Training of Convolutional Neural Networks with Simulated Image Data for Logistics Object Detection

Team Members **Karri Aravind Reddy**
Jagriti Labh

Supervisor **Hagen Borstell(ILM)**

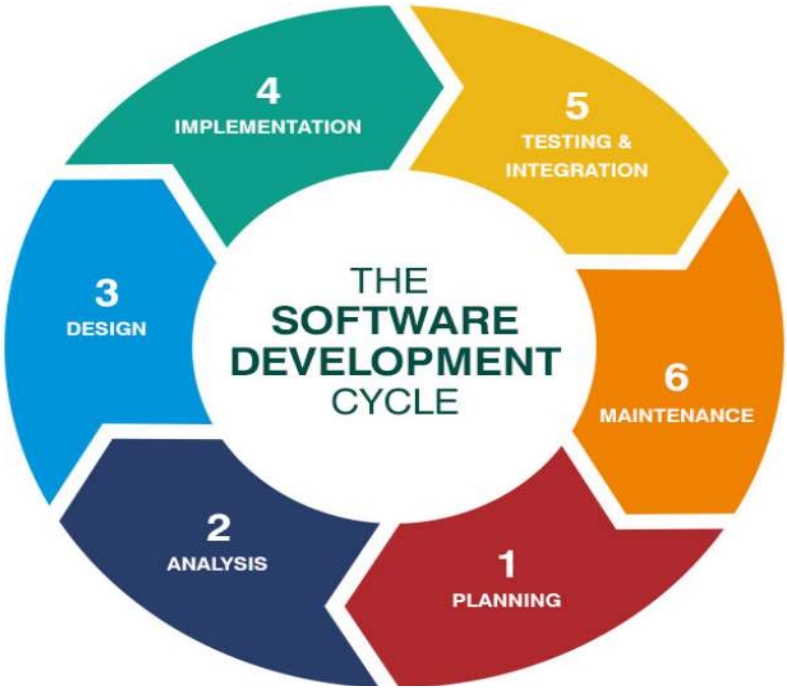
Table of Contents

1.Introduction

Localizing and Identifying the logistics objects is one of the prime concerns for the appropriate processing of the logistics operations. Due to the development of deep neural network image-based object recognition has come into scenario.
In our project we have used TensorFlow. Training and the test data had to be collected first as there was no databases and pre-trained networks which could be used directly for recognition of logistics objects. But this approach is time consuming, so Transfer learning was used as it has the capability to adapt the pre-trained network to an application even with a small amount of training data.
In this project training of the convolution neural network with simulated image data is done in Order to detect the logistics object.

2.Working Methodologies

We chose a Scrum based agile methodology to develop our application. Agile Methodology is a type of project management process. The agile method anticipates change and allows for much



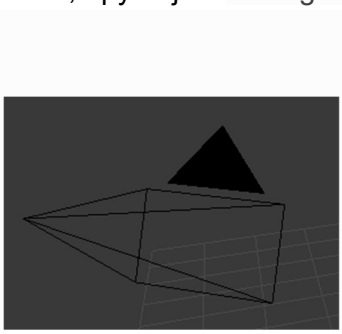
more flexibility than water flow methods. Clients can make small objective changes without huge amendments to the budget or schedule. Scrum is a lightweight agile project management

framework with broad applicability for managing and controlling iterative and incremental projects of all types.

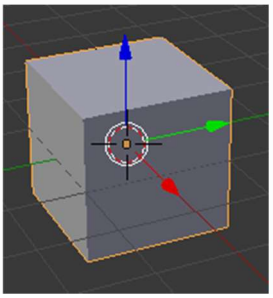
3.Overview of Tools

3.1. Blender

Blender is a 3D animated tool which is used to create still images, video, and real-time interactive video games. It is an open source software tool is basically written in Python. In our project, we have used blender to generate images. There are three elements in the blender - camera, object, light. Each element of the blender can be accessed as an Object. Example- bpy.camara, bpy.object The figure below shows the elements in the blender.



Camera



Object

Light

3.2. Keras

Keras is an open source neural network library written in Python, written in Python and capable of running on top of Tensorflow. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research. [1]

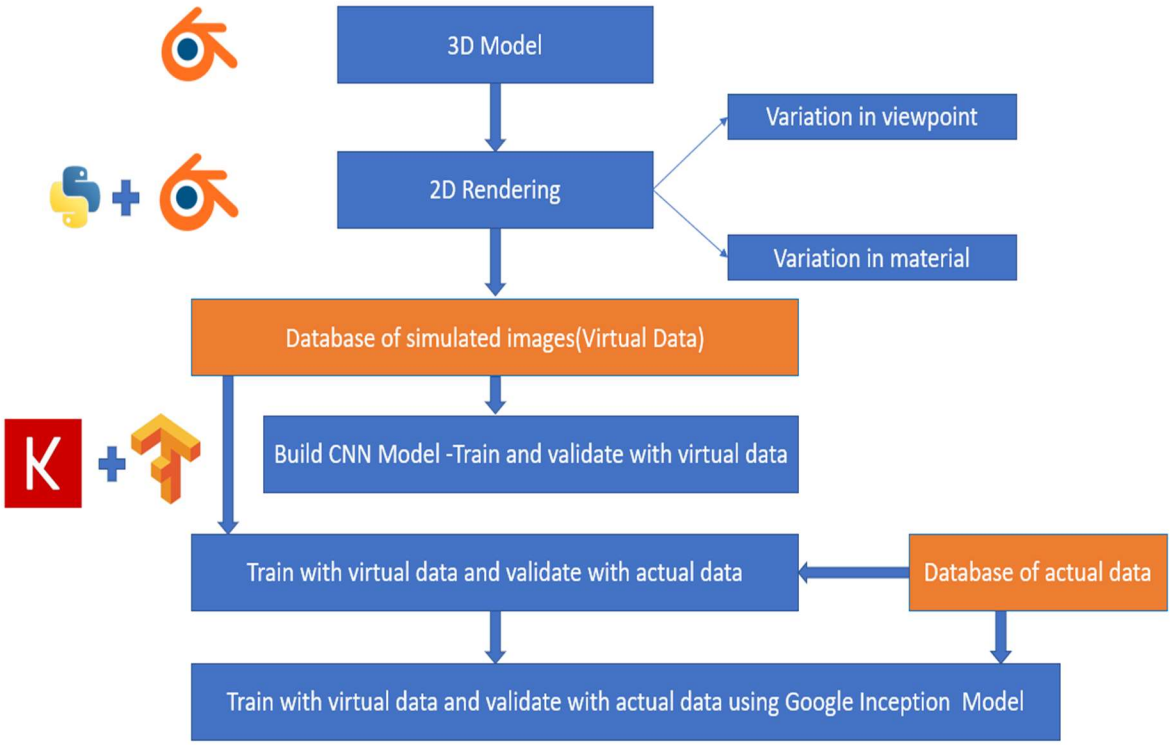
3.3. TensorFlow object detection API

Creating accurate machine learning models capable of localizing and identifying multiple objects in a single image remains a core challenge in computer vision. The TensorFlow Object Detection API is an open source framework built on top of TensorFlow that makes it easy to construct, train and deploy object detection models. [2]

4.Implementation

The CAD Modes are appended into the blender which is preloaded with our custom plugin. when we click the animate button in blender the camera starts to render images with the predefined settings. In the generate more data panel with the configuration kept to maximum i.e levels:15, Degrees: 1 and render resolution to full HD the blender generates around 5600 high-resolution images with all the variation in light.
In the initial stage of the project models, cube and sphere were taken into consideration and images were rendered then the rendered images were manually divided into a test set and training set. These images were then fed to the machine learning algorithm and the algorithm trains itself for that image and then when the images from the test sets are fed into the algorithm then based on the trained data set it calculates the accuracy of the test data

4.1 workflow

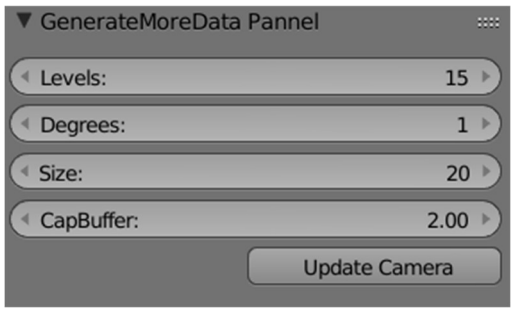


4.2 Rendering

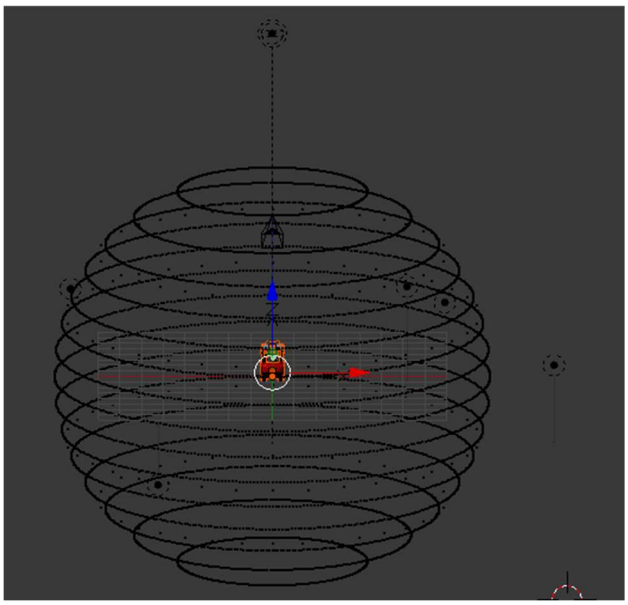
Rendering is the process in blender through which we are generating images through computer programs. Images are generated on the basis listed below.

- Variation in camera angles (pose).
- Variation in material color.
- Variation in Light sources and Number of light sources.

For variations of camera angle, we developed a tool in python which moves the camera with the variations on Levels, Degrees, Size, and Cap-buffer with the help of the UI panel. Levels indicate the number of height levels camera has to rotate around the model. Changes in degree will make changes in one of the camera levels to take photos of the model. Changes in size will make the changes in the dimensions of the model and the cap-buffer will set the buffer at the caps.



The following image shows the model with the rendering image process with the movement of the camera.



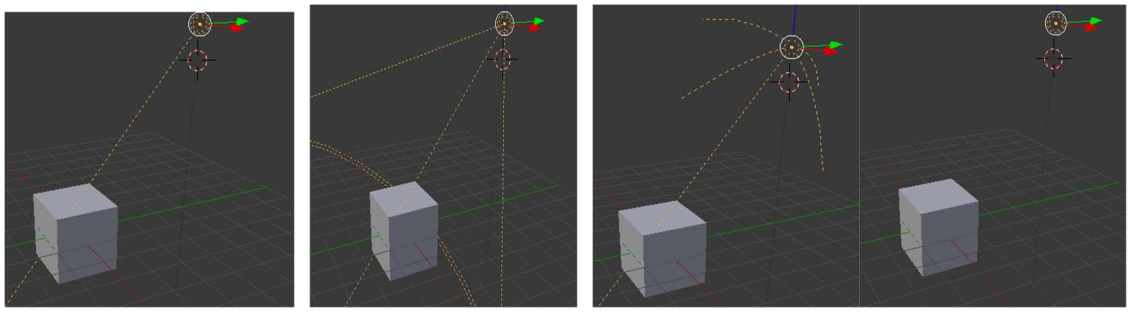
Every model is modified with some changes in the material to generate more data. Modification of each model is carefully done manually depending on the type of the model and some models are kept the same.

Table showing variations in the material of models for rendering

Model Name	Changes in model
Barrel	Original Model
Coil	Original Model Material 01
Forklift	Original Model material 01 material 02 material 03 material 04
Pallet	Original Model

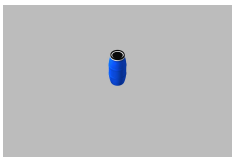
Variation in Light

Following images show the variations in light using blender

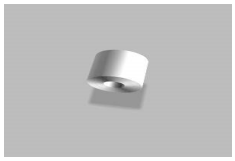


Area Spot Hemi Point

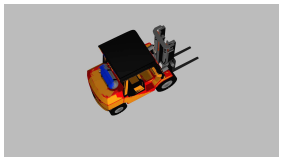
After rendering all the images from all the variations data is saved in folders which later used for training of our CNN model. Some of the sample images generated by blender by using our tool.



Barrel



coil



Forklift

Pallet

4.3 Overview of actual data

We found the data set of the actual images of logistic objects which is used for evaluation of the virtual data

A sample of actual images used for testing

Barrel

Coil

Forklift

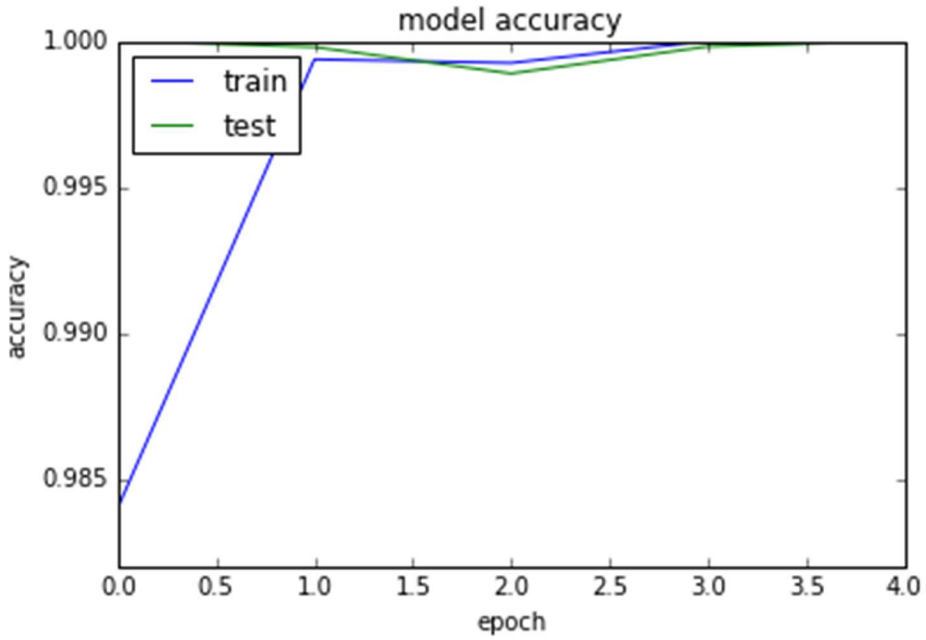
Pallet

4.4 Training and Initial Results.

In order to test our simulated data, we made a CNN model with two Convolution layers, flattening layer and a pooling layer.

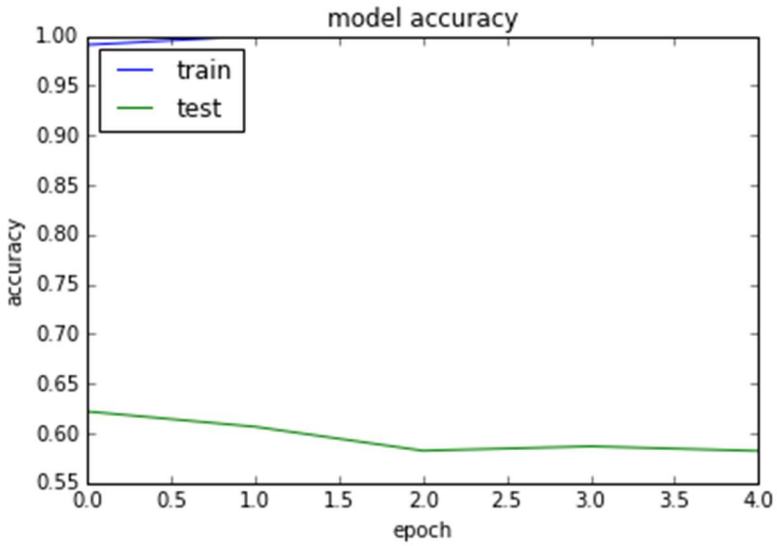
4.4.1 Training and testing with all simulated data

To test the CNN model and our simulated data we divided the simulated data randomly into test and training set and fitted them into our CNN model. Results were as expected our model could accurately predict the objects.

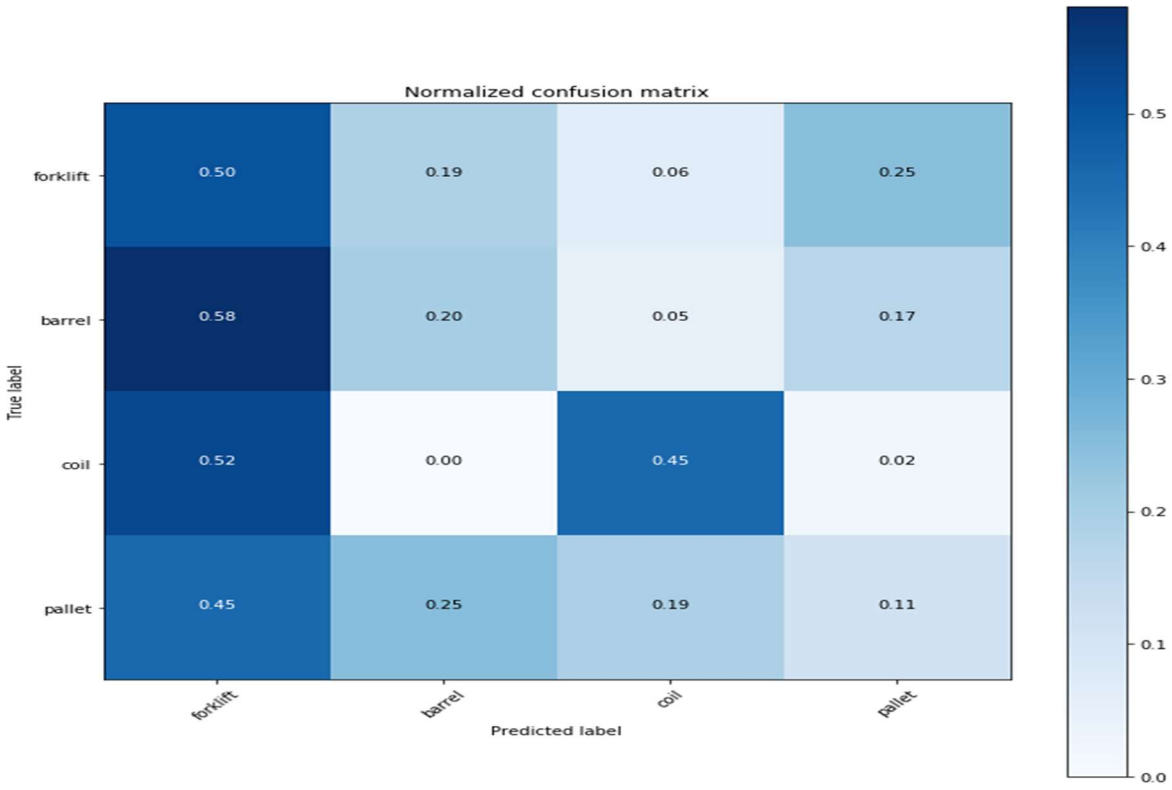


4.4.2 Training with simulated data and testing with actual data

We now tested our simulated data with the actual images and trained them with simulated data. The model gave an accuracy of around sixty percent.



In order to find the number of predicted images we generated a confusion matrix against the actual images.



4.4.3 Using Google Inception Model

OTTO VON GUERICKE
UNIVERSITÄT
MAGDEBURG

5.Conclusion

5.References

[1]. "Keras Documentation." http://keras.io/. Accessed 14 Apr. 2019.
[2]."Speed/accuracy trade-offs for modern convolutional object detectors." Huang J, Rathod V, Sun C, Zhu M, Korattikara A, Fathi A, Fischer I, Wojna Z, Song Y, Guadarrama S, Murphy K, CVPR 2017.