Student name: Nguyen Anh Tuan
Student ID: 103805949

Swinburne University of Technology

COS40007-Artificial Intelligence for Engineering

Portfolio 5

Nguyen Anh Tuan - 103805949

Studio class: Studio 1-1

Hanoi, Vietnam

Student name: Nguyen Anh Tuan
Student ID: 103805949

A. Task 1: Develop CNN and Resnet50

1. Randomly select 10 no rust images and 10 rust images into the Test set

```
[3] rust_dir = '/content/drive/MyDrive/Portfolio5/Corrosion/rust'
    no_rust_dir = '/content/drive/MyDrive/Portfolio5/Corrosion/no rust'
    test_dir = '/content/drive/MyDrive/Portfolio5/Task1Test'
```

```
[4] # Create test directories if they don't exist
    os.makedirs(f'{test_dir}/rust', exist_ok=True)
    os.makedirs(f'{test_dir}/no rust', exist_ok=True)
```

```
[ ] # Get the list of files from the rust and no rust directories
    rust_files = os.listdir(rust_dir)
    no_rust_files = os.listdir(no_rust_dir)

    # Randomly select 10 files from each directory
    selected_rust_files = random.sample(rust_files, 10)
    selected_no_rust_files = random.sample(no_rust_files, 10)

    # Move selected rust files to the test set
    for file in selected_rust_files:
        shutil.move(os.path.join(rust_dir, file), os.path.join(f'{test_dir}/rust', file))

    # Move selected no rust files to the test set
    for file in selected_no_rust_files:
        shutil.move(os.path.join(no_rust_dir, file), os.path.join(f'{test_dir}/no rust', file))

    print("10 rust and 10 no rust images for testing is created in the Task1Test folder. ")
```

```
10 rust and 10 no rust images for testing is created in the Task1Test folder.
```

2. CNN model

To begin with, I declared global constants and conducted the data augmentation part for the model.

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
[6] # Constant
    train_dir = '/content/drive/MyDrive/Portfolio5/Corrosion'

    BATCH_SIZE = 32
    IMG_WIDTH, IMG_HEIGHT = 128, 128
    EPOCHS = 10
    NUM_CLASSES = 2  # rust and no rust

    train_data = ImageDataGenerator(rescale=1./255)
    test_data = ImageDataGenerator(rescale=1./255)

    train_generator = train_data.flow_from_directory(
                train_dir,
                target_size=(IMG_WIDTH, IMG_HEIGHT),
                batch_size=BATCH_SIZE,
                class_mode='binary',
                color_mode='rgb' )

    test_generator = test_data.flow_from_directory(
                test_dir,
                target_size=(IMG_WIDTH, IMG_HEIGHT),
                batch_size=BATCH_SIZE,
                class_mode='binary',
                color_mode='rgb',
                shuffle=False)
```

```
shuffle=False)

Found 9 images belonging to 2 classes.
Found 20 images belonging to 2 classes.
```

The image below is the CNN model architecture.

```
[ ] model = Sequential([
        Input(shape=(IMG_WIDTH, IMG_HEIGHT, 3)),  # Input shape for RGB images (3 channels)
        Conv2D(8, 3, activation='relu'),
        MaxPooling2D(pool_size=2),
        Flatten(),
        Dense(10, activation='relu'),
        Dense(1, activation='sigmoid'),  # Output layer for binary classification
    ])
```

For the CNN model, I also use the Adam optimizer, loss function 'categorical_crossentropy', and the metric ‘accuracy

```
# Compile the model
model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])
```

Then, I train the model with 10 epochs.

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
[ ] model.fit(
        train_generator,
        epochs=EPOCHS,
        validation_data=test_generator
    )
```

However, since the Corrosion dataset is small, the accuracy is lower than I expected

```
[ ] # Evaluate the model
    loss, accuracy = model.evaluate(test_generator)
    print(f'Test accuracy: {accuracy * 100:.2f}%')

    1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 122ms/step - accuracy: 0.5000 - loss: 0.0000e+00
    Test accuracy: 50.00%
```

After that, I created a test outcome of 20 images for the CNN model table containing true class and predicted class and final overall accuracy. Then, save it in a CSV file

```
[ ] # Test Set prediction
    model_output = model.predict(test_generator)
    predicted_classes = np.round(model_output).astype(int)  # Round model_output to get 0 or 1

    true_classes = test_generator.classes

    class_labels = list(test_generator.class_indices.keys())

    # Test Set prediction
    model_output = model.predict(test_generator)
    predicted_classes = np.round(model_output).astype(int)  # Round model_output to get 0 or 1

    # True labels
    true_classes = test_generator.classes

    # Class labels
    class_labels = list(test_generator.class_indices.keys())

    # Create a table of true class and predicted class
    comparison_df = pd.DataFrame({
        'Filename': test_generator.filenames,
        'True Class': [class_labels[int(i)] for i in true_classes],
        'Predicted Class': [class_labels[int(i)] for i in predicted_classes]
    })

    #Get the result
    print(comparison_df)
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 388ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 125ms/step
                      Filename True Class Predicted Class
0   no rust/001_kwmsqgk2.yfv.jpg    no rust         no rust
1   no rust/002_2bvturws.zss.jpg    no rust         no rust
2   no rust/002_dr3selb3.a1h.jpg    no rust         no rust
3   no rust/002_mjcgc4b1.ru0.jpg    no rust         no rust
4   no rust/002_yoamnba2.wdq.jpg    no rust         no rust
5   no rust/002_zar1fupp.kf2.jpg    no rust         no rust
6   no rust/003_zomak3wq.43c.jpg    no rust         no rust
7   no rust/006_2vq3zs35.40h.jpg    no rust         no rust
8   no rust/006_esktmdp0.1e4.jpg    no rust         no rust
9   no rust/006_pwvc1uiv.zhi.jpg    no rust         no rust
10     rust/001_w1nojjln.bdv.jpg       rust         no rust
11     rust/002_cql5orz2.p0y.jpg       rust         no rust
12     rust/003_glbjresp.01k.jpg       rust         no rust
13     rust/004_1xgwoahx.lqq.jpg       rust         no rust
14     rust/004_i1dmciub.oop.jpg       rust         no rust
15     rust/004_uouhxp2m.0ax.jpg       rust         no rust
16     rust/005_txq3nygx.w0v.jpg       rust         no rust
17     rust/007_1upl22nx.1gb.jpg       rust         no rust
18     rust/007_1vtpg0jx.nkh.jpg       rust         no rust
19     rust/007_2pzsqkji.l3n.jpg       rust         no rust
<ipython-input-12-b73ebf7d6ea6>:23: DeprecationWarning: Conversion of an array with ndim > 0 to a scalar is deprecated,
  'Predicted Class': [class_labels[int(i)] for i in predicted_classes]
```

```
[ ] comparison_df.to_csv('cnn_truth_predict_table.csv')
```

Finally, I saved the test outcome of 20 images for the CNN model in the ‘cnn_test’ folder.

```
[ ] cnn_dir = 'cnn_test'

    os.makedirs(cnn_dir, exist_ok=True) # CNN test directory
```

```
[ ] # Save images with predicted labels
    for i, filename in enumerate(test_generator.filenames):
        img_path = os.path.join(test_dir, filename)
        imgBGR = cv2.imread(img_path)  # Read the image in BGR format
        imgRGB = cv2.cvtColor(imgBGR, cv2.COLOR_BGR2RGB)  # Convert to RGB format

        # Get the true and predicted class labels
        true_label = class_labels[true_classes[i]]
        predicted_label = class_labels[predicted_classes[i][0]]

        # Add predicted class to the image
        imgRGB = cv2.putText(
            imgRGB, f"True: {true_label} | Predicted: {predicted_label}", (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2, cv2.LINE_AA
        )

        # Save the output image
        cnn_img_path = os.path.join(cnn_dir, f"test_image_{i+1}.jpg")
        cv2.imwrite(cnn_img_path, cv2.cvtColor(imgRGB, cv2.COLOR_RGB2BGR))
```

3. ResNet50 model

For the ResNet50 model, first, I declared global constants.

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
[17] # Constant
     BATCH_SIZE = 32
     IMG_SIZE = 224
     NUM_CLASSES = 2

     NUM_EPOCHS = 10
     EARLY_STOP_PATIENCE = 3
     BATCH_SIZE_TRAINING = 32
     BATCH_SIZE_VALIDATION = 32



     train_dir = '/content/drive/MyDrive/Portfolio5/Corrosion'
```

Then, I augment the data.

```
[ ]  data_generator = ImageDataGenerator(preprocessing_function=preprocess_input)

     train_generator = data_generator.flow_from_directory(
         train_dir,
         target_size=(IMG_SIZE, IMG_SIZE),
         batch_size=BATCH_SIZE_TRAINING,
         class_mode='categorical'
     )

     validation_generator = data_generator.flow_from_directory(
         test_dir,
         target_size=(IMG_SIZE, IMG_SIZE),
         batch_size=BATCH_SIZE_VALIDATION,
         class_mode='categorical'
     )

     steps_per_epoch = train_generator.samples // BATCH_SIZE_TRAINING
     validation_steps = validation_generator.samples // BATCH_SIZE_VALIDATION

Found 9 images belonging to 2 classes.
Found 20 images belonging to 2 classes.
```

The image below shows how I model the ResNet50 architecture.

```
[ ] # Resnet50 model
    base_model = ResNet50(include_top=False, weights='imagenet',
                          input_tensor=Input(shape=(IMG_SIZE, IMG_SIZE, 3)))
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(NUM_CLASSES, activation = 'softmax')(x)
    model = Model(inputs = base_model.input, outputs=x)

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
```

```
[ ] # Freeze the base model layers to prevent training them
    for layer in base_model.layers:
        layer.trainable = False
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

For the model's optimizer, I use SGD. The loss function 'categorical_crossentropy', and the metric 'accuracy', EarlyStopping and ModelCheckpoint are also used to train the model.

```
sgd = SGD(learning_rate=0.01, momentum=0.9, nesterov=True)
model.compile(optimizer=sgd, loss = 'categorical_crossentropy', metrics = ['accuracy'])
```

```
cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = EARLY_STOP_PATIENCE)
cb_checkpointer = ModelCheckpoint(filepath = 'best_model.keras', monitor = 'val_loss', save_best_only = True, mode = 'auto')
```

```
fit_history = model.fit(
    train_generator,
    steps_per_epoch= steps_per_epoch,
    epochs = NUM_EPOCHS,
    validation_data = validation_generator,
    validation_steps= validation_steps,
    callbacks = [cb_checkpointer, cb_early_stopper]
)

# Load the best model weights
model.load_weights('best_model.keras')
```

This is the test accuracy result.

```
# Evaluate the Model on Test Data
test_loss, test_accuracy = model.evaluate(validation_generator)
print(f'Test accuracy: {test_accuracy * 100:.2f}%')
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 201ms/step - accuracy: 0.7500 - loss: 0.4732
Test accuracy: 75.00%
```

After that, I generate a table containing true class and predicted class and overall accuracy for the test outcome of 20 images for RestNet50 model and save it in a CSV file.

```
validation_generator.reset()  # Resetting the generator for new predictions
predictions = model.predict(validation_generator, steps = validation_steps, verbose=1)
predicted_class_indices = np.argmax(predictions, axis=1)

# True class indices from the validation generator
true_class_indices = validation_generator.classes
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/models/functional.py:237: UserWarning: The structure of `inputs` doesn't match the expected structure.
Expected: ['keras_tensor_6']
Received: inputs=Tensor(shape=(20, 224, 224, 3))
  warnings.warn(msg)
1/1 ━━━━━━━━━━━━━━━━━━━━ 4s 4s/step
```

```
true_class_indices = validation_generator.classes

class_labels = list(validation_generator.class_indices.keys())
predicted_labels = [class_labels[i] for i in predicted_class_indices]
true_labels = [class_labels[i] for i in true_class_indices]

compare_df1 = pd.DataFrame({
    'Filename': validation_generator.filenames,
    'True Class': true_labels,
    'Predicted Class': predicted_labels
})

print(compare_df1)
```

Student name: Nguyen Anh Tuan
Student ID: 103805949

```
                    Filename True Class Predicted Class
0   no rust/001_kwmsqgk2.yfv.jpg    no rust           rust
1   no rust/002_2bvturws.zss.jpg    no rust        no rust
2   no rust/002_dr3selb3.a1h.jpg    no rust        no rust
3   no rust/002_mjcgc4b1.ru0.jpg    no rust           rust
4   no rust/002_yoamnba2.wdq.jpg    no rust           rust
5   no rust/002_zar1fupp.kf2.jpg    no rust           rust
6   no rust/003_zomak3wq.43c.jpg    no rust        no rust
7   no rust/006_2vq3zs35.40h.jpg    no rust           rust
8   no rust/006_esktmdp0.le4.jpg    no rust           rust
9   no rust/006_pwvc1uiv.zhi.jpg    no rust           rust
10     rust/001_w1nojjln.bdv.jpg       rust        no rust
11     rust/002_cql5orz2.p0y.jpg       rust        no rust
12     rust/003_glbjresp.01k.jpg       rust           rust
13     rust/004_1xgwoahx.lqq.jpg       rust        no rust
14     rust/004_i1dmciub.oop.jpg       rust           rust
15     rust/004_uouhxp2m.0ax.jpg       rust        no rust
16     rust/005_txq3nygx.w0v.jpg       rust           rust
17     rust/007_1upl22nx.1gb.jpg       rust           rust
18     rust/007_1vtpg0jx.nkh.jpg       rust           rust
19     rust/007_2pzsqkji.l3n.jpg       rust           rust
```

```
[ ] compare_df1.to_csv('resnet50_true_predict-table.csv')
```

```
[ ] # Calculate final accuracy
    correct_predictions = np.sum(predicted_class_indices == true_class_indices)
    final_accuracy = correct_predictions / len(true_class_indices)

    print(f'Final Overall Accuracy: {final_accuracy * 100:.2f}%')
```

```
Final Overall Accuracy: 45.00%
```

Following that, I save the test outcome of 20 images for the ResNet50 model in the ‘resnet50_test’ folder.

```
[ ] resnet_dir = 'resent50_test'

    os.makedirs(resnet_dir, exist_ok=True) # Resnet50 test directory
```

```
[ ] # Save images with predicted labels
    for i, filename in enumerate(validation_generator.filenames):
        img_path = os.path.join(test_dir, filename)
        imgBGR = cv2.imread(img_path)  # Read image in BGR format
        imgRGB = cv2.cvtColor(imgBGR, cv2.COLOR_BGR2RGB)  # Convert to RGB format

        # Get the predicted class label
        predicted_class = predicted_labels[i]

        # Add predicted class to the image
        imgRGB = cv2.putText(
            imgRGB, f"True: {true_label} | Predicted: {predicted_label}", (10, 30),
            cv2.FONT_HERSHEY_SIMPLEX, 0.6, (255, 0, 0), 2, cv2.LINE_AA
        )

        # Save the output image
        resnet_img_path = os.path.join(resnet_dir, f"test_image_{i+1}.jpg")
        cv2.imwrite(resnet_img_path, cv2.cvtColor(imgRGB, cv2.COLOR_RGB2BGR))
```

B. Task 2: Develop Mask RCNN for detecting log

The mask RCNN in this task has some library issues, so the code cannot be run. Therefore, skip this part.

Student name: Nguyen Anh Tuan
Student ID: 103805949

C. Task 3: Extending log labeling to another class

In this task, I update the labels of those 10 images using labelme and replace the label of the logs that are broken as detected_log.

For more details, access this link:

- 10 labelled images (JSON + Png files): https://drive.google.com/drive/folders/1n8ulx-jpEWH_GGrqr1mS3hzlScCaAslQ?usp=sharing
- 10 labeled images (output): https://drive.google.com/drive/folders/1csfhUBIgLTrcVtcsreKviyarA2BlZ3EP?usp=sharing

D. Appendix:

- Source code: https://colab.research.google.com/drive/1hkuivSvOUhNSPZMLiMmZ3icGQS1eOo4y?usp=sharing
- Dataset: https://drive.google.com/drive/folders/1mL-nN2fSl6iYHVu0R20OMUs0avCjNR-7