Capstone Project

Background

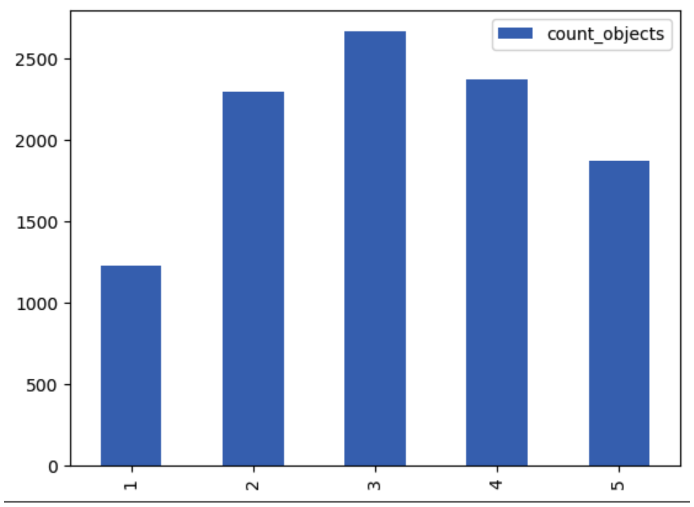
For this Udacity Capstone project the Amazon Bin Image Dataset was used to classify the number of objects in Amazon warehouse bins. Amazon is an internet based enterprise that focuses on providing e-commerce, cloud and digital services. In order to fulfill the orders placed on their e-commerce site they often store the products a customer has purchased in warehouse facilities. In these facilities the goods are stored in bins. Currently Amazon's warehouses are overseen by fulfillment workers. Initially, pickers were used to update a bin's capacity, but since 2012 robots have been added to improve the workload. To meet the 2 day delivery standards Amazon has in place for some of their customers they need to ensure they have the most efficient layout possible.

Problem Statement

Currently, Amazon associates work in conjunction with Kiva robots to ensure the warehouse floor is in order. One issue with the bin storage layout is that employees currently have to estimate or manually keep track of the number of items in a bin if they want to be aware of its capacity. This can consume extra time when an associate needs to manually inspect and log goods in a bin. Given how Amazon tries to automate most of its processes this leaves room for an area of improvement. The time it takes employees to manually track the bin's capacity can be used better elsewhere. This capstone project serves to provide a solution to this issue.

Datasets and Inputs

In order to work on this project data will be collected from the [1]Amazon Bin Image Dataset. The Bin Image dataset contains over 500,000 images of bins from an Amazon Fulfillment Center. Although the entire dataset will not be used, part of it will be sampled to test, train and validate the model. The dataset consists of images of bins with 1-5 objects per image. The distribution of the number of objects in the various images can be seen below:



Solution Statement

This project will serve to remedy the aforementioned issue by predicting the number of items in a bin. Given an image of the Amazon storage bin, this project will use a convolutional neural network to classify the number of objects in each bin. A pre-trained Residual Network (RESNET) model will be used to perform this image classification task. The RESNET model was selected as RESNET models often work well at accurately identifying objects in images. Given the large amount of training data, as well as the additional layers the RESNET model will provide make this ideal for this analysis. This algorithm will be built using Amazon Sagemaker. Using Sagemaker will allow the model to be built on the AWS Cloud for ease of training data storage in S3 as well as the ability to deploy the endpoint to make future predictions. The platform will allow users to scale to meet the demands of the model.

Benchmark Model

The benchmark model that will be used for this analysis is a convolutional neural network. Various types of Convolutional Neural Networks are often used for object counting. As authors Wang, Xiao, Guo and Zhang note,[2] “the accuracy of object detection has been dramatically improved thanks to the advance of deep convolutional neural networks”. Given others tend to have the most success when counting objects with a convolutional neural network, that will be used for this analysis as well.

Evaluation Metrics

The primary metric that will be used to evaluate this model will be accuracy. Other metrics that may be considered include MAE and RMSE. The accuracy will be used to ensure the neural network is giving correct predictions on the number of objects in a bin. A higher accuracy will suggest that the model is working well.

Project Design

The process to complete this project will include understanding the problem/business objective, then working to process and analyze the data. Once the data is processed and in a state fit to build a model on, the model building process will commence. Upon model build completion it will be tested on a portion of the dataset mentioned prior. Depending on the accuracy the model may be tuned to fit the best hyperparameters. Once the performance is satisfactory the model will be evaluated using the evaluation metrics above.

Analysis

For this image classification algorithm the dataset was explored to understand the number of classes as well as the best model that should be used. Upon initial inspection of the dataset it was found that there were images containing 1-5 products each. This was used to determine that the final model should have 5 classes.

Due to the budget and computation constraints associated with this nanodegree the number of images used to train the model was downsized from the overall number of images available. Overall there were 500,000 images that could be used. Only 2% ended up being sampled to perform the model development process. Of that 2%, 60% of images were used to train the model, 20% were used to test the model and 20% were used to validate the model.

Given the nature of the image classification algorithm minimal preprocessing steps were required. The raw images were pulled from the Amazon Bin Dataset and transformed via data loaders before running the algorithm. The transformations resized each image to be 224 by 224 before running in the model.

After the data was allocated to the proper split the hyperparameters were selected. Due to the lack of remaining project budget many of the hyperparameters chosen were fairly conservative. The batch size chosen was 32, the number of epochs was 4 and the learning rate was .01. If cost wasn't a concern, instead of determining these hyperparameters up-front, hyperparameter tuning could have been used to find those that would lead to the best performance.

As mentioned before, this model was built using a RESNET model. The objective across the hyperparameters was to maximize accuracy. The table below shows the results that came from training the algorithm. Accuracy and loss improved with each new epoch.

Training set:

Metric	Epoch 0	Epoch 1	Epoch 2
Accuracy	24.40%	29.48%	32.40%
Loss	1.56	1.35	1.32

The final results from the test set were accuracy - 31% and average loss - 1.50. When looking at these evaluation metrics to understand the performance of this model it did quite poorly. Typically one would strive to have an accuracy score of at least 70% or above. The next section will talk through issues encountered that may have led to this poor performance as well as ways this score could've been improved.

Potential Model Improvements

Hyperparameter tuning could've vastly improved the performance of the model. As seen from the training set table with each iteration the accuracy was improving. Increasing the number of epochs would've allowed the model more time to train and ideally produce a better score.

Additionally, given the size of images available the model could've been trained on a larger dataset. Of the 500,000 images available only 2% were used. The smaller portion of images was used to ensure the computational power behind the model didn't exceed the amount budgeted for the course. In a production setting the model would be trained on a much larger sample of the data.

Lastly, given more time I would’ve looked at more ways to optimize cost. When initially running this model the training failed due to memory issues. A larger instance was used within Udacity’s allowed choices, however an instance with a GPU/TPU may have been more efficient. Additionally, spot instances could’ve been used to lower costs although the training may not have been as stable.

During a future build of this model I would spend more time finding the best hyperparameters between learning rate, epochs and batch size. I’d look at other types of RESNET algorithms and CNN algorithms in general to see if they might lead to a more accurate outcome. I’d also train on at least 100,000 of the 500,000 images I have available.

Conclusion

Overall I would not recommend this model for deployment in production. While the type of model (RESNET) may have been correct, more iterations would need to be done to achieve the target accuracy score above 70%.

This model is not accurate enough to be a good solution to the problem statement. Compared to the ideal model accuracy benchmark score (70%+) this model falls short. After performing the recommendations above, this model may then be a potential solution to the problem statement above.