AWS Machine Learning Engineer Nanodegree Program

Capstone Project Report

INVENTORY MONITORING AT DISTRIBUTION CENTERS

Lu Zhu

April 2023

DEFINITION

Overview

A long time before the industrial age, keeping track of things was mainly done by manually counting items. The earliest form of inventory management dates back over thousands of years in which people used tally sticks to count. The empowering of the distribution centers to automate their process through a digital workplace is the need of the hour to manage the demand and supply chain. During the time of pandemic, there is a huge demand for logistics services. Nowadays, objects are carried in bins and robots are used to move them in the distribution centers. To make sure that the delivery consignments have the correct number of items, a model that can count the number of objects in each bin and track inventory is essential. Therefore, this project aimed to train an image processing model to be able to count the number of objects in an imagery.

Problem Statement

The challenge is to count the numbers of items present in the bin, which are randomly positioned and in various sizes. Nevertheless, the possible numbers of items present in a bin are limited (less than five in the training dataset), and the background of the items in a bin should be the color of the bin rather than any random, complex backgrounds. These conditions can simplify the problem to some degree. By providing the appropriate training dataset we saved time in training the model.

Metrics

To solve the problem stated above, the model performance is the key to success. Because this model is a multi-class classification model, I used the entropy loss and the overall accuracy of the classification to evaluate the model. Additionally, related parameters (such as epoch number, train/test phase, accuracy of each batch in training and testing) are logged through the training process to monitor the training progress.

ANALYSIS

Data Exploration

To train the model I will use the Amazon Bin Image Dataset. It contains 500,000 images of bins that contain one or more items. There is a metadata file associated with each image, which contains information such as the number of items in the bin, the dimensions, and the types of items (Fig. 1). For this task, only a subset (10, 441 randomly selected images) of the dataset is used for the training. Because the model classifies the numbers of items in the bins (which is no more than five), only the numbers of items in the bins are extracted from the metadata. Please see the distribution of numbers of items in the bins in the randomly selected dataset (Fig. 2).

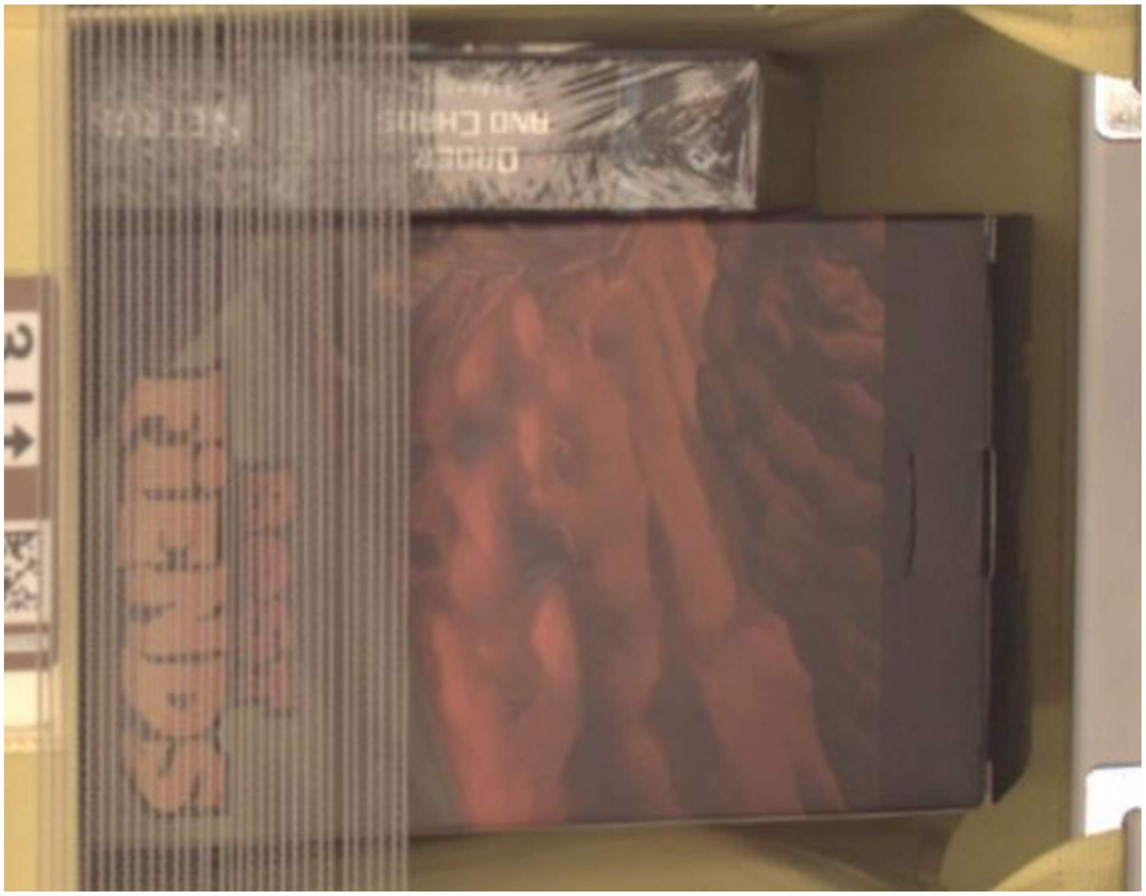


Figure 1. Example image of the dataset that is used to train and evaluate the model.

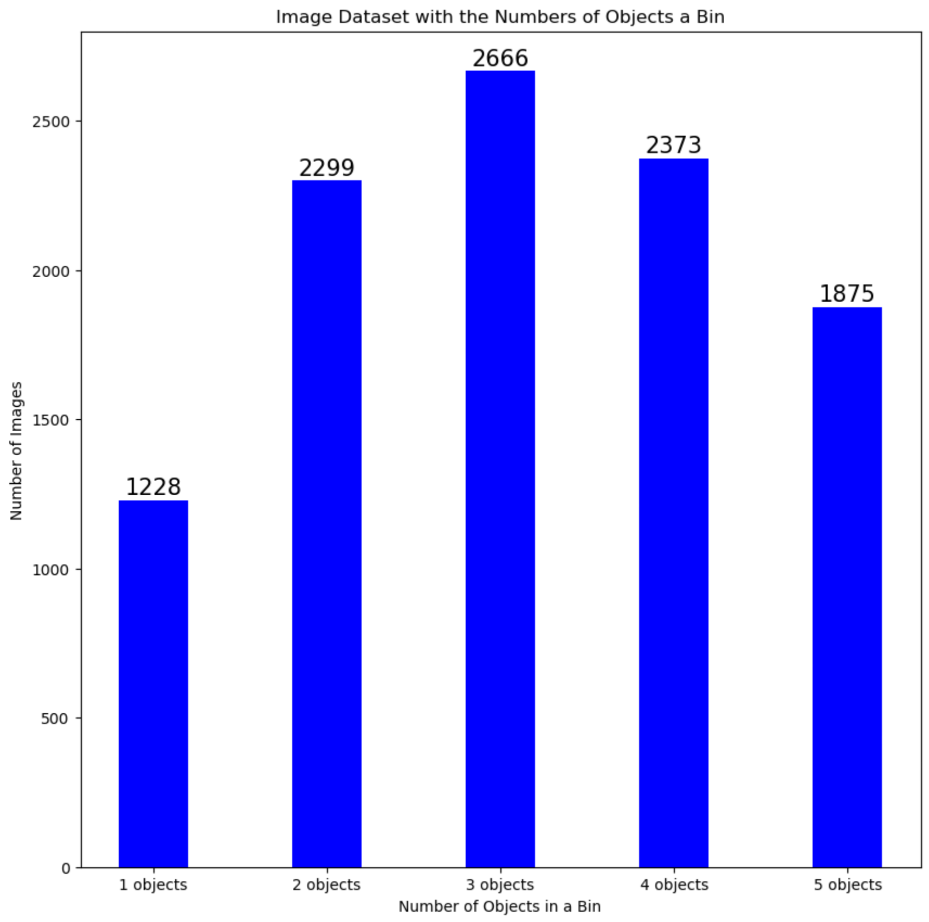


Figure 2. The distribution of numbers of objects in a bin for the dataset used to train and evaluate the model.

Algorithms and Techniques

This project used the pre-trained Convolution Neural Network – ResNet-50 to develop the image classification model. The Convolution Neural Network follows a hierarchical model which works on building a network, like a funnel, and finally gives out a fully connected layer where all the neurons are connected to each other, where graphic characteristics can be picked up and processed to give an output. Therefore, the Convolution Neural Network is fit for this project to perform the image classification.

The images are fed to train the model. A corresponding SageMaker instance was created to run Jupiter Notebook, and the data was fed from the S3 bucket. The training is a supervised learning process that the model learned to classify based on the numbers of items in the bins.

To improve the training efficiency, this model is also tuned to find out the best hyperparameters. Hyperparameter tuning jobs are run in SageMaker to tune the learning rate, batch size and epoch number. The Learning rate for the optimizer is identified as one of the hyperparameters which could help improve the training process outperform the pre-defined default. Batch Size could help optimizing the computation speed and convergence. The number of Epochs could help better train the model. The following values are the best hyperparameters resulted from the fine tuning and searching:

Learning Rate: 0.00796
Batch Size: 128
Epochs: 10

Benchmark

The detailed results of the benchmark model mentioned in the proposal (Verma et al, 2016) is not publicly available (I also tried to contact the author on ReaserchGate to get a copy but did not get a reply). Another repository, contributed by silverbottlep (Repository URL see reference) on GitHub provides a similar model to this project and uses the same dataset. Therefore, silverbottlep's project is used as the benchmark model for this project.

The benchmark model uses the same Amazon Bin Image Dataset. It used the entire image dataset (instead of a subset) to train and validate the model. The entire dataset contains images of various numbers of objects in the bins, which ranges from 0 object to 25 objects. Among which, 90% of the images contain less than 10 objects in a bin. Therefore, comparing to this project, the Benchmark model uses a larger dataset and with more classes to classify.

The benchmark model also used a Deep Convolutional Classification Network the ResNet-34 model for classification. The ResNet-34 model is of less layers than the ResNet-50 model that this project used. The benchmark model was trained with a batch size of 128 and reached its best validation accuracy 55.67% at 21 epoch and started to overfit afterwards (Fig. 4).

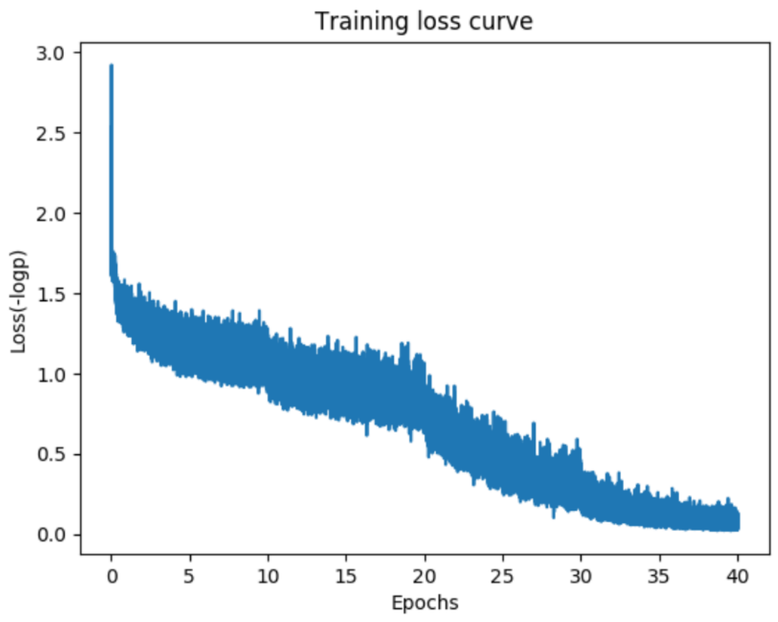


Figure 3. Training loss curve of the benchmark model. The training loss decreases as the training epoch number increases.

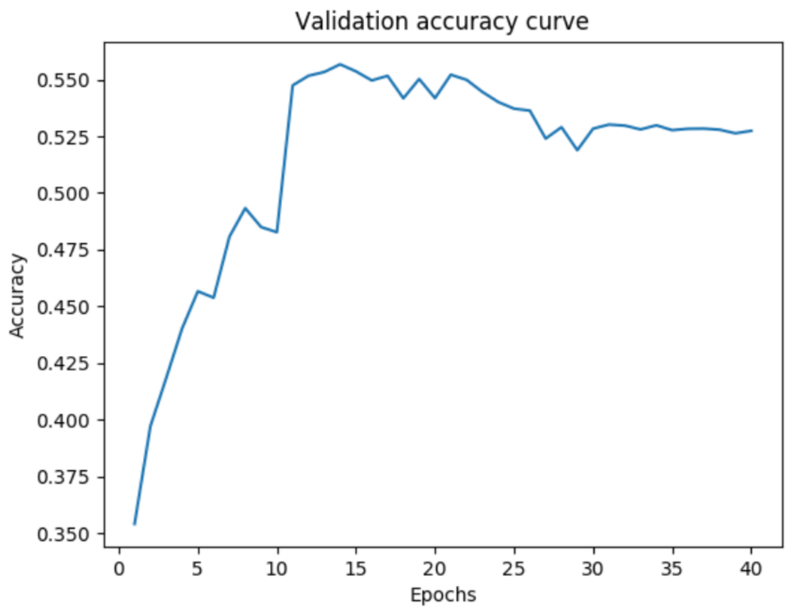


Figure 4. Validation accuracy curve of the benchmark model. The accuracy increases with a high rate at the beginning, then the increase rate decreases and slowly reaches to a peak, and finally the accuracy slowly decreases due to overfit.

METHODOLOGY

Data Preprocessing

The images are downloaded from the Amazon Bin Image Dataset via URL. Every image is converted into image bytes. To ensure that the images are of the same size, images are transformed and resized. All images are resized to 224x244 pixels using PyTorch transformers. The data loaders are created at the batch size for training, testing and validation.

Implementation

This project used the pre-trained Convolution Neural Network – ResNet-50 to develop the image classification model. The images which are pre-processed following the above described steps are fed to train the model. A corresponding SageMaker instance was created to run Jupiter Notebook. The training is a supervised learning process that the model learned to classify based on the numbers of items in the bins.

To improve the training efficiency, this model is also tuned to find out the best hyperparameters. Hyperparameter tuning jobs are run in SageMaker to tune the learning rate, batch size and epoch number. The Learning rate for the optimizer is identified as one of the hyperparameters which could help improve the training process outperform the pre-defined default. Batch Size could help optimizing the computation speed and convergence. The number of Epochs could help better train the model. After the hyperparameter tunning, the training was initiated with the learning rate of 0.00796, batch size of 128, and epochs of 10.

For this multi-class classification model, the entropy loss and the overall accuracy of the validation are used to evaluate the model. Additionally, related parameters (such as epoch number, train/test phase, accuracy of each batch in training and testing) are logged through the training process to monitor the training progress.

The implementation procedures are documented in the sagemaker.ipynb file in this archive.

Refinement

In this project, we proceeded training the model following the hyperparameter tuning job. Like discussed above, the hyperparameter tuning is supposed to enhance the training efficiency. Nevertheless, the performance of the model was not close to benchmark model (please see the detailed results in the following session). In future development to improve this model, there are two aspects worth considering: increasing the dataset volume to train the model and using gradually changing learning rates rather than a fixed learning rate during the training. These two aspects are the major differences between this project and the benchmark project. They are worth trying to achieve a comparable performance to the benchmark model.

RESULTS

Model Evaluation

The model finally reached an accuracy of 9% after 40 min's training (Table 1). The training of the model was initiated with the learning rate of 0.00796, batch size of 128, and epochs of 3. The cross-entropy loss of training gradually decreased at the very beginning but then continued to fluctuate up and down throughout the training without either decreasing or increasing trend. The cross-entropy loss of evaluation fluctuated up and down throughout the entire training without either decreasing or increasing trend (Fig. 5).

The accuracies for each epoch did not increase significantly like the benchmark model did in the beginning of its training. This may be because the initiation of the learning rate is not desirable. The benchmark model had a greater learning rate at the beginning of the training and lowered the learning rate gradually through the training.

The overall performance of this model is significantly lower than that of the benchmark model. This could be caused by the significantly smaller size of the training dataset and less training epochs. To improve the model performance, investing more computing resources and increasing the training dataset is also a necessary aspect to consider.

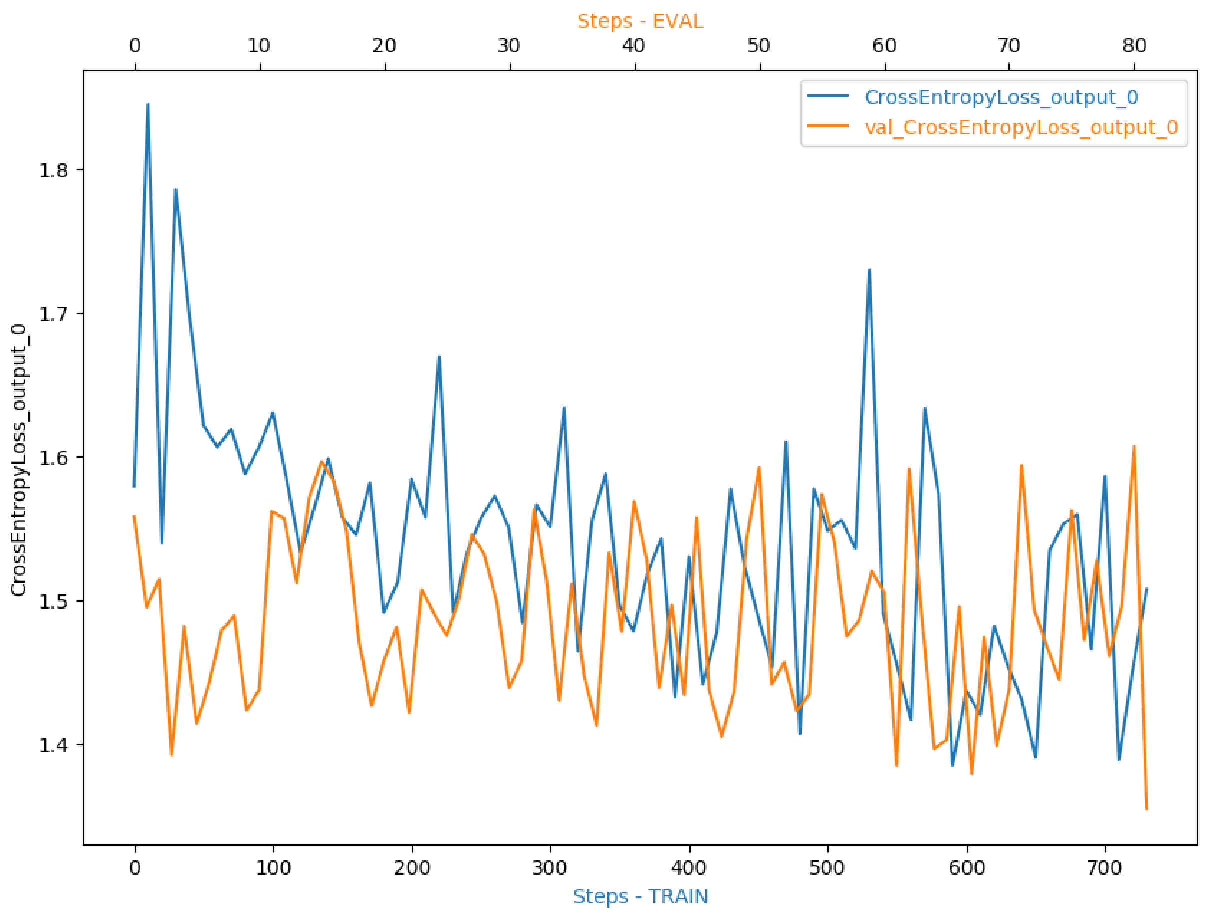
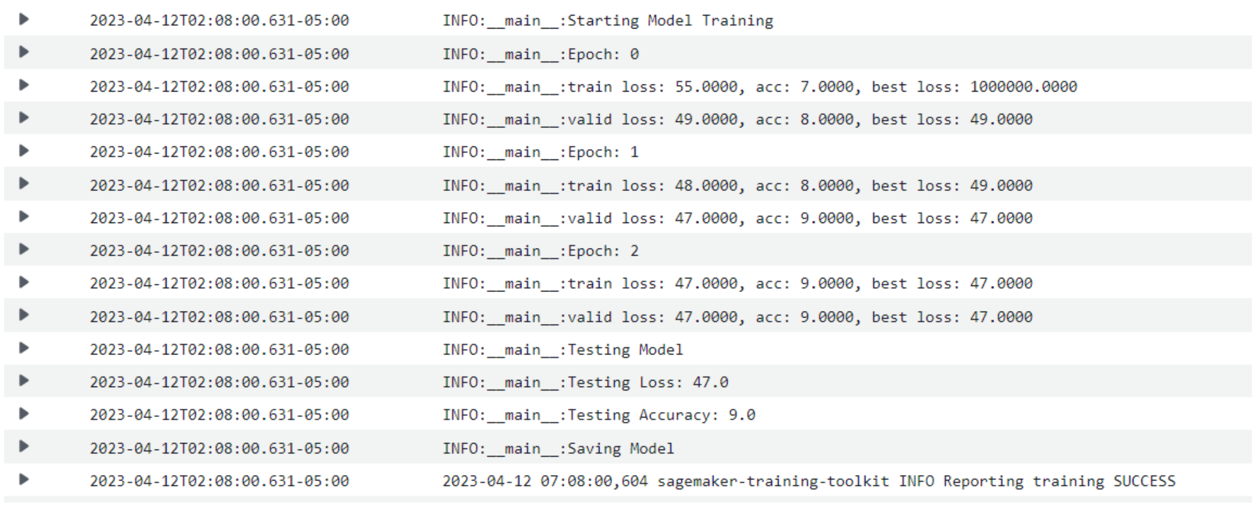


Figure 5. Cross Entropy Loss curve for training and evaluation.

Table 1. Training results of the model showing Entropy Loss and accuracy.



```
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Starting Model Training
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Epoch: 0
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:train loss: 55.0000, acc: 7.0000, best loss: 1000000.0000
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:valid loss: 49.0000, acc: 8.0000, best loss: 49.0000
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Epoch: 1
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:train loss: 48.0000, acc: 8.0000, best loss: 49.0000
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:valid loss: 47.0000, acc: 9.0000, best loss: 47.0000
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Epoch: 2
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:train loss: 47.0000, acc: 9.0000, best loss: 47.0000
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:valid loss: 47.0000, acc: 9.0000, best loss: 47.0000
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Testing Model
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Testing Loss: 47.0
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Testing Accuracy: 9.0
▶   2023-04-12T02:08:00.631-05:00    INFO:__main__:Saving Model
▶   2023-04-12T02:08:00.631-05:00    2023-04-12 07:08:00,604 sagemaker-training-toolkit INFO Reporting training SUCCESS
```

Justification

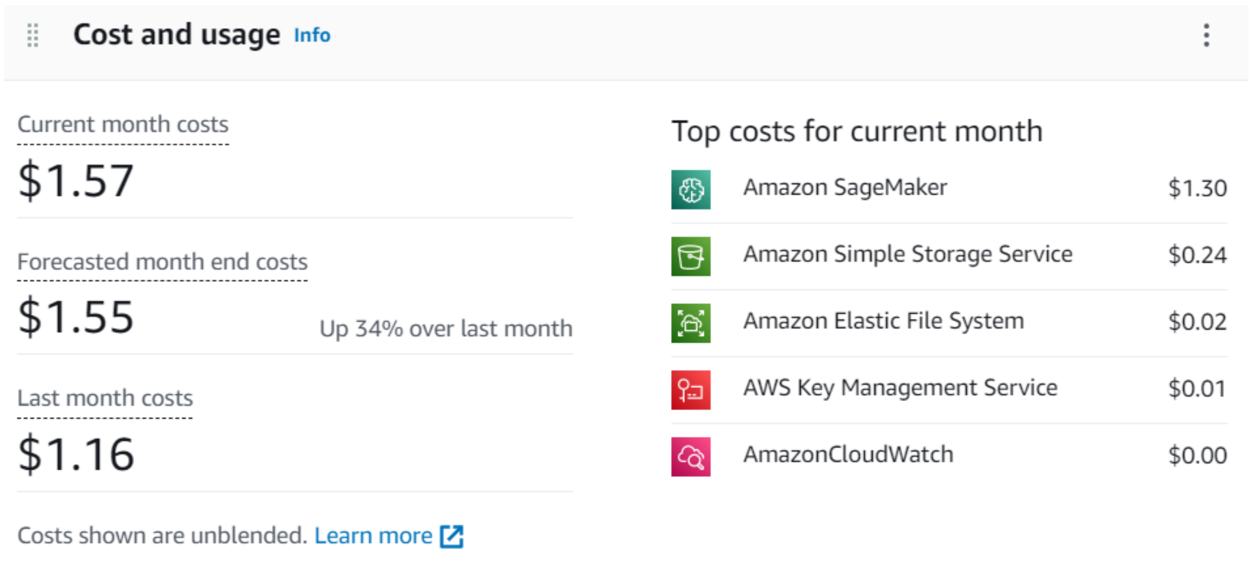
The performance of this model is not comparable to, and less ideal than that of the benchmark model. This could be caused by the significantly smaller size of the training dataset and less training epochs. To improve the model performance, investing more computing resources and increasing the training dataset is also a necessary aspect to consider. Additionally, the benchmark model used a gradually decreasing learning rate rather than a fixed learning rate throughout the training process. This is a good strategy of training and could be the reason that the entropy loss decreased at a very high rate at the beginning of training in the benchmark model (Fig. 3).

Overall, despite the fact that the performance still needs to improve, this model provides a preliminary solution to the problem. Within the provided implementation scale, the model has potential to be improved with more computation resources, larger datasets, and gradual changing learning rate initiation.

CONCLUSIONS

This project provides a preliminary solution to the image classification of the numbers of items in the bins for distribution centers. The model also shows the potential to be improved to better performance with little more investment in computational resources and better training strategy. More importantly, this project demonstrated the workflow of how to use machine learning to tackle a close-to-real-world problem.

The cost incurred on the AWS platform is shown in Figure 6. The overall cost is $1.57. If a spot instance was used for the training, the cost would have been less than the current incurred costs. SageMaker is very affordable for small businesses and individual researchers or students to explore machine learning solutions.


Cost and usage Info

Current month costs
$1.57

Forecasted month end costs
$1.55 Up 34% over last month

Last month costs
$1.16

Costs shown are unblended. Learn more

Top costs for current month

Amazon SageMaker	$1.30
Amazon Simple Storage Service	$0.24
Amazon Elastic File System	$0.02
AWS Key Management Service	$0.01
AmazonCloudWatch	$0.00

Figure 6. Cost and usage in AWS account for this project.

REFERENCES

Verma, N.K., Sharma, T., Rajurkar, S.D. and Salour, A., 2016, October. Object identification for inventory management using convolutional neural network. In 2016 IEEE Applied Imagery Pattern Recognition Workshop (AIPR) (pp. 1-6). IEEE.

Benchmark model Repository URL: https://github.com/silverbottlep/abid_challenge