Amazon Inventory Reconciliation Using AI

1st Pablo Rodriguez Bertorello
Computer Science
Stanford University
Palo Alto, California
prodrigu@stanford.edu

2nd Sravan Sripada
Computer Science
Stanford University
Palo Alto, California
sravansr@stanford.edu

3rd Nutchapol Dendumrongsup
Computational & Mathematical Engineering
Stanford University
Palo Alto, California
nden@stanford.edu

***Abstract*—Inventory management is critical to Amazon's success. Thus, the need arises to apply artificial intelligence to assure the correctness of deliveries. The paper evaluates Convolutional Neural Networks for inventory reconciliation. The network detects the the number of items being carried, relying only on a photo of the bin. Convolutional performance is evaluated against Support Vector Machines, both non-linear and linear. Our Convolutional method performed over 3x better than random, and over 75% better than our best Support Vector classifier.**

***Index Terms*—Convolutional Neural Network, Deep Learning, Support Vector Machine, Radial Basis Function, Amazon Bin Image Dataset**

I. Introduction

Amazon Fulfillment Centers are bustling hubs of innovation that allow Amazon to deliver millions of products to over 100 countries worldwide. These products are randomly placed in bins, which are carried by robots.

Occasionally, items are misplaced while being handled, resulting in a mismatch: the recorded bin inventory, versus its actual content. The paper describes methods to predict the number of items in a bin, thus detecting any inventory variance. By correcting variance upon detection, Amazon will better serve its customers.

Specifically, the input to our model is a raw color photo, of the products in a bin. To find the best solution to the inventory mismatch problem, Amazon published the Bin Image Dataset, which is detailed in Section II.

The output of a model is the bin's predicted quantity, the number of products in the image.

While we started with linear methods, the quest for model performance lead us to non-linear algorithms, and ultimately to convolutional deep learning. Section III summarizes each algorithms applied. They include:

- Logistic Regression and Classification Trees, summarized in Section III-A and Section III-B
- Support Vector Machines: linear kernel, polynomial kernel, radial kernel. The algorithms are summarized in Section III-C
- Convolutional Neural Network: ResNet, cross-entropy loss function, with learning rate optimizations. The algorithm summary in Section III-D

Section IV summarizes performance resuts. With Convolutional Neural Networks we were able to achieve an overall accuracy exceeding 56%. This is over 60% better than with Support Vector Machines. For the latter, we attained accuracy of over 30%. We operated on a reduced dataset, for bins that contained up to 5 products, for a random baseline probability of 16.6%.

We are among the first to publish results on Amazon's Bin Image Dataset. Prior art by Eunbyung Park of the University of North Carolina at Chapel Hill [2] applied Deep Convolutional Classification (ResNet 34). It achieved 55.67% accuracy.

II. Dataset and Features

A. Input Data Set

The data set contains 535,234 images, which contain 459,476 different product skews, of different shapes and sizes.

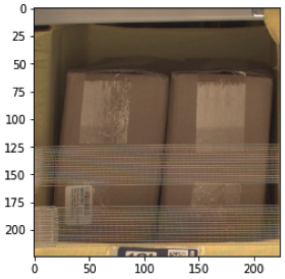


Fig. 1: Sample Image

Each image/metadata tuple corresponds to a bin with products. The metadata includes the actual count of objects in the bin, which is used as the label to train our model.

We worked with the subset of 150K images, each containing with up to five products. We split this as follows: 70% training, 20% validation, and 10% test.

B. Data Engineering

Before model training, images were normalized:

1) Re-sized to 224x224 pixels
2) Tansformed for zero mean, and unit variance
3) For convolutional models, the dataset was augmented with horizontal flips of every image

C. Feature Engineering: Blobs

We explored the Blob features extraction. Blobs are bright on dark or dark on bright regions in an image. All the bins in which items are placed are similar and if items are

present in the bin, the idea is to make an attempt to create features assuming items in the bin are relatively bright on the backgrounds. We used Laplacian of Gaussian approach, it computes the Laplacian of images with successively increasing standard deviation and stacks them up in a cube. Blobs are local maximas in this cube. Note the yellow circles in Figure 2

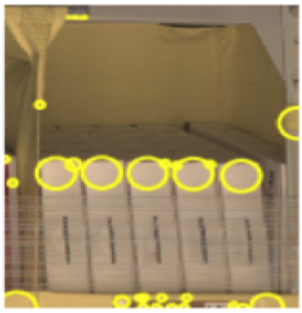


Fig. 2: Blob Features

D. Feature Engineering: Histogram of Oriented Gradients

In addition, Histogram of Oriented Gradients (HOG) is explored because objects in the images tend to have dominant orientations. The distributions of directions of pixels gradients are used as features. Unfortunately, the tape wrapping the products in the bin skews the HOG diagram. We attempted to remedy this by pre-processing the images for edge-detection. Still, HOG under-performed.

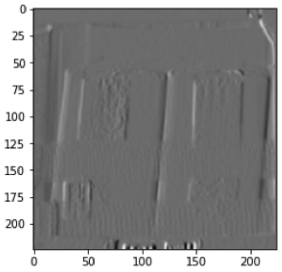


Fig. 3: HOG Features

III. Methods

A. Logistic Regression

The probability that each observation is classified to each class is defined as a logistic function as follow

$$p(X) = \frac{e^{(\beta_0+\beta_1 X_1+\ldots+\beta_p X_p)}}{1+e^{(\beta_0+\beta_1 X_1+\ldots+\beta_p X_p)}} \tag{1}$$

The observation is assigned to the class in which it has highest probability.

B. Classification Tree

In a classification tree, each observation belongs to the most commonly occurring class of training observations in the region to which it belongs. We use the Gini index, of which measures total variance across the K classes, as the loss function. Gini index is defined by

$$G = \sum_{k=1}^{K} \hat{p}_{mk}(1-\hat{p}_{mk}) \tag{2}$$

C. Support Vector Machines

The support vector machine (SVM) is an extension of the support vector classifier that results from enlarging the feature space in a specific way, using kernels. Feature space is enlarged in order to accommodate a non-linear boundary between the classes. The kernel approach enable an efficient computational approach for SVM. We have attempted several kernel types as follow

- Linear kernel

$$K(x_i, x_{i'}) = \sum_{j=1}^{P} x_{ij}x_{i'j} \tag{3}$$

- Polynomial kernel

$$K(x_i, x_{i'}) = (1+\sum_{j=1}^{P} x_{ij}x_{i'j})^d \tag{4}$$

- Radial kernel

$$K(x_i, x_{i'}) = exp(-\gamma \sum_{j=1}^{P}(x_{ij}-x_{i'j})^2) \tag{5}$$

D. Convolutional Neural Networks

ResNet [4] consists of a series of convolutional filters followed by a fully connected layer. Deeper networks usually suffer with vanishing gradient issues and ResNet attempts to solve this problem by stacking identity mappings.

1) Classifier and Loss Function: Softmax layer (σ) and Cross entropy loss (CEL) function were used since we are solving multi class classification problem

$$CEL = -\sum_{c=1}^{M} y_{o,c}\log(p_{o,c}) \tag{6}$$

$$\sigma_j = \exp z_j / \sum_{k=1}^{K} \exp z_k \tag{7}$$

2) Learning Rate Finder: Learning rate determines the step size of the update and is one of the key hyper-parameters to training a network. For some of our experiments, we set the learning rate based on an approach introduced in the paper "Cyclical Learning Rates for Training Neural Networks" Smith, and Leslie N [7]. Fig 4 depicts the approach for one of our experiments with ResNet34. Essentially, this algorithm runs one epoch on the network increasing learning rate every few iterations. Eventually, the learning rate producing the

steepest reduction in validation loss is picked. This eliminates the need to rely on experimentation to find the best learning rate.

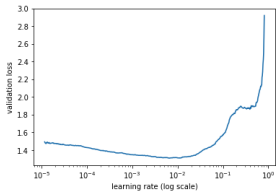


Fig. 4: Learning Rate Finder - Loss

3) Optimization Algorithms:

a) Stochastic Gradient Descent (SGD) and Stochastic Gradient Descent with Restarts (SGDR): Training CNNs involve optimizing multimodal functions. SGDR helps reset learning rate every few iterations so that gradient descent can pop out of a local optimum and find its way to global minima, an idea shown to work effectively in the paper Loshchilov, et al. [5]. Fig 5 shows SGDR with cosine annealing.

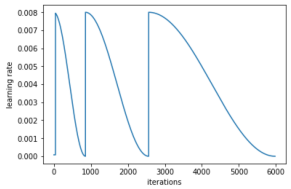


Fig. 5: SGDR Cosine Annealing

b) Adam: Adam is an algorithm that is computationally efficient, has little memory requirements, and is well suited for training deeper ResNets for Images, which are large in terms of data and parameters In the paper Kingma, et al [6], results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods.

IV. Experiments

The performance of the best methods, as well as the rationale leading to identifying them is outlined below.

Given the nature of the data set, we expect the decision boundary to be highly non-linear.

A. Multi-Class Classification

Several multi-class classifiers were explored with raw pixel data. Figure I shows the accuracy of Logistic Regression, Classification Tree, and Support Vector Machines. SVM performed best.

B. Feature Selection

Next, with the intention of arriving at a more useful description of an image than raw pixel data, a number of feature extraction algorithms were attempted. Figure III show the performance of Blob features, Histogram of Oriented Gradients (HOG), and Principal Component Analysis (PCA).

TABLE I: MULTI-CLASS CLASSIFIER ACCURACY ON RAW DATA

Object Quantity	Logistic Regression	Decision Tree	SVM
0	0.33	0.12	0.33
1	0.19	0.14	0.12
2	0.24	0.15	0.29
3	0.25	0.28	0.26
4	0.24	0.21	0.29
5	0.20	0.21	0.24
overall	0.24	0.19	0.26

[a]Decision Trees suffer severe over-fitting

Given the large number of columns in a raw RGB image (total of 150,528), compressing work-load it to as few as 1000 columns, it is important to identify most effective solver. The following Principal Component Analysis solvers were tried: 'auto', 'full', 'arpack', 'randomized'. The PCA solver resulting in the highest accuracy: 'randomized'.

With respect to Histogram of Oriented Gradients (HOG), evaluation suggests that the images in the data set do not have enough dominant gradients. Thus, identifying products in a bin is difficult. In part, this may be due to Amazon's usage of tape to cover products in a bin. For many images, the tape occludes the products, causing a significant information loss.

TABLE II: ACCURACY OF FEATURE EXTRACTION METHODS, FOR SUPPORT VECTOR MACHINE CLASSIFIERS

Object Quantity	Blob	HOG	PCA
0	0.04	0.33	0.71
1	0.17	0.12	0.24
2	0.22	0.29	0.32
3	0.53	0.26	0.48
4	0.22	0.29	0.28
5	0.02	0.24	0.12
overall	0.26	0.26	0.32

[a]Blob and HOG under-performed

C. Support Vector Machines

The following SVMs were evaluated in Python, performing the corresponding parameter search:

1) svm.LinearSVC: with a linear kernel. Both 'l1' and 'l2' regularization were pursued. Additionally, 'hinge' and 'squared_hinge' loss functions. And 'ovr' and 'crammer_singer' multi-class strategies. As well as C penalty in the range 1 to 1e4
2) svm.NuSVC: Similar to SVC but uses a parameter to control the number of support vectors. Nu ranges 1e-6 to 1e-2 were tested.
3) svm.SVC: C-support vector classification. Tested penalty range from 1e-9 to 1e12 for C, and from 1e-11 to 1e-10 for gamma

Ten-fold cross-validation was used to find the best parameters against the validation set. Thus over-fitting and bias were traded off to achieve the greatest overall accuracy, while achieving greater than zero accuracy for every class. The

highest accuracy was achieved by svm.SVC, with the following parameters: C=1e2, gamma=1e-7. The corresponding confusion matrix is in Figure 6.

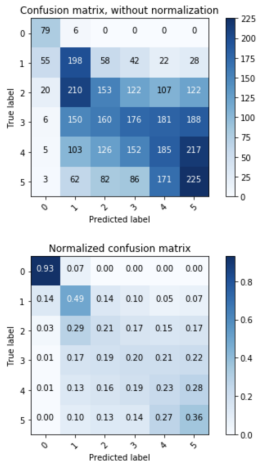


Fig. 6: SVM SVC Confusion Matrix

D. Convolutional Neural Networks via Transfer Learning

ResNet18, ResNet34 and ResNet50 ConvNets that have been pretrained on ImageNet have been considered for all our deep learning experiments.

a) ResNet18 - SGD Learning Rate Step Decay: We initiated learning process with a batch size of 128 and trained the full network of ResNet18 architecture using SGD with weight decay of 1e-4 and a learning rate of 0.1, set to decay by a factor of 10 every 10 epochs. We noticed that the model starts to over fit after 11 epochs. Model achieved an accuracy of 49.38% and Root Mean Squared Error (RMSE) 0.9905 on the validation set. However, training accuracy increased to over 94% after 24 epochs indicating that the model is failing to generalize, hence regularization was increased by setting weight decay to 1e-3 for the next iteration. After training the model for 20 epochs, model achieved an accuracy of 50.4% and RMSE of 0.9810 on the validation set. This indicates that regularization has prevented over fitting by constraining the parameter weights. Upon experimenting by increasing regularization further and decreasing learning rate, the accuracy on the validation set did not improve.

Fig 7 and Fig 8 depict training loss and Training / Validation Accuracy respectively for the best performing ResNet18 Model. Upon diving into per class accuracy numbers for ResNet18, we noticed that model is performing poorly on images with 4 and 5 items. Our hypothesis was that images with larger number of items needed more complex features to predict well. In order to learn more complex features in considerable time, we tried a deeper ResNet34 architecture.

b) ResNet34 - SGD Learning Rate Step Decay: Using the same learning process and initial set of learning rates and regularization settings from ResNet18, we trained ResNet34 models with step annealing. Model iterations were performed by examining training, validation loss and accuracy. It was

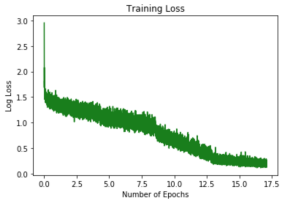


Fig. 7: ResNet18 Training Loss

Fig. 8: ResNet18 Train/Val Accuracy

surprising to notice that the accuracy of the best performing ResNet34 model, obtained with learning rate 0.01 and weight decay 1e-2, improved the accuracy by just 1.3% to 51.78% and RMSE decreased to 0.9647. Further, no improvement was observed in accuracy of the images with 4, 5 objects. Upon analyzing the images that the model failed to correctly predict, it was observed that some incorrect classifications were due to large amount of noise and occlusion in images (it was difficult even for a human to determine the right count). However, some were due to model not being able to learn key features pertaining to shapes associated with a combination of certain object categories in the bin. This motivated the experiments of using SGDR to improve the performance of the network based on research by Jeremy Howard [9]. Fig 9 and Fig 10 depict Training Loss and Train/Val accuracy for best performing ResNet34 model.

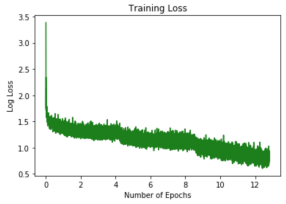


Fig. 9: ResNet34 Training Loss

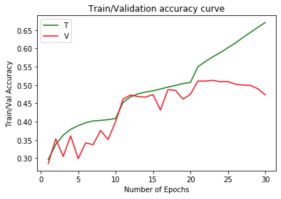


Fig. 10: ResNet34 Train/Val Accuracy

c) ResNet34 - SGDR: We set the batch size to 128 and transformed the images to 64x64 with horizontal flip data augmentation for training. Learning rate finder (Introduced in Methods Section) was used to set the initial learning rate (lr) to 0.008. Trained the network by freezing all the layers except for the final fully connected layer for 3 epochs using SGDR with cosine annealing. Weights on the Fully connected layer are very specific to the problem at hand so weight initialization for the last layer will be done somewhat randomly as the Amazon Bin Images are very different than ImageNet. Training last layer will reduce randomness in weights and can be performed quickly as the other layers are frozen. Underlying theory is described in the paper osinski,et al [8]. Now the training is performed on the entire network by using SGDR with cosine annealing with differential learning rates for 3 epochs. Differential learning rates set a lower learning rate of lr/9 to first few layers of the network followed by lr/3 for the middle layers and lr for the fully connected layer. The premise is that the first few layers are generic to all images and the last layers are more specific to the problem. Since ResNet was pretrained on ImageNet we wouldn't expect the weights of the earlier network to change much. Refer to osinski,et al [8] for more details. The process above was repeated by setting the image size to 128x128 and finally changing the size to 224x224 and training the full network until the model starts to over fit. Re sizing the images is useful to reduce over fitting since the same features are extracted from images with varying dimensions of features that prevent over fitting on the training data. Training accuracy after the final set of epochs on the 224x224 images resulted in an accuracy of 52.78% on the training set. Using test time augmentation, i.e. applying data augmentation on test image and letting model predict for all the augmented images and taking average of predictions, the accuracy improved by 100 basis points to 53.8%

We have seen from the training and validation loss on the last few epochs that training on different sized images is indeed preventing over fitting.

Upon inspecting some images that were incorrectly classified, it was observed that primary culprit was heavy noise and occlusions in images, especially images with fewer than 3 items that were misclassified. Given over 450,000 object categories, we expect training and test set to have different object categories which could affect the learning of the model.

d) ResNet34, ResNet50 with Adam Optimizer: Trained a ResNet34 model using Adam optimizer initially with a learning rate of 0.001 and 0 weight decay. Experimented with regularization and learning rate, the best performing model provided an accuracy of 50.6. SGD performs better although Adam converges much faster than SGD taking considerably lesser time for each gradient descent update. So far, all the experiments involved working with 150K images split into 70% train, 20% val and 10% test. Given, the computational efficiency of Adam, we were able to execute a ResNet34 Model on the entire dataset of 324K images with 80% train, 10% val and 10% test with a learning rate of 0.001 and 0 weight decay. We were able to achieve an accuracy of 56.2% on the validation set (Fig 11 depicts the confusion matrix) Repeating the experiment with ResNet50, a deeper architecture, did not improve the results. Table III compares the performance of various deep learning algorithms.

Although ResNet34 with Adam optimizer performed poorly when compared to ResNet34 with SGD on training with 150K images, upon increasing the dataset size to 324K, accuracy shot up by 6% indicating that training on the larger set of images by following the approaches discussed in this paper may provide better results.

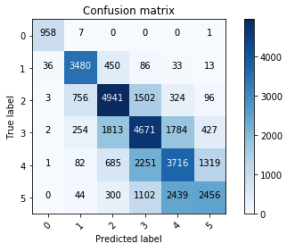


Fig. 11: ResNet34 Adam Confusion Matrix

TABLE III: ACCURACY OF CNN MODELS

Object Quantity	Training Accuracy	Val Accuracy	Val RMSE
ResNet18 SGD	55.9	50.4	0.98
ResNet34 SGD	55.2	51.2	0.99
ResNet34 SGDR	57.8	53.8	0.94
ResNet34 Adam	50.6	51.8	0.99
ResNet34 Adam All	62.3	56.2	0.90
ResNet50 Adam All	61.2	55.2	0.91

[a]All - Refers to training on 324K Images instead of 150K

V. Conclusions

Convolutional Neural Networks model outperforms SVM by 75% and achieves 3x performance over random. Our best performing model was able to gain 70 basis points improvement over the existing work [2] that was performed on a similar dataset. Overall accuracy on the test set is over 56% and overall RMSE is 0.90. Upon analyzing images that were incorrectly classified, it was observed that some images have such large noise and occlusions that it is not possible even for a human to count the number of items, resulting in confounding the training data set.

VI. Future Work

First, manually cleaning the data to remove such images would enhance learning. Second, with the Adam optimizer, we have seen that performance increases with a larger dataset. We are intrigued by the possibility that photos be taken by from different angles. And that metadata connect the content of a bin over time, so belief state may be tracked. Third, with over 450K product skews, images are bound to violate our assumption that they're drawn from a single distribution. Thus, forming ensemble models with different architectures and learning approaches may achieve higher accuracy.

The project's repository is: https://github.com/OneNow/AI-Inventory-Reconciliation

VII. Contributions

Pablo's primary contribution was on Support Vector Machines, Sravan's on Convolutional Neural Networks, and Nutchapols across the board.

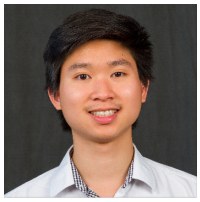
Acknowledgment

The authors are grateful to Professor Andrew Ng for his masterly transmission of machine learning to us.

Authors

Pablo Rodriguez Bertorello leads Next Generation data engineering at Cadreon, a maketing technology platform company. Previously he was CTO of Airfox, which completed a successful Initial Coin Offering. He is the co-inventor of cloud platform company acquired by Oracle. And the original designer of the data bus for Intel's Itanium processor. Pablo has been issued over a dozen patents.

Sravan Sripada works at Amazon. He is interested in applying artificial intelligence techniques to solve problems in retail, cloud computing and voice controlled devices.



Nutchapol Dendumrongsup is a Master's student at the Institute for Computational and Mathematical Engineering and Deapartment of Energy Resources Engineering at Stanford. He is interested in the application of machine learning in the energy industry and the traditional reservoir simulation in oil and gas industry.

References

[1] Joe Flasher, *Amazon Bin Image Dataset*, OpenData.
[2] Eunbyung Park, *silverbottlep/abid_challenge*, GitHub, Jul 20, 2017.
[3] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, *I*CML Deep Learning Workshop, 2015
[4] Zhang, et al. Deep Residual Learning for Image Recognition. [Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society, 10 Dec. 2015, arxiv.org/abs/1512.03385
[5] Loshchilov, et al. SGDR: Stochastic Gradient Descent with Warm Restarts. [Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society, 3 May 2017, arxiv.org/abs/1608.03983.
[6] Kingma, et al. Adam: A Method for Stochastic Optimization. [Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society, 30 Jan. 2017, arxiv.org/abs/1412.6980v9.
[7] Smith, and Leslie N. Cyclical Learning Rates for Training Neural Networks. [Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society, 4 Apr. 2017, arxiv.org/abs/1506.01186.
[8] Yosinski, et al. How Transferable Are Features in Deep Neural Networks? [Astro-Ph/0005112] A Determination of the Hubble Constant from Cepheid Distances and a Model of the Local Peculiar Velocity Field, American Physical Society, 6 Nov. 2014, arxiv.org/abs/1411.1792.
[9] https://www.fast.ai/