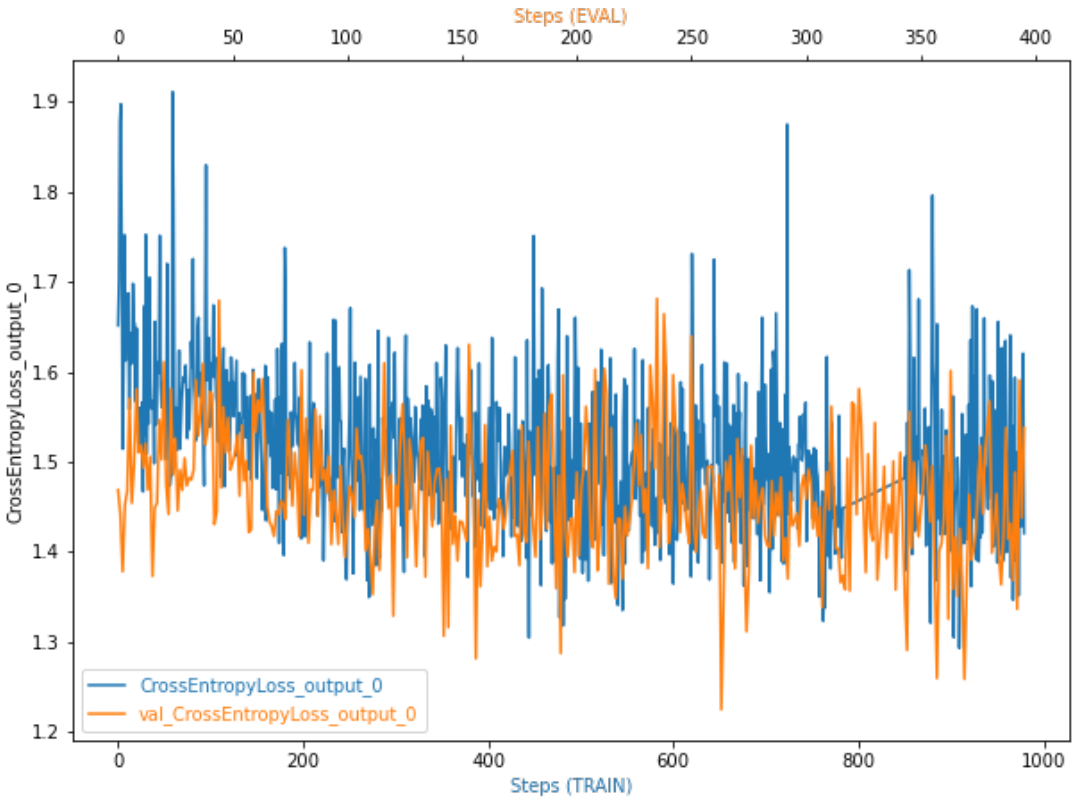


FIGURE 5: CROSS ENTROPY LOSS FUNCTION

Model Deployment

After successfully training the model, it was deployed to endpoint to be queried using a sample image. The script inference.py handles the model inference, it can be used from Lambda or Sagemaker.

After querying the model using the sample image shown in the following figure, the model correctly counted 3 items in the picture, which is the correct answer.

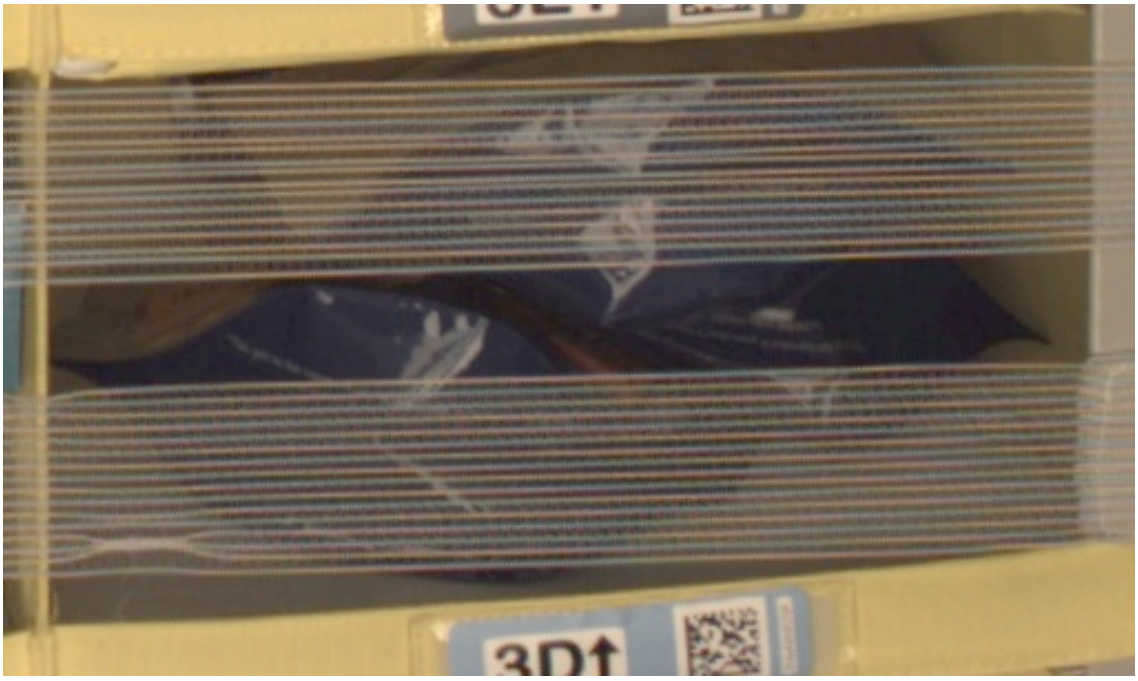


FIGURE 6: TEST IMAGE

Checking the training logs reveals the final accuracy of the trained model, which is 31%

▶	2022-12-27T16:29:01.218+02:00	INFO:__main__:Testing ...
▶	2022-12-27T16:29:01.218+02:00	DEBUG:__main__:Testing model
▶	2022-12-27T16:29:01.218+02:00	#015 0%| | 0/66 [00:00<?, ?it/s]#015 2%|| | 1/66 [00:03<03:2
▶	2022-12-27T16:29:01.218+02:00	INFO:__main__:Loss: 1.4360308570578366
▶	2022-12-27T16:29:01.218+02:00	INFO:__main__:Accuracy: 0.3168744007670182

FIGURE 7: TRAINING LOGS

Model Comparison

In order to benchmark the trained model and compare its results, another model is trained without hyperparameter tuning, this model was trained using instance count of 20 and instance type of ml.m5.2xlarge to speed things up. This new model also uses train.py script but without hyperparameter optimization.

After checking AWS CloudWatch logs, it appeared that the untuned model finished training on the second epoch due to growing loss function, and that it has an accuracy of 22%

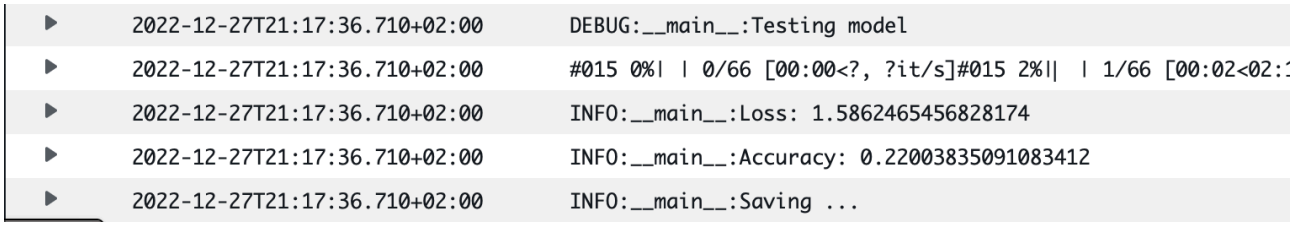


FIGURE 8: UNTUNED MODEL LOGS

These findings shows that the model with tuned hyperparameters performed better than the untuned model.

Future Improvements

To improve the model's accuracy which is about 31%, the model should be trained on the entire Amazon Bin Images dataset. This project was programmed to halt the training when the loss function is growing, a future improvement might be allowing the training to proceed even if the overall loss function is growing. Also there is an optimization technique called "annealing" which temporarily increases object entropy which allows it to decrease more than before, this technique can be used to improve the model.

Conclusion

Considering that this model was trained on only 5% of the Amazon Bin Images Dataset, its final accuracy of 0.31 is quite acceptable. A future improvement can be training the model on the entire dataset to achieve better accuracy.

Due to the low accuracy of this model, it would perform successfully on only 31% of the measurements and would fail 69% of the time, so this model can not be used to track the inventory at distribution centers, but future improvements can certainly make it achieve great and reliable performance.