The Development of a Defect Detection System using Deep Learning and Computer Vision for the American Axle and Manufacturing (AAM) Valencia Facility's Advanced Inspection System

Tavin Ardell, BSCS Candidate, CoAS; CJ Chung, PhD, Professor, Advisor, CoAS
Department of Math and Computer Science, Lawrence Technological University, Southfield, MI 48075

Introduction

Manufacturing companies are looking for innovative ways to automate quality inspection processes to reduce cost, ensure reliability, and make processes more efficient. One of the major methods for this type of automation is by means of computer vision paired with deep learning. However, this type of technology can come at a very high cost to the manufacturing company. For example, the Advanced Inspection Cell at American Axles Valencia Spain facility has an estimated cost of $250,000.

The purpose of this project is to demonstrate the efficacy of a low cost defect detection system using a type of deep learning called object detection. One of the goals of this project is to produce more reliable defect detection than a traditional vision inspection system in a manufacturing facility. Along with this, it was important to maintain a small budget for this project in order to demonstrate how defect detection, and automation processes in general, can be implemented at a low cost utilizing the correct scientific and engineering principles. The defect detection system should be able to verify that newly manufactured steel parts are correctly identified as 'defective' and should be able to properly identify the location of a defect.

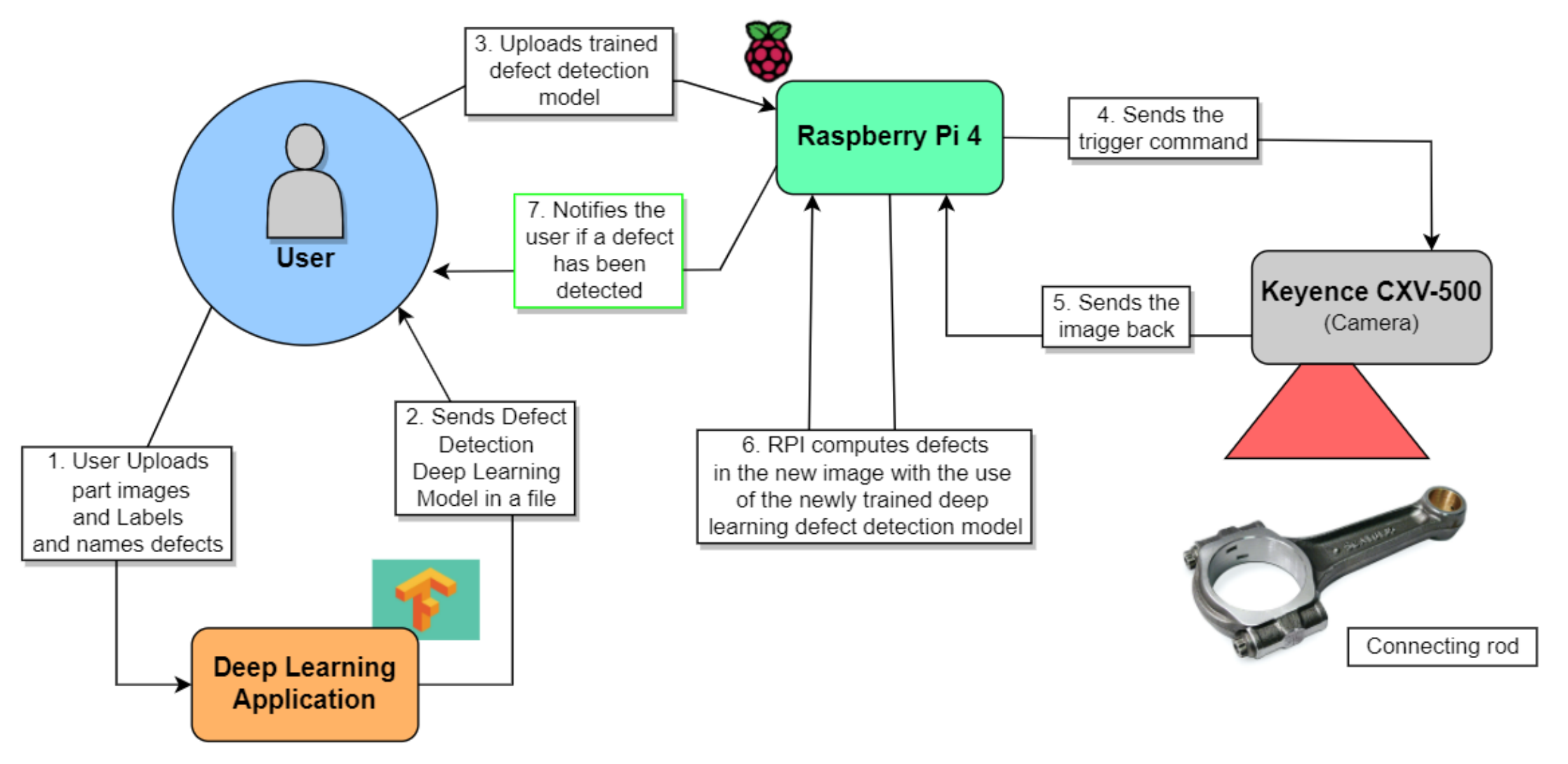
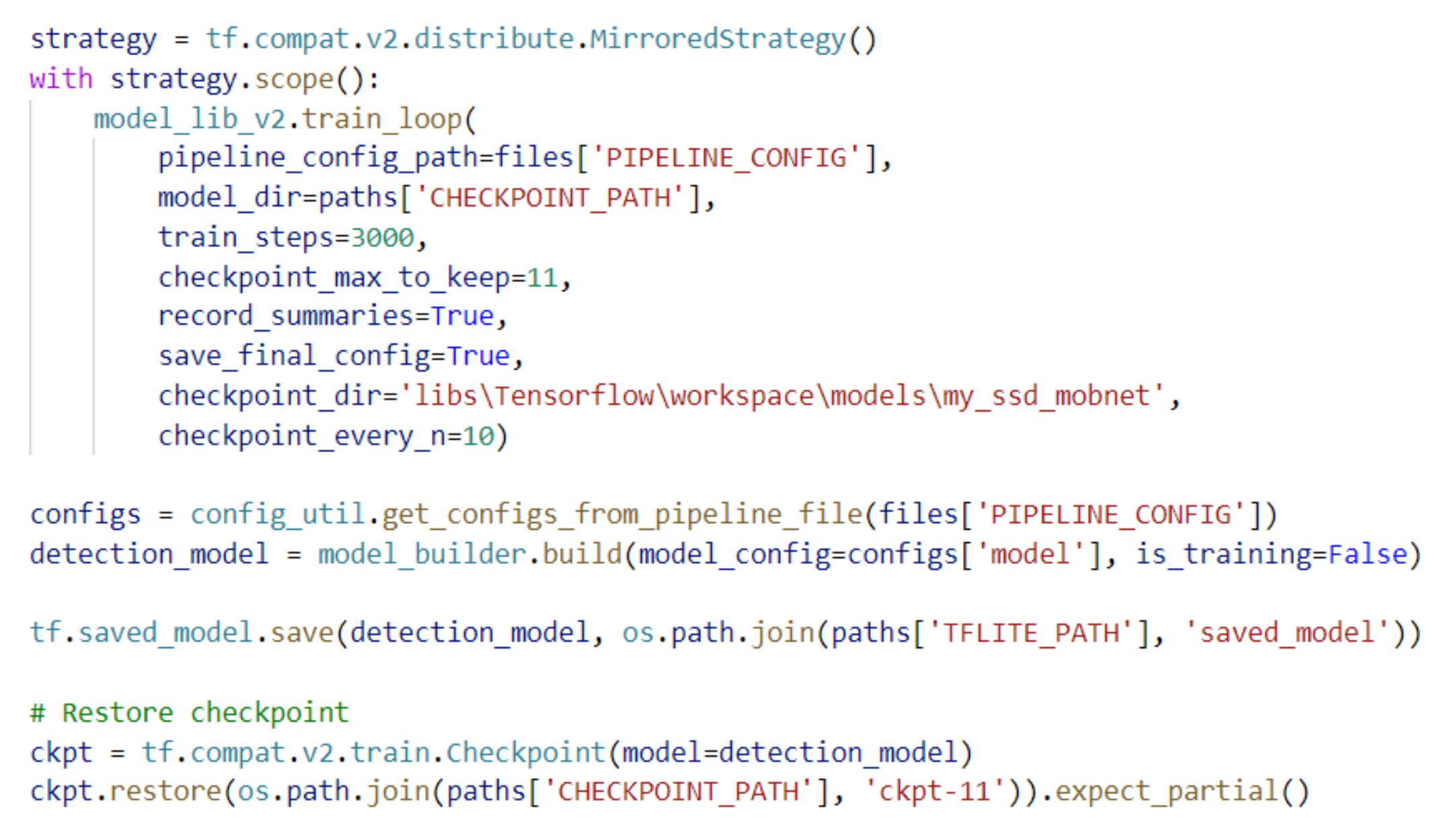

3. Uploads trained defect detection model
Raspberry Pi 4
4. Sends the trigger command
User
7. Notifies the user if a defect has been detected
Keyence CXV-500 (Camera)
5. Sends the image back
1. User Uploads part images and Labels and names defects
2. Sends Defect Detection Deep Learning Model in a file
6. RPI computes defects in the new image with the use of the newly trained deep learning defect detection model
Deep Learning Application
Connecting rod

Figure 1: Defect Detection System Overview
This system allows for a user to use an application to train a model that can be transferred to a raspberry pi. This Raspberry Pi will be integrated with a camera to detect defects in manufactured parts.

Defect Detection Model Training



```
strategy = tf.compat.v2.distribute.MirroredStrategy()
with strategy.scope():
    model_lib_v2.train_loop(
        pipeline_config_path=files['PIPELINE_CONFIG'],
        model_dir=paths['CHECKPOINT_PATH'],
        train_steps=3000,
        checkpoint_max_to_keep=11,
        record_summaries=True,
        save_final_config=True,
        checkpoint_dir='libs\Tensorflow\workspace\models\my_ssd_mobnet',
        checkpoint_every_n=10)

configs = config_util.get_configs_from_pipeline_file(files['PIPELINE_CONFIG'])
detection_model = model_builder.build(model_config=configs['model'], is_training=False)

tf.saved_model.save(detection_model, os.path.join(paths['TFLITE_PATH'], 'saved_model'))

# Restore checkpoint
ckpt = tf.compat.v2.train.Checkpoint(model=detection_model)
ckpt.restore(os.path.join(paths['CHECKPOINT_PATH'], 'ckpt-11')).expect_partial()
```

Figure 2: Object Detection Training Function
This function allows for a user to upload a directory of properly labeled defective part images and train a defect detection model. It also will upload a tfLite model that the user will be able to upload to a raspberry pi.

Design and Approach

The Deep Learning Application allows a user to train an object detection model for deep learning without any coding background. This application allows a user to upload images of defective parts, then label the defect within each part image with an associated name. After the labeling of each image's defect is completed, the user will be able to train a TFLite model that will properly identify the defects within part images. This application utilizes Google's TensorFlow software library, a Cuda and cuDNN GPU configuration, an SSD MobileNet V2 TensorFlow model, and a labelImg program to create a label map for training a deep learning model to correctly identify defects within part images. The figure below shows an example of the Defect Detection Training Application.

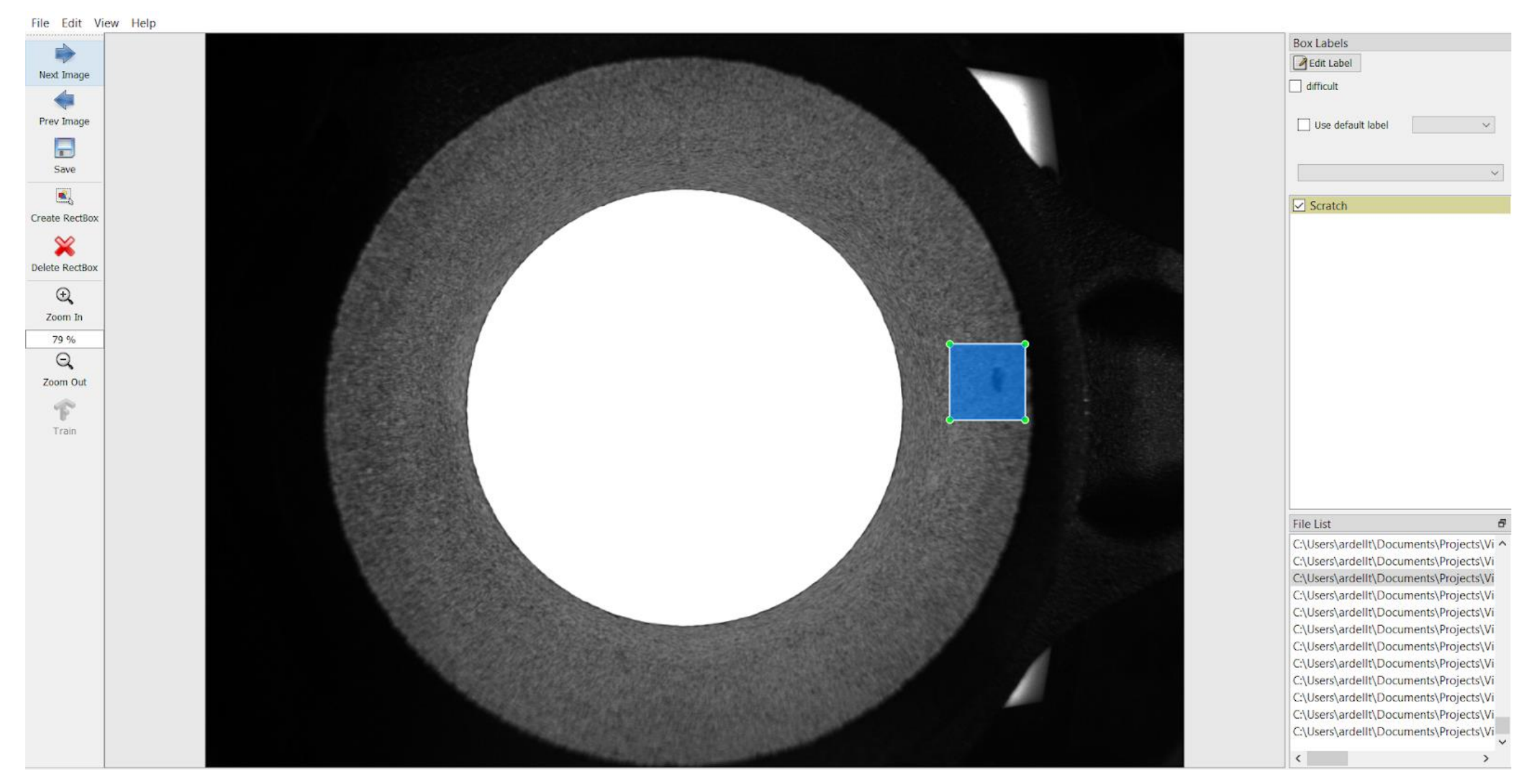


Figure 3: Defect Detection Application
This application integrates a SSD MobileNet V2 object detection model with Tzu Talin's labelImg software.

The implemented Defect Detection System uses a raspberry pi connected to a camera which allows the user to take images of manufactured parts, send those images to the Raspberry Pi running a deep learning model that will identify the defects within images, and notify the user if a part has defects and where the location of those defects are on the part image. The advantage of using a Raspberry Pi is that it is more cost effective than traditional PC's and it allows for a more compact end product for the consumer. Along with this, a user is able to use this system without any extensive computer science or computer networking background. This system requires a Raspberry Pi 4 with TensorFlow-lite and OpenCV as well as a working camera that is able to network to the Raspberry Pi.
Utilizing both of these systems, a user can reliably train, test, and implement a defect detection system for under 100 dollars.

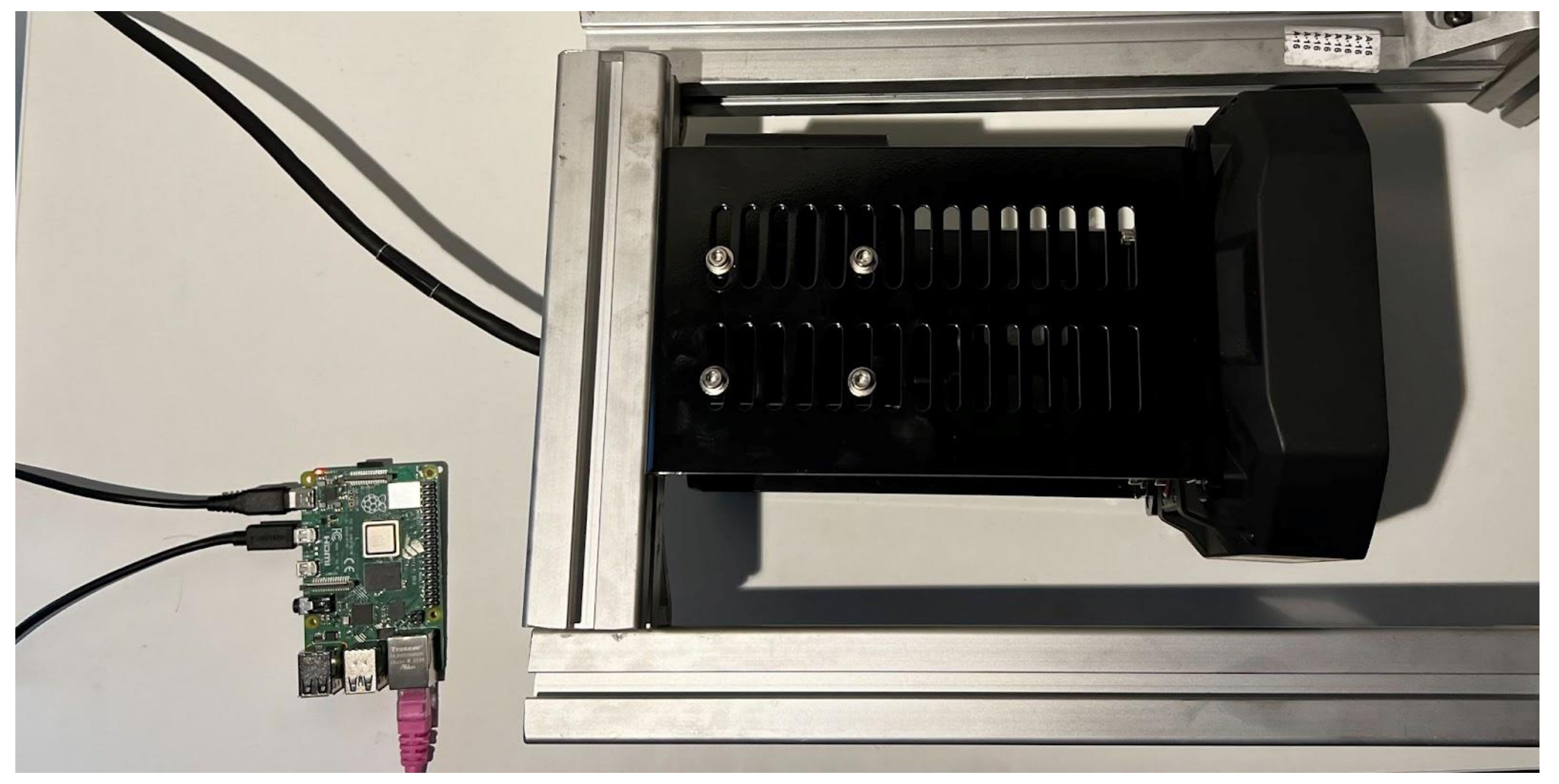


Figure 4: Defect Detection Implementation
This system uses a Raspberry Pi 4 and a Keyence CV-X482F connected via Ethernet take images and detect defects in newly imaged parts.

Testing

Testing for the model training software was done through a series of user cases. These cases involves things such as a user loading an image, successfully labeling a defect, saving an image, starting the initialization of the training process, and downloading the training model once it was successfully trained.

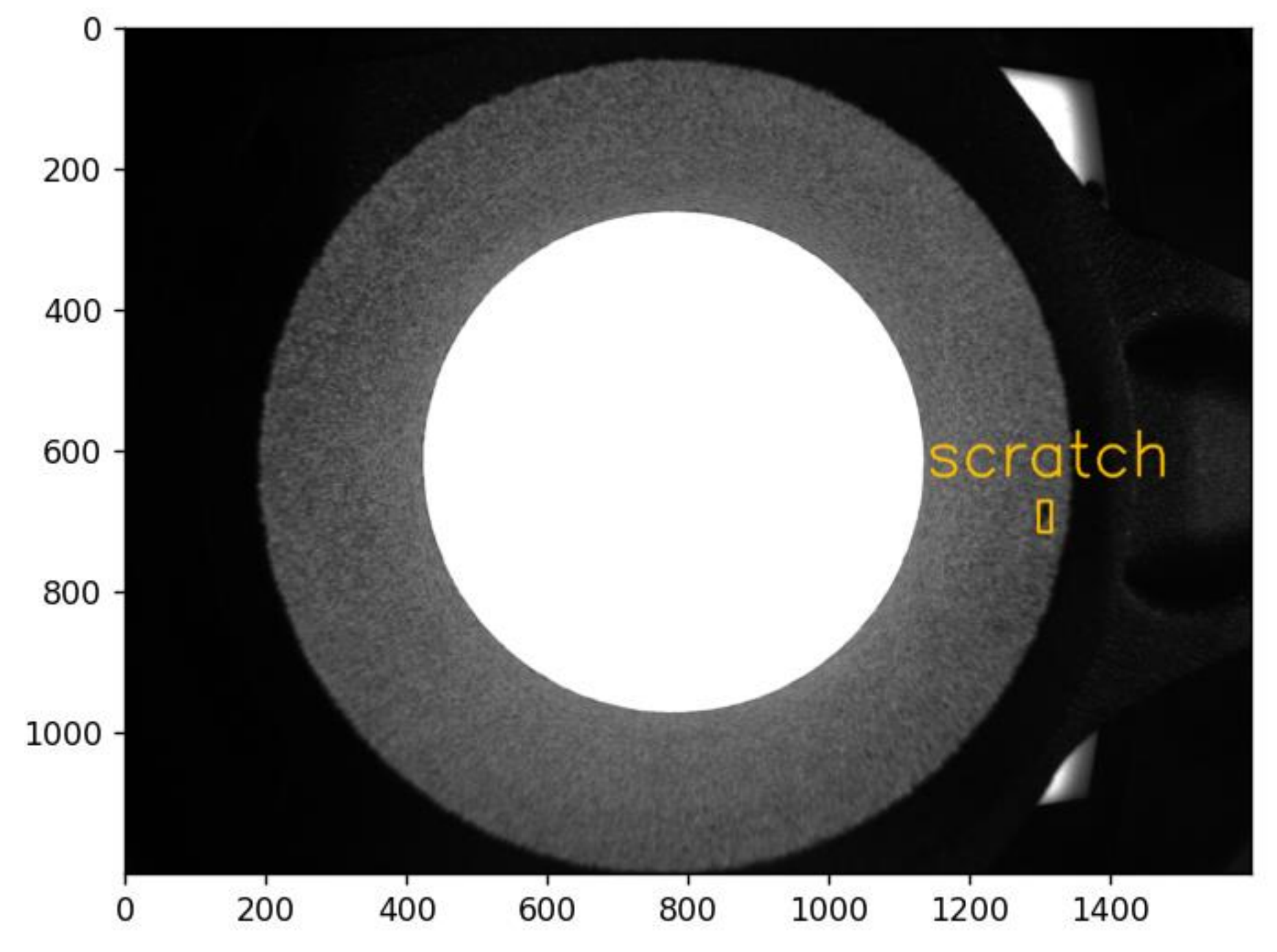


Figure 5: Outputted Defect Detection Image
Pictured above is a defected part image and what the user would see when the system has correctly identified a detect.

Testing for the implementation of the system involved taking an image with the camera, the raspberry pi successfully receiving the image from the camera, and then the Raspberry Pi correctly identifying the image as a defect or not a defect.
The implemented Defect Detection System was testing at the American Axle Valencia Manufacturing facility in Spain in order to further test the systems efficacy. The Defect Detection System was compared to their Advanced Inspection Cell vision technology. The testing involved comparing the false reject parts (parts which were labeled as defected parts but were actually non defective parts), the true reject parts, true reject parts, and false positives (defective parts that had been identified as non defective parts).

Discussion

The Defect Detection System proves that a cheap and effective automated defect detection system can be made using TensorFlow SSD MobileNet and a Raspberry Pi 4. The results have proven that this model works for steel parts with rough surfaces and variable defects. Future implementations could have the potential to view medical diagrams for diagnoses, review the quality of food being output in the service industry, or even detecting objects in the path of an autonomous vehicle. However, further research needs to be done in order to properly verify these claims.

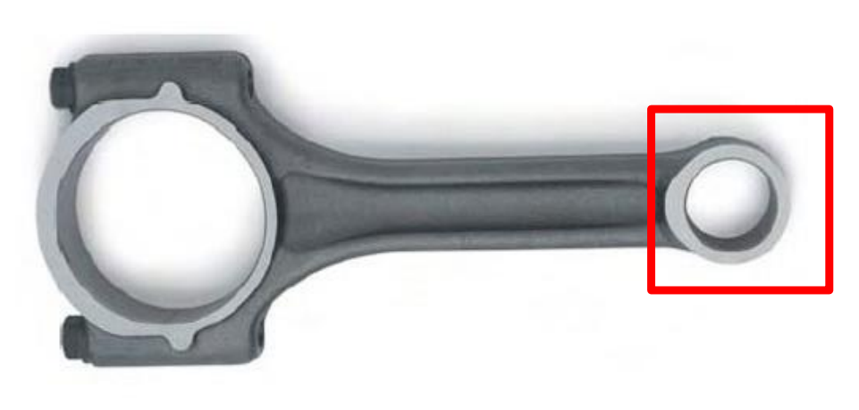


Figure 6: Connecting Rod
Pictured to the left is the steel part used for this project. Highlighted in red is the portion of the connecting rod that was tested for defect detection

Findings and Results

The Defect Detection Software was able to pass all of the test cases and was able to effectively train an object detection model for defect detection. However, one of the downsides to creating an application that uses the GPU on a users machine is it can be quite slow. For example, when training a model for the connecting rods, it would take about 4 hours on the Defect Detection Application, whereas it would only take about an hour on Google Colab.

The Implemented Defect Detection system in the Advanced Inspection Cell in Valencia Spain proved to have promising results. The false positive for Valencia's implemented vision system have an average false detection rate of 2.18% and the Defect Detection System was able to lower that percentage to 0.50%. The total rejection percentage for Valencia's vision system was 2.25% whereas, once again, the Defect Detection System outperformed their system with a percentage of 0.57%. Compared to Valencia's proprietary computer vision systems the Defect Detection System was able to increase the accuracy of detecting defects on manufactured parts by 1.71%, from 97.75% to 99.43% accuracy. The results show that the Defect Detection System not only out performs perennial vision systems, it is also approximately a thousandth of the cost.

Future Implementations

Defect Detection Application:
•Allow the user to select the type of model they wish to use for training
•Allow the user to select the number of epochs they wish to train the model
•Integrate a testing environment for unseen data
•Create a concise software package that a user can download and immediately start to train their own models
Defect Detection Implementation
•Further test the system in real world manufacturing lines
•Use different types of camera to verify different detection accuracies for different megapixel values
•Integrate a 'plug and play' system for the raspberry pi and different camera models that allows minimal configuration for the end user

Acknowledgements

This project was supported by American Axle and Manufacturing who helped provide the Raspberry Pi 4 as well as the Keyence CV-X482F camera. This project would not have been possible without the help of the American Axle facility in Valencia Spain who provided this project with defective part images and implemented this system into their Advanced Inspection Cell.

References

[1] Nicholas Renotte's Tensorflow Object Detection Course: https://www.youtube.com/watch?v=yqkISICHH-U&t=9207s
[2] Tzu Talin's LabelImg Github: https://github.com/tzutalin/labelImg
[3] Tensorflow Model Github: https://github.com/tensorflow/models
[4] SSD MobileNet V2 Architecture: https://tfhub.dev/tensorflow/ssd_mobilenet_v2/fpnlite_320x320