ASW Machine Learning Engineer Nanodegree – Capstone Project Proposal
Inventory Monitoring at Distribution Centers

Davide Martizzi

Domain Background

The study and engineering of solutions for inventory logistic and management is a mature and thriving field [1]. Cutting-edge companies are continuously adopting innovative solutions to automate and increase the efficiency of processes such as manufacturing, quality assurance, inventory management, product distribution and marketing. The adoption of new AI-driven solutions has significantly improved this transformation, but several challenges still exist, because the performance of some of these solutions is still falls short of expectations [2]. The goal of this project is to solve a business problem related to inventory monitoring at distribution centers.

Product distributions centers often use robots to move objects as a part of their operations. In order to obtain full automation of the process and assess whether robots are operating as intended, monitoring systems are deployed to perform quality control. One of the most basic applications for quality control is counting the number of objects in each bin without human involvement, e.g., by processing of images of each bin.

In this project, I will develop a deep-learning-based computer vision model to count the number of objects in bins. The model will be deployed on AWS and its predictions will be accessible via a web app.

Problem Statement

Formally, the problem falls in the realm of weakly-supervised object detection, the main task being the identification and counting of objects in bins within a distribution center using images. The task is weakly-supervised because it's difficult to get detailed annotations for the bounding-box position and label of objects in the images, which are typically required to train most object detection models.

However, since we are only interested in the count of objects in each bin, some of the complexity of identifying the correct location and label for each object can be dropped. This should allow me to choose simpler model architectures and use weaker labels to perform the task.

The model will take images as an input and yield the count of objects in the image as an output.

Datasets and Inputs

The dataset that will be used for this project is the Amazon Bin Image Dataset: https://github.com/awslabs/open-data-docs/tree/main/docs/aft-vbi-pds [3]. The dataset contains > 500,000 bin JPEG images and corresponding JSON metadata files describing items in the bin. Among the metadata is the count of the number of objects in each bin, which can

be used to define target labels for the machine learning problem. The dataset will be split in training, validation, and test subsets, which will be stored on S3.

Solution Statement

The model used to solve this problem is a variant of the approach implemented in https://github.com/silverbottlep/abid_challenge [4]. The idea is to reduce the object count problem to an image classification model in which the class corresponds to the count of objects in the image. In my implementation, I plan to load the weights of the pre-trained ResNet50 classifier for ImageNet, and only train the deepest layers of the network plus the custom output layers. With this approach, I should be able leverage the low-level image features extracted from image classification and re-purpose the rest of the network for object identification and counting. The output layer will yield a softmax distribution over the possible values of a class label corresponding to the number of objects in the image.
Figure 1 shows a diagram of the model.

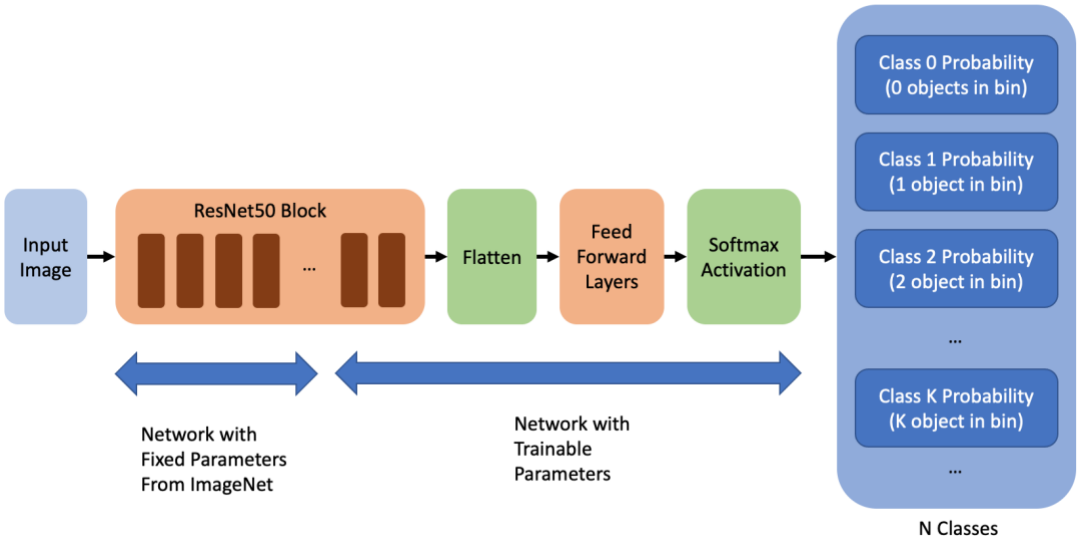

Figure 1. Model Architecture.

The solution will be deployed in a web app where a user would be able to select the ID of one of the images in the test set. The client will only maintain a list of the IDs of the images in the test set. The client will send a json request to a lambda function which will retrieve the image stored on S3 and pass it to a model endpoint. The model will yield the inferred number of objects in the image, which will then be sent by the lambda function back to the client together with the image. The client will then visualize the image and the inferred number of objects. Details of the implementation are given in the Project design section.

Benchmark Model

The benchmark model chosen for this project will be based on pre-trained ResNet models with custom output layers. This approach is inspired by the following implementation of a solution to the same problem: https://github.com/silverbottlep/abid_challenge [4]. This implementation achieves an accuracy of 55.67% and a RMSE of 0.93.

In the benchmark implementation, the model is trained from scratch and uses ResNet34, whereas in my implementation I plan to load the weights of the pre-trained ResNet50 classifier for ImageNet, and only train the deepest layers of the network plus the custom output layers.

Evaluation Metrics

The evaluation metrics chosen for this task will be:

1. The accuracy in retrieving the exact number of objects in the image.
2. The RMSE obtained by comparing the inferred number of objects with the true one.

The metrics will be tracked for training and validation sets during hyperparameter optimization sweeps. The same metrics will then be used to evaluate the best model but computed on the test set.

Project Design

The design of the project, timeline & workflow, general architecture of the system, and visualization of the components that need to be developed is summarized in Figure 2. The architecture of the system is very simple, and the main advantage of adopting this approach lies in the use of transfer learning to speed up the model training process while achieving competitive performance. Although simple, this architecture is relatively scalable, because it allows to take advantage of both lambda function concurrency settings and endpoint autoscaling.

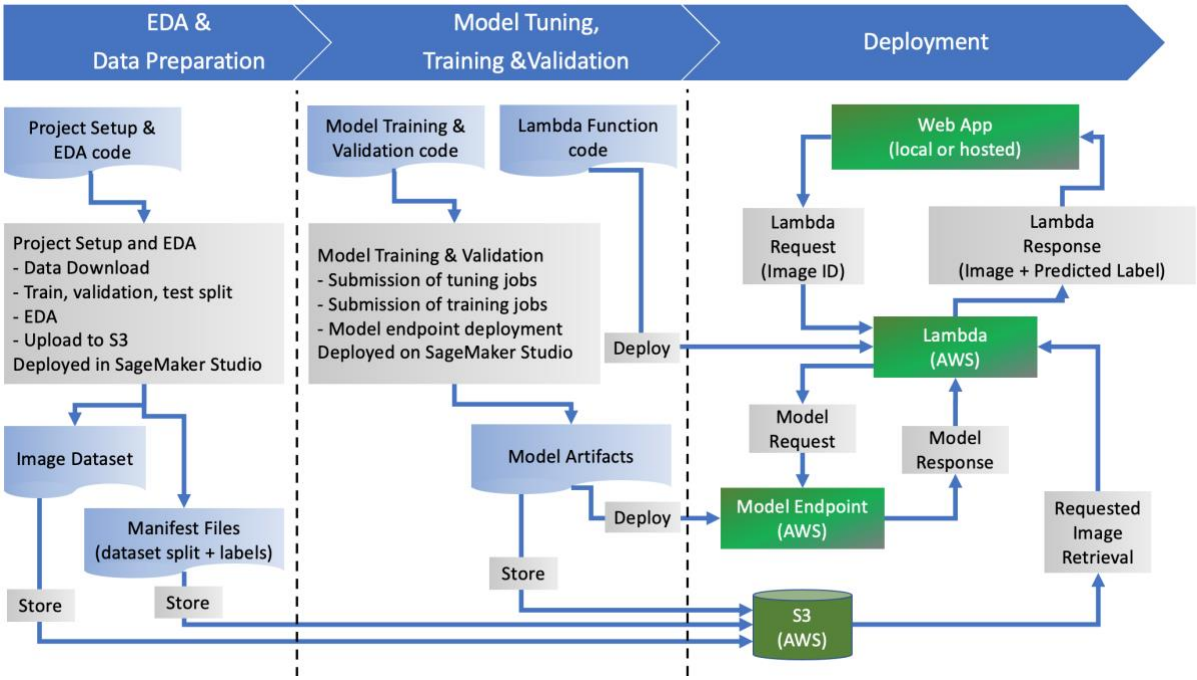

Figure 2. Project Phases and System Architecture Diagram.

Phase 1 – EDA and data preparation

The first phase of the project will involve development of the code to perform data exploration and preprocessing. In this phase:

- Dataset will be split in training, validation, and test sets.

- Data split will be documented in json format or similar.
- Data will be loaded to S3.
- EDA will be performed to better understand the dataset and design pre-processing steps.

Phase 2 – Model definition, tuning, training and validation

The second phase of will involve development of the code to define the model, tune it, train it, and validate it. In this phase:

- Training and hyperparameter tuning scripts will be developed to train the model.
- Hyperparameter tuning scripts will be deployed on AWS SageMaker.
- Best model will be selected, and performance on test set will be recorded.

Phase 3 – Deployment

The final phase of the project will involve development of the code necessary to deploy the model and query it from a web app. In this phase the following components will be deployed:

- Server side: scripts to deploy a model endpoint will be developed.
- Server side: model endpoint will be deployed.
- Server side: script defining Lambda function that is triggered by API calls and sends inputs to the model endpoint will be deployed.
- Server side: Lambda function will be deployed.
- Client side: web app that sends API requests to the Lambda function, collects the results, and visualizes them will be created.

References

[1] Williams, Brent & Tokar, Travis. (2008). A Review of Inventory Management Research in Major Logistics Journals. The International Journal of Logistics Management. 19. 212-232. 10.1108/09574090810895960.

[2] Baldassari. (2021). Industry 4.0 reality check: Separating hype from reality. https://flex.com/resources/industry-4-reality-check-separating-hype-from-reality.

[3] Amazon Bin Image Dataset. https://github.com/awslabs/open-data-docs/tree/main/docs/aft-vbi-pds.

[4] Benchmark model. https://github.com/silverbottlep/abid_challenge.