**COS40007 – Artificial Intelligence for Engineering**

**Portfolio 5-Studio 5-1**

Student Name: Nguyen Dinh Dung

Student ID: 104772138

A black and red sign with white text

Description automatically generated

Table of Contents

[Abstract 2](#_Toc190720667)

[Task 1: 2](#_Toc190720668)

[Task 2: 7](#_Toc190720669)

[Task 3: 8](#_Toc190720670)

# Abstract

This portfolio submission shows how CNN, ResNet50, and Mask R-CNN are used to create and assess deep learning models for image classification and object recognition tasks. Labeled datasets, model results, and source code organized as follows are all included in the submission:

- The LabelMe program and the associated JSON file were used to annotate ten photos in the Labeled Log Dataset.   
- CNN Model: Test results of a simple CNN model that was trained to distinguish between photos that include rust and those that do not, with the images and results stored in the "cnn\_test" folder.  
- Test results for the same classification job are stored in the "resnet50\_test" file for the ResNet50 model.   
- A model called Mask R-CNN was created to identify log objects in pictures. The model produces segmentation masks, bounding boxes, and confidence ratings as detection results. The "rcnn\_test" folder is where the results are stored.   
- Source Code: All models' source code may be found in the "code" folder.

# Task 1:

1. To begin, select ten rust and ten non-rust photos at random for testing (this is known as the Test Set). Therefore, these 20 photos won't be in your training set yet.

A screenshot of a computer program

AI-generated content may be incorrect.

1. Create a basic CNN model that resembles the minst classification model, but train it using the given corrosion data with the classes "rust" and "no rust" (without including the test set). After training and saving the model, test it using your test dataset and gauge its accuracy by correctly classifying 20 of the test set's images.

A screenshot of a computer program

Description automatically generated  
Result Tables:

|  |  |
| --- | --- |
| True Class | Predicted Class |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 0 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |

Accuracy: 100%

The result for **test outcome** containing **images** can be found at **cnn\_test** folder!

Next, create a more sophisticated CNN called Restnet50. Train it using the same dataset as in step 2, test it using the test dataset, and calculate its accuracy using the test set's 20 images.  
A screenshot of a computer program

AI-generated content may be incorrect.  
Result Tables:

|  |  |
| --- | --- |
| True Class | Predicted Class |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 0 | 0 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |
| 1 | 0 |
| 1 | 1 |
| 1 | 1 |
| 1 | 1 |

Accuracy: 95%

The result for **test outcome** containing **images** can be found at **resnet50\_test** folder!

# Task 2:

1. First, select ten images at random for testing (referred to as the Test Set). For this work, I have actually divided the training and testing in half directly through labelme2coco  
   A screenshot of a computer

   AI-generated content may be incorrect.
2. Using the labelled log as training data (excludes test set) to develop a Mask RCNN model  
   A screenshot of a computer program

   AI-generated content may be incorrect.  
   The train loss decreased to around 2.6 after I trained the model for about 17 epochs. I then trained the model for an additional 20 epochs and was able to reduce the train loss to about 0.6, which is a huge improvement over 2.6. It took about six to seven hours to complete.
3. Test the model with Test set and generate images of detected log objects along with confidence score. For example, the test outcome of one image will look like similar to the following image. You will need to use OpenCV to produce such image
4. Write a python program that count number of detected logs in each output image (Log counting)

As seen in the portfolio requirements PDF example, I have merged the responses for needs 3 and 4 to provide log counting, discovered objects, a confidence score, and segmentation. In order to eliminate poor or erroneous results that the models identified, I have also established a threshold. An illustration of the model's output may be found below.  
A log stack with numbers and text

AI-generated content may be incorrect.

The result for **test outcome** containing **images** can be found at **rcnn\_test** folder!

# Task 3:

The outcome of this task is available at my\_coco\_annotations, and the dataset is the JSON file that must be submitted.JSON

A screenshot of a computer

Description automatically generated