Machine Learning Engineer Nanodegree

AWS Capstone Project

Ledezma, Lucia del Valle

April, 2023

Table of Contents

1. Definition 3
    1.1 Project Overview 3
    1.2 Problem statement: 3
    1.3 Metrics: 4
2. Analysis 4
    2.1 Data Exploration 4
    2.2 Exploratory Visualization 6
    2.3 Algorithms and Techniques 8
    2.4 Benchmark Model 9
3. Methodology 10
    3.1 Data Preprocessing 10
    3.2 Implementation 10
    3.3 Refinement 11
4. Results 11
    4.1 Model Evaluation and Validation 11
    4.2 Justification 12
Conclusion 12
References 13

1. Definition

1.1 Project Overview

Computer vision is a field of artificial intelligence that enables computers to derive meaningful information from digital images, videos and any other visual inputs [1]. For this project the subfield related to is Image Classification, it is the task of identifying images and categorizing them in one of several predefined distinct classes [2]. A convolutional neural network is a method of deep learning that takes an input image and assigns importance (learnable biases and weights) to various objects in the image, distinguishing one from the other [3]; furthermore, it has demonstrated exceptional image classification, recognition, segmentation, and retrieval [4].

This project is focused in the computer vision area, we are going to work with images and classify them.

1.2 Problem statement:

In a distribution center, all the objects are managed by robots, these robots are responsible to take every object into bins. How can we count how many objects there are in each bin?. The objects can be countered by a system in order to get an inventory. This involves knowing how to classify the images into object's amount labels by using Computer Vision techniques.

The images look like:

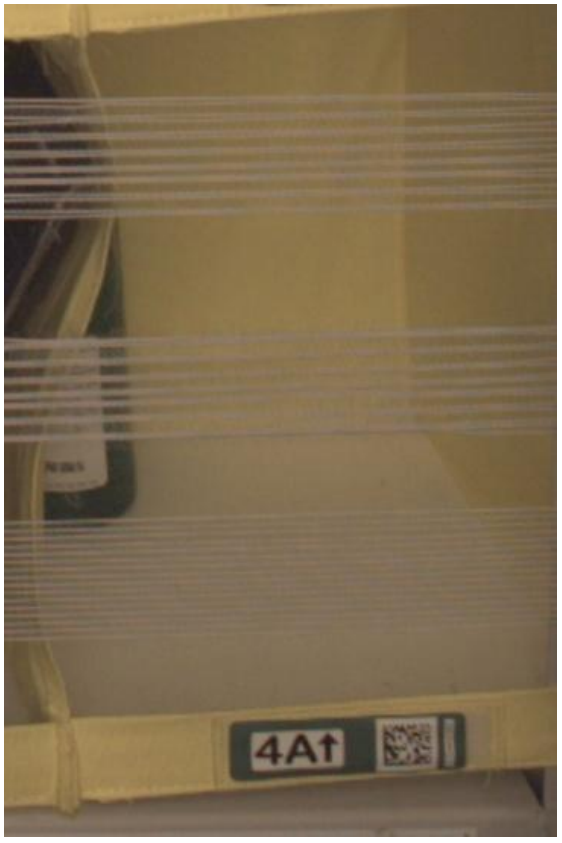

Fig 1. Image of a bin with only one object

1.3 Metrics:

Accuracy is one of the most important evaluation metrics, this represents the percentage of correct predictions of the model:

$Accuracy = (Correct\ Predictions)/(total\ predictions)$

By having the confusion matrix, it could being explained as:

$Accuracy = (TP + TN)/(TP + TN + FP + FN)$

This is the metric we need in this problem, because it involves classifying images between 5 classes.

2. Analysis

2.1 Data Exploration

Amazon Bin Images is the dataset for this project, this dataset contains over 500,000 images and metadata from bins of a pod in an operating Amazon Fulfillment Center. The bin images in this dataset are captured as robot units carry pods as part of normal Amazon Fulfillment Center operations.

Because this dataset is too large, approx. 84 GB; I only used a subset of this. I downloaded it from s3://aft-vbi-pds/.

Data was provided into 5 folders, each folder is a label class (number of objects in a bin), in other words, each image is classified into the folder according to the number of objects it contains.

To detail the dataset, is necessary to understand their distribution:

` 1. For the training dataset, 70% of the original data were used.
2. For the test set, 20% of the original data
3. Finally, for the validation set, only 10% of the original data.

Table 1. Original Dataset

Original Dataset	
Labels	Number of objects
1	1228
2	2299

3	2666
4	2373
5	1875

As we can see, We have different amounts of images in each class.

Table 2. Train Dataset

Training Dataset	
Labels	Number of objects
1	863
2	1613
3	1869
4	1664
5	1315

Table 3. Test Dataset

Test Dataset	
Labels	Number of objects
1	244
2	458
3	532
4	473
5	374

Table 4. Validation Dataset

Validation Dataset	
Labels	Number of objects
1	121
2	228

3	265
4	236
5	186

2.2 Exploratory Visualization

In the images below, we can see three plots showing the labels frequencies in each dataset, train, test and validation set. In other words, we can observe the amount of images in each class of the mentioned datasets.

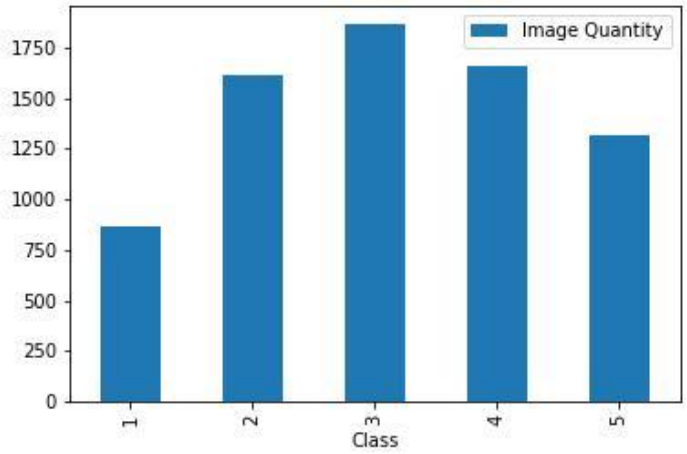

fig. 2 Train dataset bar plot.

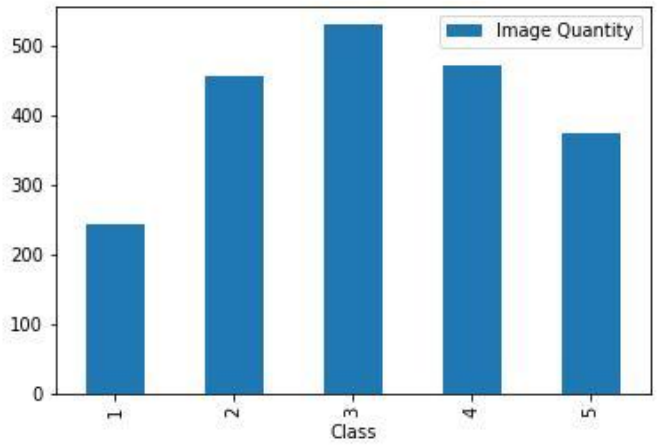

Fig 3. Test dataset bar plot.

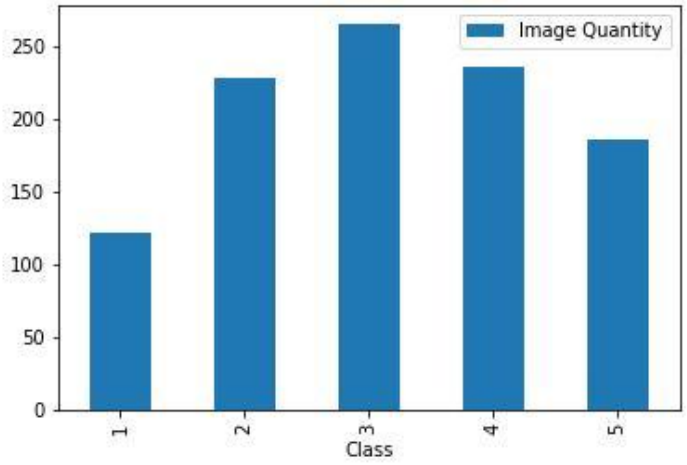

Fig 4. Validation dataset bar plot.

2.3 Algorithms and Techniques

Convolutional neural networks are distinguished from other neural networks by their superior performance with image, speech, or audio signal inputs. They have three main types of layers, which are:

Convolutional layer

Pooling layer

Fully-connected (FC) layer [5]

Deeper neural networks are more difficult to train. With Resnet, it becomes possible to surpass the difficulties of training very deep neural networks [6].

ResNet18

ResNet-18 is a convolutional neural network that is 18 layers deep. We can load a pre-trained version of the network trained on more than a million images from the ImageNet database. The pretrained network can classify images into 1000 object categories, such as keyboard, mouse, pencil, and many animals. As a result, the network has learned rich feature representations for a wide range of images. The network has an image input size of 224-by-224 [7].

All the steps below are executed by using Sagemaker environment, we can access all scripts in the github repository of this capstone project.

ResNet50

ResNet-50 is a convolutional neural network that is 50 layers deep. This model is an improvement of the previous one.

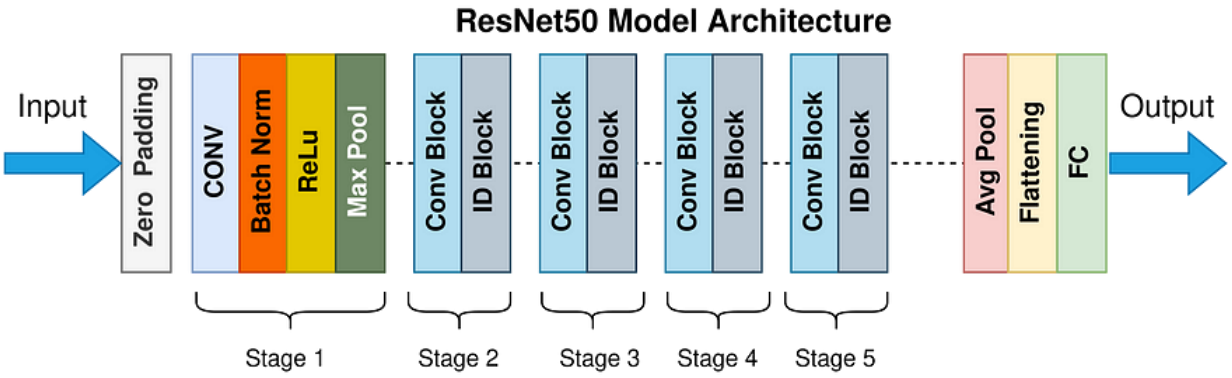

Fig 6. resnet50 architecture [8].

this is image classification problem, so with resnet50, I will have a good solution.

2.4 Benchmark Model

I used the resnet18 pre-trained model as a base for creating my benchmark model, I did transfer learning adding the output layer 5 classes according to the number of objects in each bin.

This is the architecture schema:

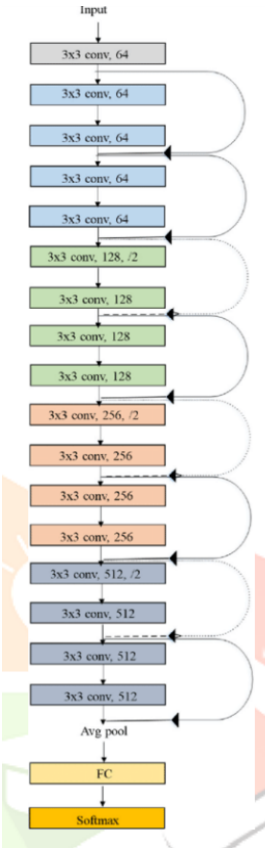

Fig 5. resnet18 [9]

In order to accomplish this objective, I trained the model by executing the same script of the proposal model, setting the model name parameter 'model-arch' with 'resnet18'. The training process was done with the same dataset explained before.

I trained the model by setting:

hyperparameters = {'batch-size': 64, 'lr': 0.009, 'model-arch': 'resnet50', 'model-n-classes': 5, 'epochs': 6}

the model got this performance:

Testing Accuracy: 29.072561268620856

Testing Loss: 1.827670568402486

3. Methodology

3.1 Data Preprocessing

When I downloaded the dataset from amazon bin, I only got a folder which contains subfolders according to each class.
It was necessary to split the data into the folders train, test and validation, so the model could be trained by using the ImageFolder method of Pytorch and create the data loaders.
To accomplish this objective two methods were developed : train_test_split which does the splitting,

```
def train_test_split(dataset_path, per_test, per_val):
    classes_directory = listdir(dataset_path)
    result_classes_split = {'train':{ class_dir: [] for class_dir in classes_directory},
                            'val':{ class_dir: [] for class_dir in classes_directory},
                           'test':{ class_dir: [] for class_dir in classes_directory}}
    if per_test + per_val >= 100:
        raise Exception( 'ERROR')

    for class_directory in classes_directory:
        elements_class =  glob.glob(f'{path.join(dataset_path, class_directory)}/*')
        n_elements_class = len(elements_class)
        n_sample_test  =  (n_elements_class * per_test) //100
        n_sample_val  =  (n_elements_class * per_val) //100
        result_classes_split['test'][class_directory] = elements_class[:n_sample_test -1 ]
        result_classes_split['val'][class_directory] = elements_class[n_sample_test - 1 : n_sample_test  + n_sample_val - 2]
        result_classes_split['train'][class_directory] = elements_class[n_sample_test  + n_sample_val - 2 : ]
    return result_classes_split
```

Fig 6. train_test_split function.

and write_sample_elements for saving the images into its respective folder.

```
def write_sample_elements(dir_name, data):
    makedirs(dir_name, exist_ok= True)
    classes = list(data.keys())
    for class_name in classes:
        dir_class = path.join(dir_name, class_name)
        makedirs(dir_class, exist_ok=True)
        for file_path in tqdm(data[class_name]):
            filename_output = path.join(path.join(dir_name, class_name), file_path.split('/')[-1])
            with open(file_path, 'rb') as img_input, open(filename_output, 'wb') as img_output:
                img_output.write(img_input.read())
```

Fig 7. write_sample_elements function.

Finally, the splitted dataset folders were uploaded to s3, so the model can train with it.

3.2 Implementation

I built the scripts train.py and train_hook.py, where are developed all the instruction in python 3.x for trained a pre-trained model, and save the model into s3.

The next method load resnet18 or resnet50 pretrained models and add the last layer, for the specific problem:

```
def net(n_classes: int = 5,  model_arch: str ='resnet50' ):
    '''
    Args:
    n_classes: number of neurons in the last layer,  it correspond to the number of classes
    model_arch: pretrained model to load.
    '''
    model = None
    if model_arch == 'resnet50':
        model = models.resnet50(pretrained=True)
    elif model_arch == 'resnet18':
        model = models.resnet18(pretrained=True)
    else:
        raise('Model name error!!!!')
    for param in model.parameters():
        param.requires_grad = False

    num_features=model.fc.in_features
    model.fc = nn.Sequential(
                nn.Linear(num_features,  n_classes))
    return model
```

Fig 7. net function.

So, I did transfer learning by adding the 5 classes needed to solve this problem.

First I have setted the hyperparameter with specific values

hyperparameters = {'batch-size': 64, 'lr': 0.009, 'model-arch': 'resnet50', 'model-n-classes': 5, 'epochs': 6}

and I got the performance of:

Testing Accuracy: 30.225852955309946, Testing Loss: 1.805760314062431

3.3 Refinement

I tried to improve more my model, so I did Hyperparameter tuning with this values:

```
hyperparameter_ranges = {
    "lr": ContinuousParameter(0.001, 0.1),
    "batch-size": CategoricalParameter([32, 64, 128]),
}
```

finally, I got this new values:

hyperparameters_tuned = { 'batch-size': 128, 'lr': 0.00254525353426351}

Then I trained the model once more with this new setting.

4. Results

4.1 Model Evaluation and Validation

With my proposal refined model I got this performance:

Testing Accuracy: 30.033637674195095,

Testing Loss: 1.5222831945588875

Also, I deployed an endpoint in sagemaker for making predictions:

```
predictor=estimator.deploy(initial_instance_count=1,
                           instance_type='ml.m5.large')
```

I selected four images from the test set labeled with:

tensor([4, 1, 2, 1])

then I invoked the endpoint with those images:

```
inference = predictor.predict(images)
```

and got:

array([2, 0, 2, 0])

4.2 Justification

As we can observe, The proposal model with resnet50 had a little better performance than the benchmark model (with resnet18). because *30.033637674195095 > 29.072561268620856.*

As I mentioned before, the benchmark model got:
Testing Accuracy: 29.072561268620856
Testing Loss: 1.827670568402486

so I got an improvement of:

tuned_model_acc - benchmark_accuracy = (30.033637674195095 - 29.072561268620856)

= 0.9610764055742393 points.

Conclusion

As we could see, the model predictions could be better, for this reason we can do more adjustments to the model. Some hyperparameters could be changed, such as the number of epochs, and we can try by increasing the max_jobs parameter in the HyperparameterTuner object, so it can try with more values (lr and batch-size). Furthermore we can train the model by increasing the computing power, for example by GPU instead of CPUs.

Finally, I realized the dataset was not uniform, because we have differing amounts of images in each class, So , maybe we can use the same number of objects in each class and got an improved model.

References

[1] What is computer vision?. [Online]. Available: https://www.ibm.com/topics/computer-vision

[2] What is image recognition and computer vision?. [Online]. Available: https://www.ibm.com/topics/computer-vision .

[3] Khan, A.; Sohail, A.; Zahoora, U.; Qureshi, A.S. A Survey of the Recent Architectures of Deep Convolutional Neural Networks. Artif. Intell. Rev. 2020, 53, 5455–5516.

[4] Liu, X. Recent progress in semantic image segmentation. Artifical Intell. Rev. 2019, 52, 1089–1106.

[5] How do convolutional neural networks work?. [Online]. Available: https://www.ibm.com/topics/convolutional-neural-networks

[6] Deep Residual Networks (ResNet, ResNet50) – 2023 Guide. [Online]. Available: https://viso.ai/deep-learning/resnet-residual-neural-network/

[7] resnet18. [Online]. Available: https://www.mathworks.com/help/deeplearning/ref/resnet18.html

[8] The annotated resnet50. [Online]. Available: https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758.

[9] Resnet18 Model With Sequential Layer For Computing Accuracy On Image Classification Dataset. International Journal of creative research thoughts (IJCRT). vol 10. May. 2022.