Curtin University – Department of Computing

Assignment Cover Sheet / Declaration of Originality

Complete this form if/as directed by your unit coordinator, lecturer or the assignment specification.

Last name:		Student ID:	
Other name(s):			
Unit name:		Unit ID:	
Lecturer / unit coordinator:		Tutor:	
Date of submission:		Which assignment?	(Leave blank if the unit has only one assignment.)

I declare that:

- The above information is complete and accurate.
- The work I am submitting is *entirely my own*, except where clearly indicated otherwise and correctly referenced.
- I have taken (and will continue to take) all reasonable steps to ensure my work is *not accessible* to any other students who may gain unfair advantage from it.
- I have *not previously submitted* this work for any other unit, whether at Curtin University or elsewhere, or for prior attempts at this unit, except where clearly indicated otherwise.

I understand that:

- Plagiarism and collusion are dishonest, and unfair to all other students.
- Detection of plagiarism and collusion may be done manually or by using tools (such as Turnitin).
- If I plagiarise or collude, I risk failing the unit with a grade of ANN ("Result Annulled due to Academic Misconduct"), which will remain permanently on my academic record. I also risk termination from my course and other penalties.
- Even with correct referencing, my submission will only be marked according to what I have done myself, specifically for this assessment. I cannot re-use the work of others, or my own previously submitted work, in order to fulfil the assessment requirements.
- It is my responsibility to ensure that my submission is complete, correct and not corrupted.

Signature: ______________________ Date of signature: ____________

(By submitting this form, you indicate that you agree with all the above text.)

Assignment
COMP3007
Tanaka Chitete

1 Statement of Work Done

I, Tanaka Chitete, hereby state that this assignment has been completed in full–the extent detailed in the specification.

2 Building signage detection and digit extraction

2.1 Approach

2.1.1 Loading and preprocessing an image

In the image preprocessing stage, the program applies bit manipulation, blackhat, opening, closing and thresholding operations in order to prepare the image to have its building signage extracted.

The source image (Figure 1) is read into memory by the program and then converted from BGR colour-space to grayscale colour-space (Figure 2). This is due to the fact that the colour of the image is not necessary to the implementation of the algorithm. Therefore, for the sake of efficiency, the image is converted into grayscale.

Figure 1. Original image

Figure 2. Grayscale image

Thereafter, the image is inverted using bit manipulation to prepare the image for the subsequent blackhat operation (Figure 3). This is done because the blackhat operation enhances dark objects on a comparatively bright background.

Figure 3. Inverted image

Soon after, the blackhat operation is applied to the inverted image in order to increase the brightness of the area where the numbers and building signage was identified (Figure 4).

Figure 4. Blackhat image

The opening operation is then used to remove any noise present in the blackhat image (Figure 5).

Figure 5. Opened image

As a follow-up, the closing operation is utilised to then turn the digits into a bright white blob which will be significantly easier to detect (Figure 6).

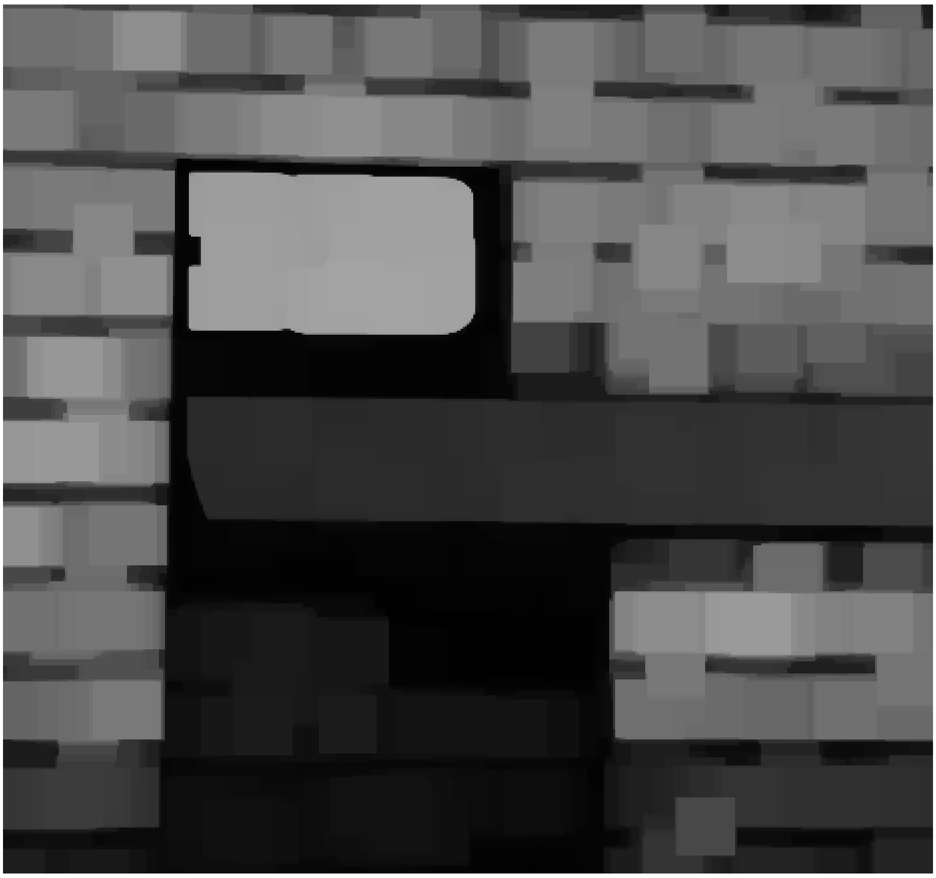


Figure 6. Closed image

Finally, the image is then thresholded to extract the blob representing the numbers (since it is the brightest spot in the image) (Figure 7).

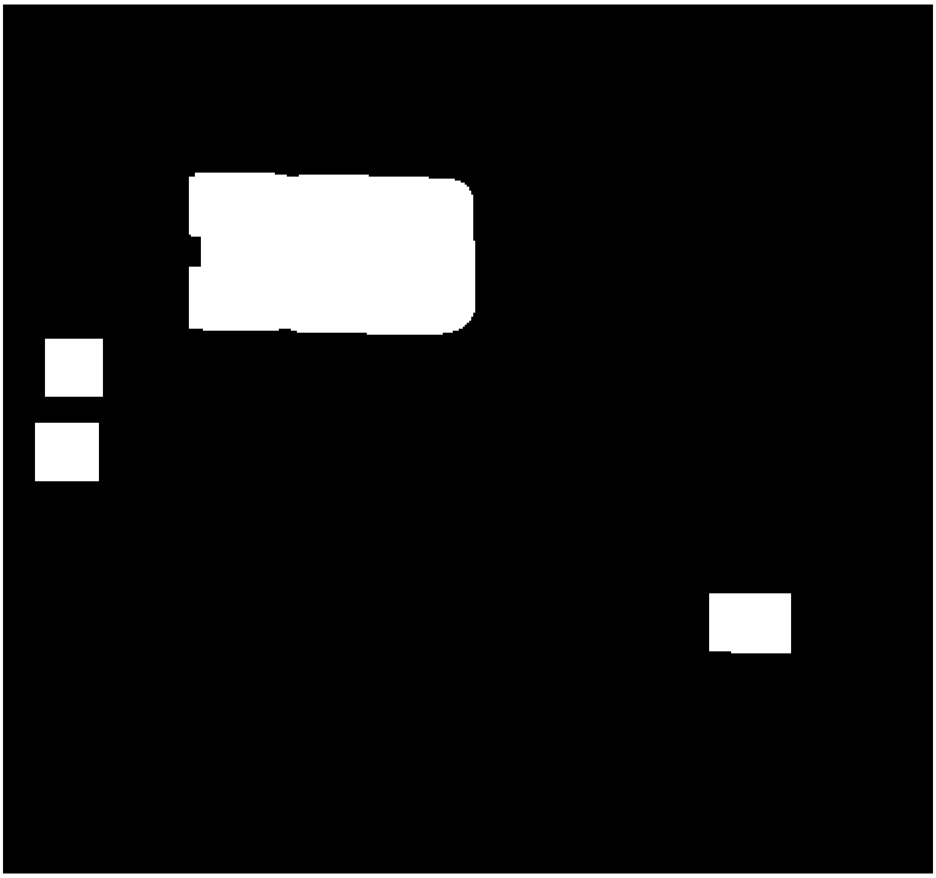


Image 7. Thresholded image

2.1.3 Extracting the building signage

After the blobs have been detected, the contours of the image are found and then sorted by size (where the largest contour represents the building signage) (Figure 8). Although the bounding box is not drawn at this stage, see below for the placement of the largest bounding box (again, representing building signage).

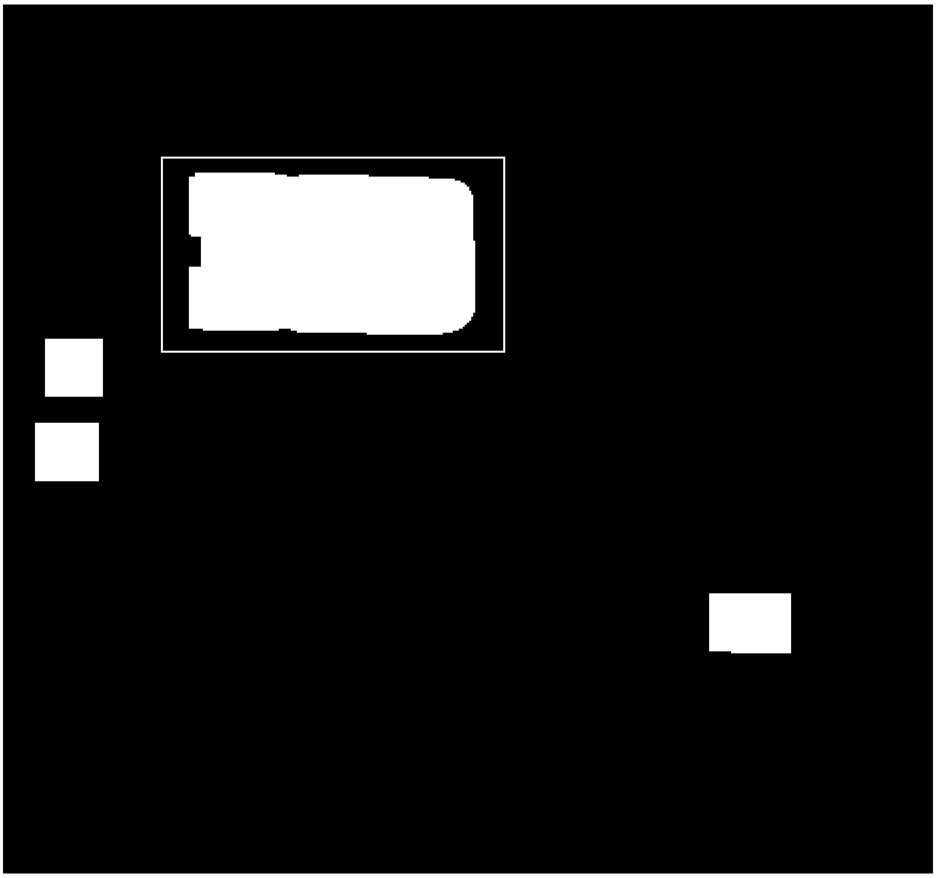


Figure 8. Image with bounding box around building signage

After the bounding box has been placed around the blob representing the building signage, the blackhat image is then cropped using a tightened bounding box to remove any additional noise or unwanted artefacts (e.g. parts of letters or bricks). The blackhat image is used for cropping since it forms the basis of the next major operation–extracting the digits (Image 9).

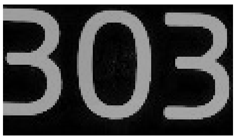


Figure 9. Cropped image

2.1.4 Preprocessing the building signage

Once the building signage has been extracted, the program then applies yet another threshold to the image. This time, however, the threshold is applied to greater differentiate between the light and dark pixels of the image (Figure 10).

Figure 10. Thresholded cropped image

2.1.5 Extracting the digits

Next, the program undergoes the process of extracting the digits present in the cropped image. To do so, the contours of the cropped image are found and then also sorted by size (where the largest three contours represent the digits of the building signage) (Figure 11). Again, although the bounding boxes are not drawn at this stage, see below for the placement of the largest bounding boxes (again, representing the digits of the building signage).

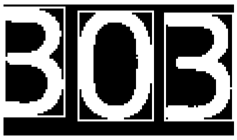


Figure 11. Cropped image with bounding boxes around digits of building signage

2.1.6 Drawing the bounding boxes

Since the program previously detected the building signage, it can now draw the bounding box on the original image (to be specific, a copy of it) (Figure 12).

Figure 12. Original image with bounding box around building signage

Immediately afterwards, the program then also draws the bounding boxes of the previously detected digits (Figure 13).

Figure 13. Original image with bounding box around building signage and bounding boxes around digits of building signage

At this stage, the execution of the program for a single image comes to an end and the process repeats for the remaining images in the directory.

2.2 Performance

2.2.1 Discussion of evaluation criteria

As per the specification, the program is considered working if:

1. It extracts all three digits
2. It draws detected areas with reference to the following criteria:
    2.1. The detected areas are rectangular in shape and do not exceed the maximum allowable size, which is determined by an intersection over unions (IoU) greater than 0.5
    2.2. The detected areas contain all digits of the building number.

However, I propose that criterion 2.2 shall be reworded as follows:

The detected areas contain all digits of the building number. However, if not all digits are extracted, the detected areas contain all extracted digits.

This is due to the fact that criteria 1 and 2.2 would result in marks being lost multiple times for the same error.

Therefore, the marking allocation shall be as follows:

- Criterion 1: 0.5 marks
- Criterion 2: 0.5 marks
    - Criterion 2.1: 0.25 marks
    - Criterion 2.2: 0.25 marks

2.2.2 Presentation of results

The results of the program were as follows:

Filename	Criterion			Total
	1	2.1	2.2	
BS01.jpg				1
BS02.jpg				1
BS03.jpg				1
BS04.jpg				1
BS05.jpg				1
BS06.jpg				1
BS07.jpg				0.5
BS08.jpg				1
BS09.jpg				1
BS10.jpg				1
BS11.jpg				1
BS12.jpg				1
BS13.jpg				1
BS14.jpg				1
BS15.jpg				1
Val01.jpg				1
Val02.jpg				0
Val03.jpg				1
Val04.jpg				1

Val05.jpg				1
Val06.jpg				1
Val07.jpg				1
Val08.jpg				1
Val09.jpg				1
Val10.jpg				1

Table 1. Results from the building signage recogniser

Ultimately, the final mark is 23.5/25 (94%).

3 Coral image classification

3.1 ResNet-18

3.1.1 Model selection

Through analysis of AlexNet, VGG16, and ResNet-18–the most notable modern machine learning models–it was found that the most performant model was ResNet-18. Therefore, the subsequent section will be discussed in reference to ResNet18. However, the results from each of the models are as follows:

AlexNet

The optimal hyperparameters for this model, `learning_rate`, `num_epochs`, and `batch_size`, were found to be `0.01`, `10`, and `256` respectively. The results were as follows:

Epoch	Loss	Training Accuracy
1	1.167	0.487
2	0.820	0.604
3	0.578	0.791
4	0.633	0.703
5	0.405	0.854
6	0.678	0.661
7	0.401	0.828
8	0.443	0.819
9	0.264	0.890

10	0.337	0.865

Table 2. Results from the fine-tuned AlexNet model

Ultimately, the validation accuracy was found to be 0.955 (95.5%).

VGG16
The optimal hyperparameters for this model, `learning_rate`, `num_epochs`, and `batch_size`, were found to be `0.05`, `10`, and `128` respectively. The results were as follows:

Epoch	Loss	Training Accuracy
1	0.621	0.686`
2	0.758	0.607
3	0.596	0.677
4	0.548	0.741
5	0.427	0.809
6	0.302	0.875
7	0.227	0.919
8	0.231	0.904
9	0.078	0.978
10	0.135	0.951

Table 3. Results from the fine-tuned VGG16 model

Ultimately, the validation accuracy was evaluated to be 0.968 (96.8%).

ResNet-18
The optimal hyperparameters for this model, `learning_rate`, `num_epochs`, and `batch_size`, were found to be `0.05`, `10`, and `256` respectively. The results are as follows:

Epoch	Loss	Training Accuracy
1	2.518	0.615
2	0.087	0.970
3	0.111	0.962
4	0.030	0.995
5	0.033	0.991

6	0.011	1.000
7	0.008	1.000
8	0.007	1.000
9	0.007	1.000
10	0.005	1.000

Table 4. Results from the fine-tuned ResNet-18 model

Ultimately, the validation accuracy was evaluated to be 0.995 (99.5%).

Therefore, it is quite evident that the fine-tuned ResNet-18 model would be the best modern model for the task.

3.1.1 Approach

3.1.1.1 Configuring output dimensions

Considering that the image classification problem consists of classifying images as coral or non-coral, it was therefore essential to change the output dimensions of the ResNet-18 model. With this in mind, I changed the dimensions of the final layer to have `out_features` changed to 2 (representing coral and non-coral images).

3.1.1.1 Loading and processing an image dataset

Since the original ResNet-18 neural network was trained using normalised images of dimensions 224 x 224, when loading and preprocessing the dataset to further train the pre-trained model on, a series of normalisations (defined by `normalize`) and transforms (defined by `transforms`) were applied to pre-process the dataset. Thereafter, a data loader for the training dataset (defined as `train_dataloader`) and a data loader for the validation dataset (defined as `val_dataloader`) were created in order to load images into memory in batches for efficient computation.

3.1.1.1 Training the model

Through hyperparameter tuning, it was found that the optimal learning rate (defined as `learning_rate`) and number of epochs (defined as `epochs`) were `0.05` and `10` respectively. Thereafter, the model was then trained over a series of 10 epochs using the learning rate of 0.05, achieving a training accuracy of 0.981 (98.1% by the end of the 10th epoch).

3.1.1 Performance

3.1.1.1 Discussion of evaluation criteria

Rather intuitively, the performance of the model shall be deemed by its ability to correctly predict new and unknown images, obtained from the completely new and unknown validation dataset. The model makes a series of predictions on the validation dataset and then the program evaluates the overall accuracy, which was ultimately determined to be 0.995 (99.5%).

3.1.1.2 Presentation of results

The results during training per epoch are given as follows:

Epoch	Loss	Accuracy
1	2.518	0.615
2	0.087	0.970
3	0.111	0.962
4	0.030	0.995
5	0.033	0.991
6	0.011	1.000
7	0.008	1.000
8	0.007	1.000
9	0.007	1.000
10	0.005	1.000

Table 4. Results from fine-tuning ResNet-18

Ultimately, the validation accuracy was evaluated to be 0.995 (99.5%).

3.2 Support Vector Machine

3.1.1 Model selection

Through analysis of both K-Nearest Neighbours (KNN), and Support Vector Machines (SVM) classifiers–the most notable classical machine learning models–it was found that the most performant model was an SVM. Therefore, the subsequent section will be discussed in reference to an SVM. The results from each of the models are as follows:

KNN Classifier

The optimal hyperparameters for the model, `n_neighbours`, was found to be 5. The results are as follows:

```
Classification report for classifier KNeighborsClassifier():
              precision    recall  f1-score   support

           0       0.84      0.97      0.90       200
           1       0.96      0.81      0.88       200

    accuracy                           0.89       400
   macro avg       0.90      0.89      0.89       400
weighted avg       0.90      0.89      0.89       400
```

Table 5. Results from training the KNN Classifier (classification report)

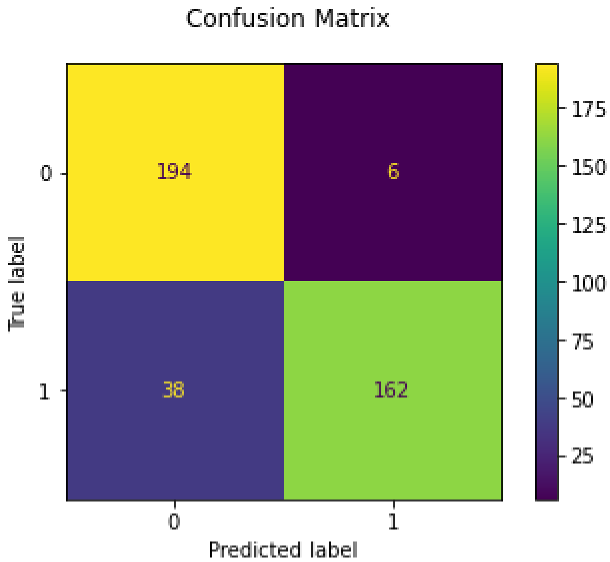


Figure 14. Results from training the KNN Classifier (confusion matrix)

SVM Classifier

The optimal hyperparameters for the model, gamma, was found to be `0.001`. The results are as follows:

```
Classification report for classifier SVC(gamma=0.001):
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       200
           1       0.94      0.95      0.95       200

    accuracy                           0.95       400
   macro avg       0.95      0.95      0.95       400
weighted avg       0.95      0.95      0.95       400
```

Table 6. Results from training the SVM Classifier (classification report)

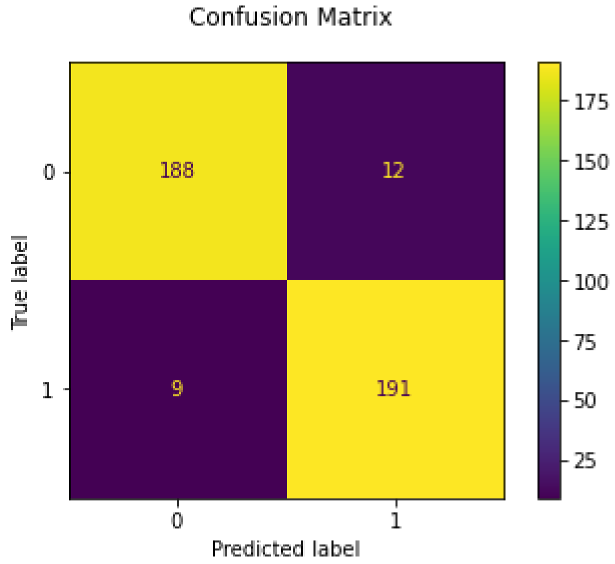


Figure 15. Results from training the SVM Classifier (confusion matrix)

Therefore, it is quite evident that the SVM Classifier would be the best classical model for the task.

3.1.1 Approach

3.1.1.1 Loading and preprocessing an image dataset

Considering the fact that the scikit-learn SVM implementation requires that input images be of the same dimensions, I changed the dimensions of each of the input images to be 224 x 224, exactly like with the ResNet-18 implementation. No other transformations were applied.

3.1.1.1 Training the model

Again, through hyperparameter tuning, it was found that the optimal gamma parameter (defined as `gamma`) was 0.001. Therefore, the SVM was then trained using a gamma of 0.001.

3.1.1 Performance

3.1.1.1 Discussion of evaluation criteria

The evaluation criteria for the SVM is identical to that of the ResNet-18 implementation. To reiterate, the performance of the modal shall be deemed by its ability to correctly predict new and unknown images, obtained from the completely new and unknown validation dataset. The model makes a series of predictions on the validation dataset and the program then evaluates the overall accuracy, outputting a report to the terminal. Ultimately, the validation accuracy was found to be 0.95 (95%).

3.1.1.2 Presentation of results

After training the model, the results of testing outputted to the terminal were given as follows:

```
Classification report for classifier SVC(gamma=0.001):
              precision    recall  f1-score   support

           0       0.95      0.94      0.95       200
           1       0.94      0.95      0.95       200

    accuracy                           0.95       400
   macro avg       0.95      0.95      0.95       400
weighted avg       0.95      0.95      0.95       400
```

Table 6. Results from training the SVM Classifier (classification report)

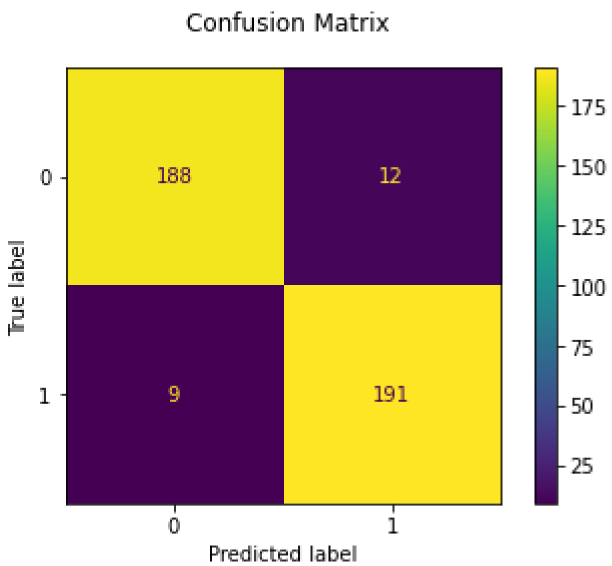


Figure 15. Results from training the SVM Classifier (confusion matrix)

Source Code

Building signage detection and digit extraction

Please find the `building_signage_recogniser.ipynb` *file attached at the end of the report.*

Coral image classification

Alexnet

Please find the `alexnet.ipynb` *file attached at the end of the report.*

VGG16

Please find the `vgg16.ipynb` *file attached at the end of the report.*

ResNet-18

Please find the `resnet18.ipynb` *file attached at the end of the report.*

K-Nearest Neighbours Classifier

Please find the `knn.ipynb` *file attached at the end of the report.*

Support Vector Machine Classifier

Please find the `svm.ipynb` *file attached at the end of the report.*

References

The references utilised in this report pertain only to the source code files. Therefore, these references are provided in the relevant source code files in accordance with *Coding and Academic Integrity*.

Assignment

Task 1. Building Signage Detection and Digit Extraction

```
# Connect to my Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

```
import numpy as np # Numpy provides various useful functions and operators for scie
import cv2 as cv # OpenCV provides various useful functions for computer vision
import os # Optional
import glob # Optional
import imutils
from matplotlib import pyplot as plt
from google.colab.patches import cv2_imshow
```

```
path = "drive/MyDrive/Code/comp3007/assignment/task1/building_signage_recogniser/re
filenames = glob.glob(os.path.join(path, "*.jpg"))
```

```
def read_image(filename):
    return cv.imread(filename)
```

```
def preprocess_image(image_bgr):
    # Convert image from BGR to gray
    image_gray = cv.cvtColor(image_bgr, cv.COLOR_BGR2GRAY)

    # Invert image colours
    image_inverted = cv.bitwise_not(image_gray)

    # Perform blackhat operation
    kernel = cv.getStructuringElement(cv.MORPH_RECT, (17, 17))
    image_blackhat = cv.morphologyEx(image_inverted, cv.MORPH_BLACKHAT, kernel)

    # Perform opening operation
    kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3))
    image_eroded = cv.erode(image_blackhat, kernel, iterations=1)
    kernel = cv.getStructuringElement(cv.MORPH_RECT, (6, 6))
    image_dilated = cv.dilate(image_eroded, kernel, iterations=3)

    # Perform closing operation
    kernel = cv.getStructuringElement(cv.MORPH_RECT, (6, 6))
    image_dilated = cv.dilate(image_dilated, kernel, iterations=3)
    kernel = cv.getStructuringElement(cv.MORPH_RECT, (3, 3))
    image_eroded = cv.erode(image_dilated, kernel, iterations=1)

    # Perform theshold operation
```

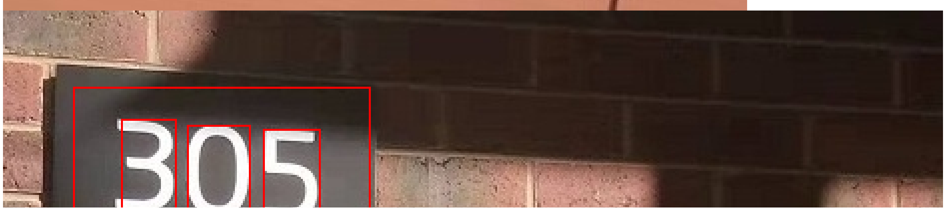
```
    ret, image_threshold = cv.threshold(image_eroded,
                                        150, 255,
                                        cv.THRESH_BINARY)

    return image_threshold, image_blackhat

def extract_sign(image_treshhold):
    # Find and sort contours
    contours = cv.findContours(image_threshold, cv.RETR_TREE, cv.CHAIN_APPROX_NONE)
    contours_sorted = sorted(contours, key = cv.contourArea, reverse = True)

    bounding_box_sign = None
    # If the sign was found
    if contours_sorted:
        x, y, w, h = cv.boundingRect(contours_sorted[0])
        bounding_box_sign = (x, y, w, h)

    return bounding_box_sign


def preprocess_sign(image_cropped):
    # Perform theshold operation on cropped image
    ret, image_threshold = cv.threshold(image_cropped, 40, 255, cv.THRESH_BINARY)

    return image_threshold


def extract_digits(image_threshold):
    # Find and sort contours
    contours = cv.findContours(image_threshold, cv.RETR_EXTERNAL, cv.CHAIN_APPROX_N
    contours_sorted = sorted(contours, key = cv.contourArea, reverse = True)

    # Store minimum digit bounding box aspect ratio,
    # based on manual analysis of digit aspect ratios
    aspect_ratio_min = 0.35
    aspect_ratio_max = 0.8

    bounding_boxes = list()
    # Loop over digits
    for i in range(min(len(contours_sorted), 3)):
        bounding_box = contours_sorted[i]
        # Create digit bounding box
        x, y, w, h = cv.boundingRect(bounding_box)
        aspect_ratio = float(w) / h

        # If bounding box aspect ratio is within specification
        if aspect_ratio_min <= aspect_ratio <= aspect_ratio_max:
            # Append bounding box to list of bounding boxes
            bounding_boxes.append((x, y, w, h))

    return bounding_boxes


for filename in filenames:
    # Read full image
    image_bgr = read_image(filename)
```

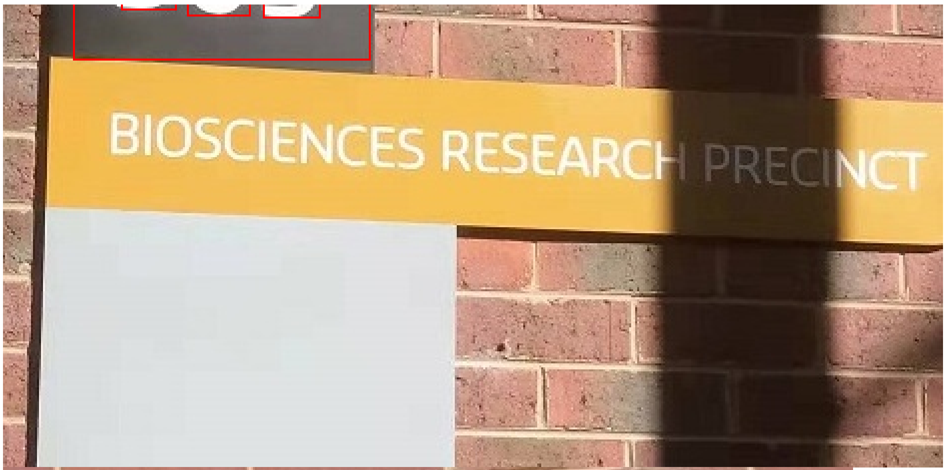
```
# Preprocess full image
image_threshold, image_blackhat = preprocess_image(image_bgr)

# Extract sign
bounding_box_sign = extract_sign(image_threshold)

# If a sign was detected
if bounding_box_sign:
    x = bounding_box_sign[0]
    y = bounding_box_sign[1]
    w = bounding_box_sign[2]
    h = bounding_box_sign[3]

    # Calculate factor to resize bounding box
    dx = w // 10
    dy = h // 10

    # Calculate tight bounding box crop points
    x_start = max(0, x + dx)
    x_end = min(len(image_blackhat[0]), x + w - dx)
    y_start = max(0, y + dy)
    y_end = min(len(image_blackhat), y + h - dy)

    # Crop full image
    image_cropped = image_blackhat[y_start : y_end, x_start : x_end]

    # Preprocess cropped image
    image_threshold = preprocess_sign(image_cropped)

    # Extract digits
    bounding_boxes_digits = extract_digits(image_threshold)

    # Copy full image
    image_bgr_copy = image_bgr.copy()

    # Annotate loose sign bounding box
    cv.rectangle(image_bgr_copy,
                 (x - dx, y - dy),
                 (x + w + dx, y + h + dy),
                 (0, 0, 255),
                 1)

    # For each digit bounding box
    for bounding_box_digit in bounding_boxes_digits:
        x = bounding_box_digit[0]
        y = bounding_box_digit[1]
        w = bounding_box_digit[2]
        h = bounding_box_digit[3]

        # Annotate digit bounding box
        cv.rectangle(image_bgr_copy,
                     (x + x_start, y + y_start),
                     (x + x_start + w, y + y_start + h),
                     (0, 0, 255),
```

```
                        1)

# Show annotated image
cv2_imshow(image_bgr_copy)
```

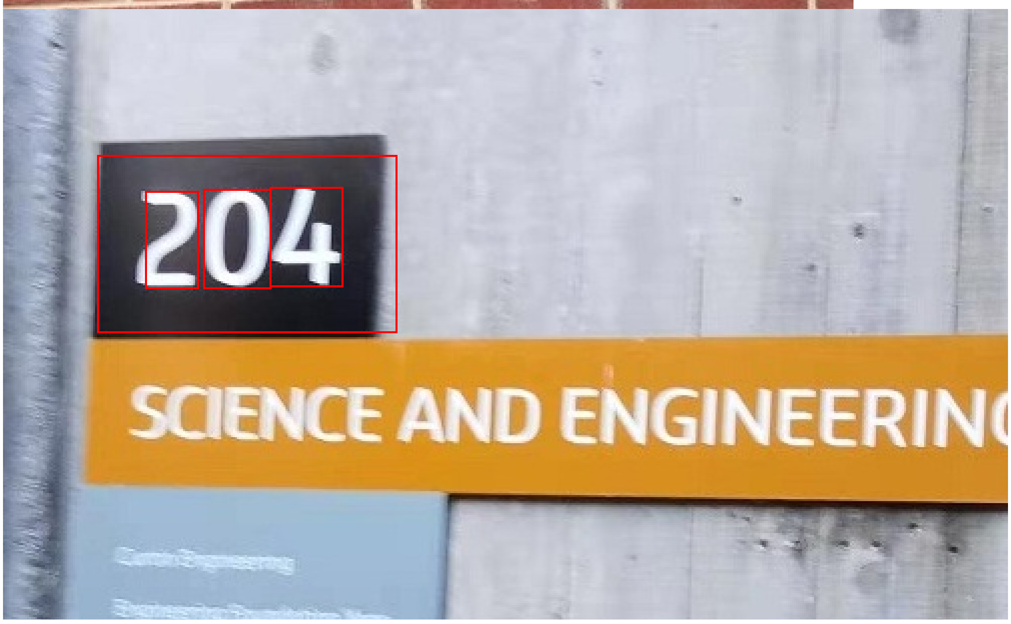

303
VIN DAVIES BUILD
Abacus Labs
Classrooms
Department of Environment
and Agriculture
Centre for Crop and Disease Management
Centre for Digital Agriculture

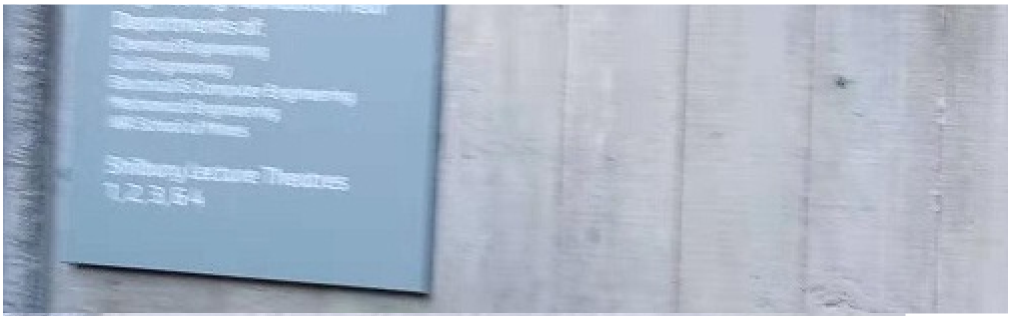

303
VIN DAVIES BUILDIN
Abacus Labs
Classrooms
Department of Environment
and Agriculture
Centre for Crop and Disease Management
Centre for Digital Agriculture

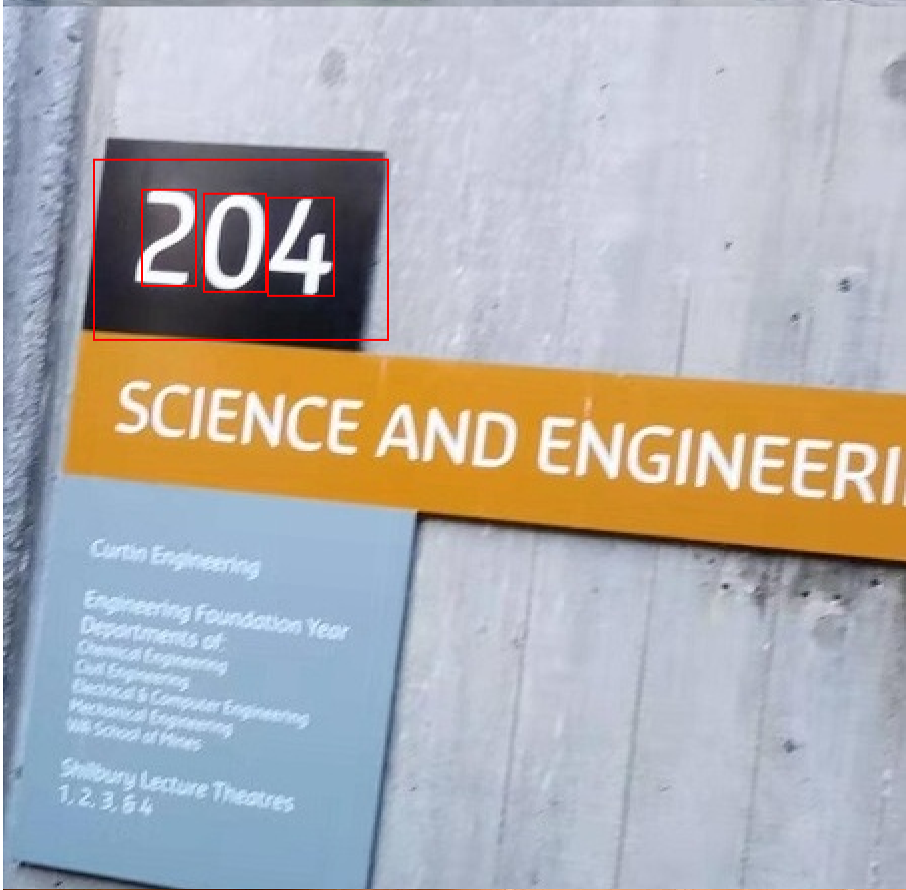

305


BIOSCIENCES RESEARCH PRECINCT


207
SPATIAL SCIENCES
Department of Spatial Sciences
Western Australian Centre for Geodesy
Global Navigation Satellite Systems Research Centre
Geographic Information and Remote Sensing Research
Photogrammetry and Laser Scanning Research
Department of Electrical and Computer Engineering


204
SCIENCE AND

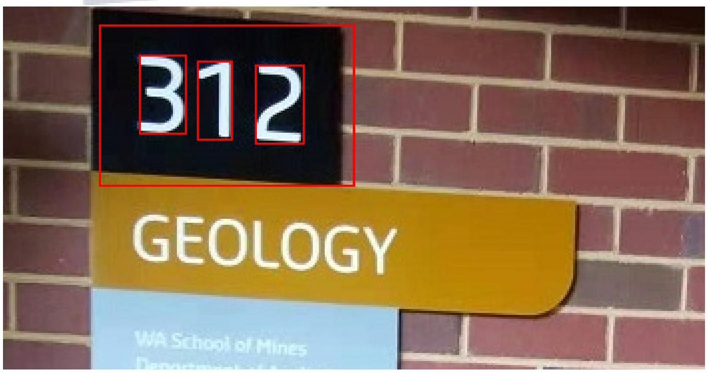


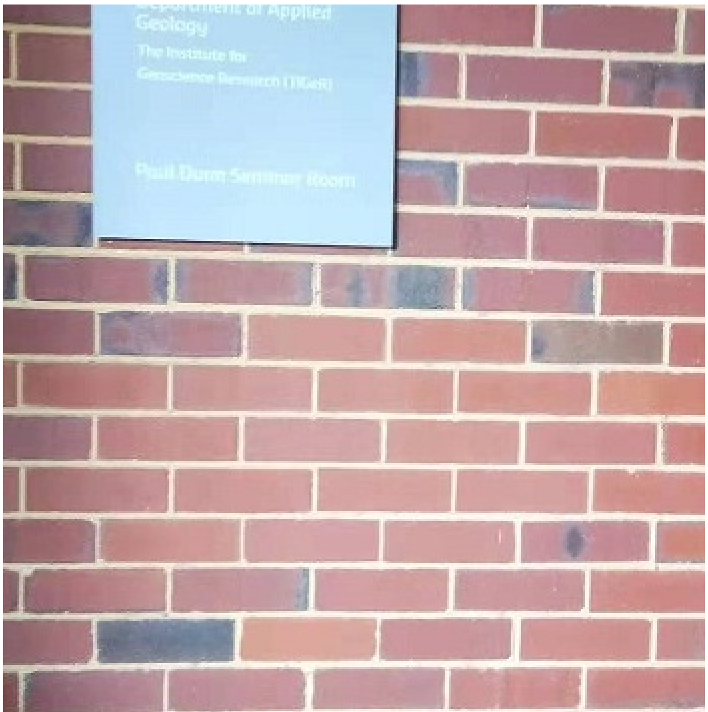

204
SCIENCE AND ENGINEERI
Curtin Engineering
Engineering Foundation Year
Departments of:
Chemical Engineering
Civil Engineering
Electrical & Computer Engineering
Mechanical Engineering
WA School of Mines
Shilbury Lecture Theatres
1, 2, 3, & 4

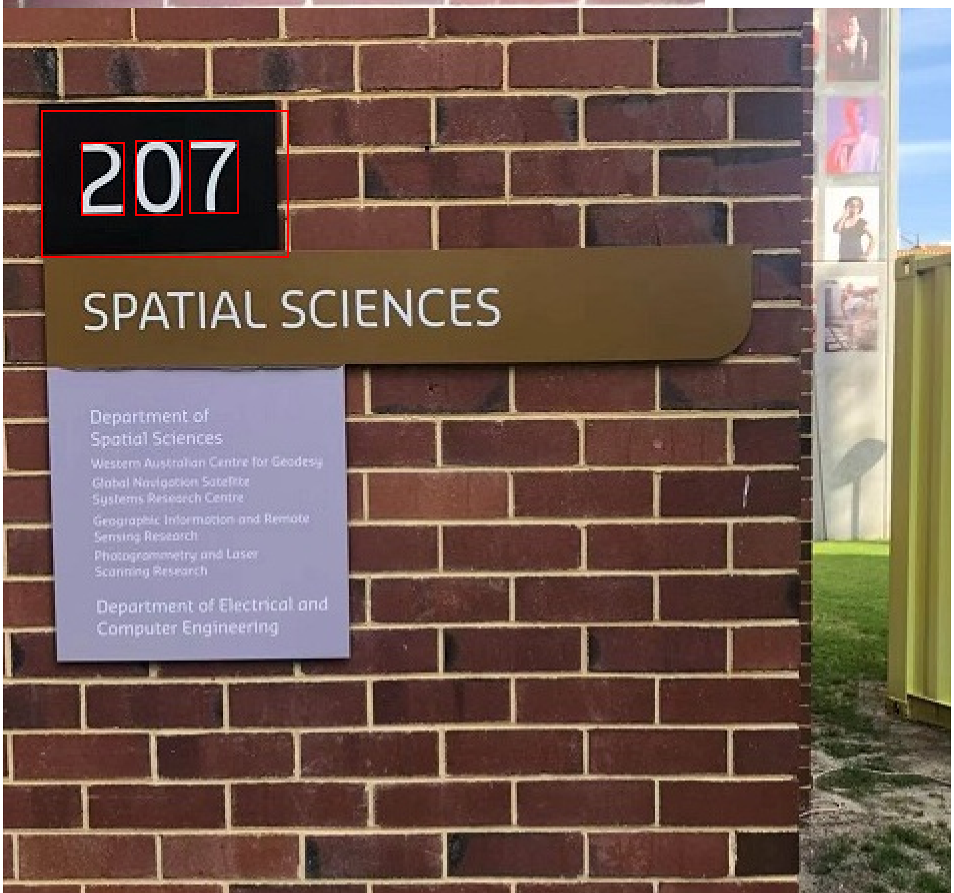

109
COUNSELLING AND HEALTH SERVICES
Health Services
Psychological
and Counselling Service
AccessAbility Services
Properties, Facilities
& Development


301
PHYSICS
Medical Imaging Science
Centre for Marine Science and Technology
Australian Research Council Centre of Excellence for Antimatter - Matter Studies
John de Laeter Centre for Isotope Research

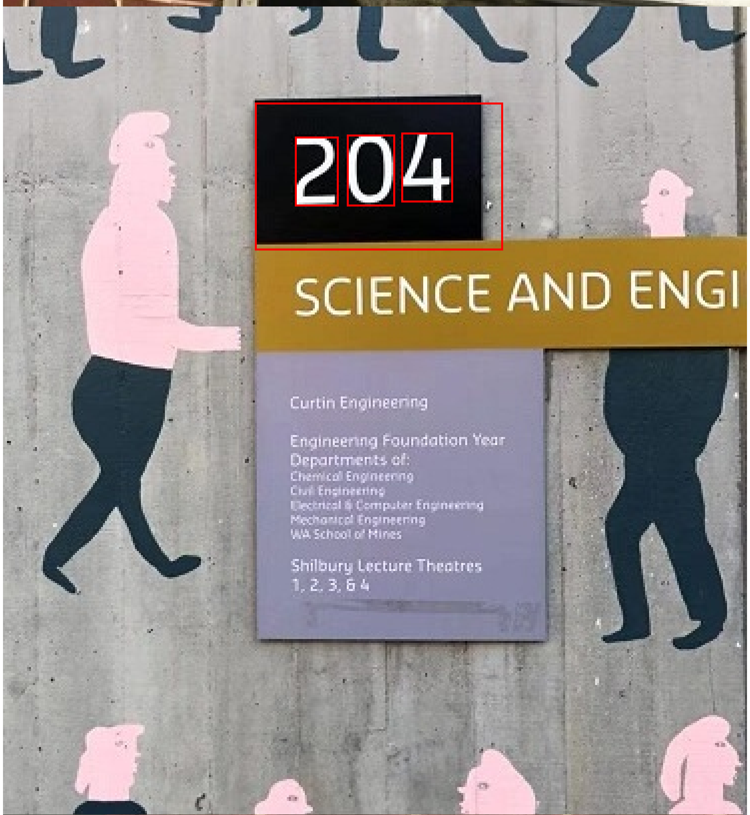

311
ENVIRONMENT AN
LEVEL 1
Department of Environment and Agriculture
Reception and Administration Office
Head of Department Office

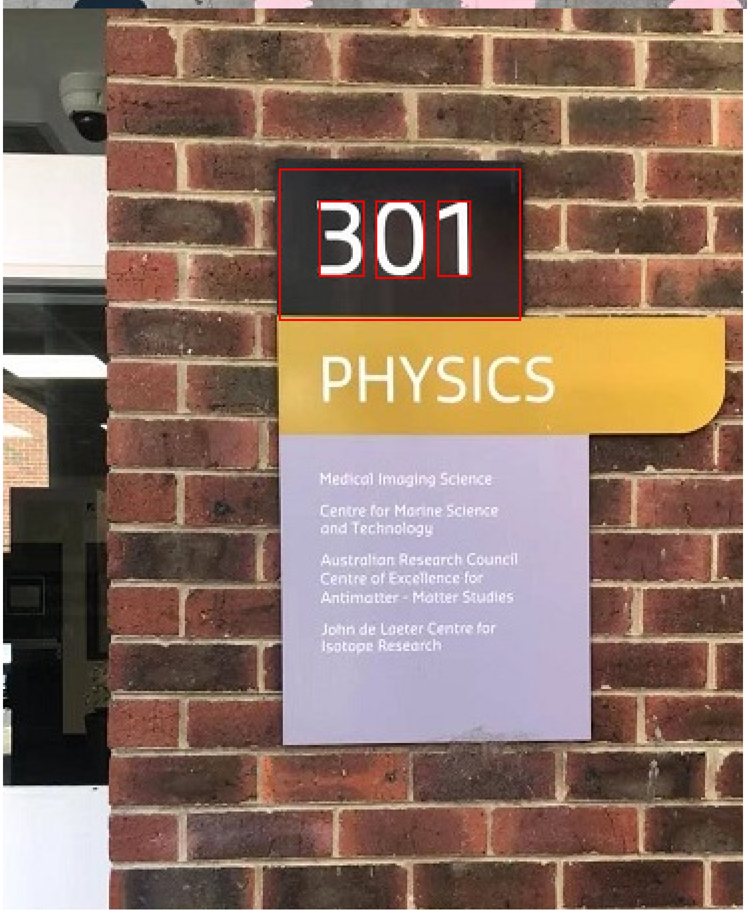

312
GEOLOGY
WA School of Mines

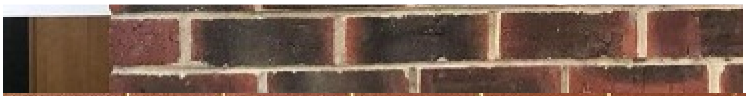

Geology
The Institute for
Geoscience Research (TIGeR)
Paul Dunn Seminar Room


207
SPATIAL SCIENCES
Department of
Spatial Sciences
Western Australian Centre for Geodesy
Global Navigation Satellite
Systems Research Centre
Geographic Information and Remote
Sensing Research
Photogrammetry and Laser
Scanning Research
Department of Electrical and
Computer Engineering


209
HUMANITIES

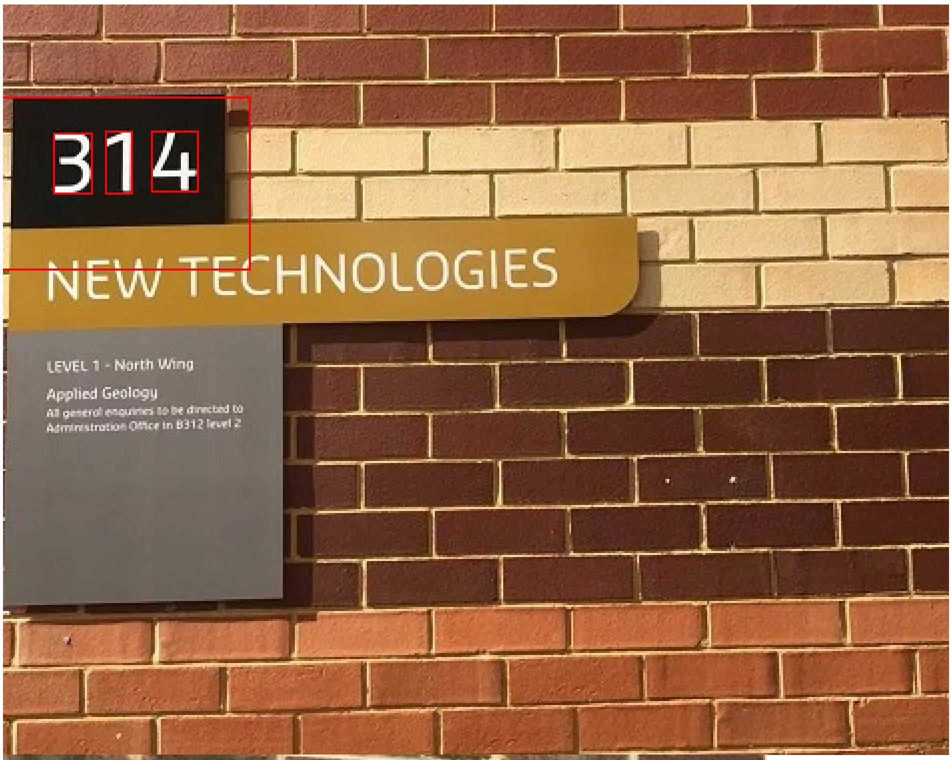

204
SCIENCE AND ENGI
Curtin Engineering
Engineering Foundation Year
Departments of:
Chemical Engineering
Civil Engineering
Electrical & Computer Engineering
Mechanical Engineering
WA School of Mines
Shilbury Lecture Theatres
1, 2, 3, & 4

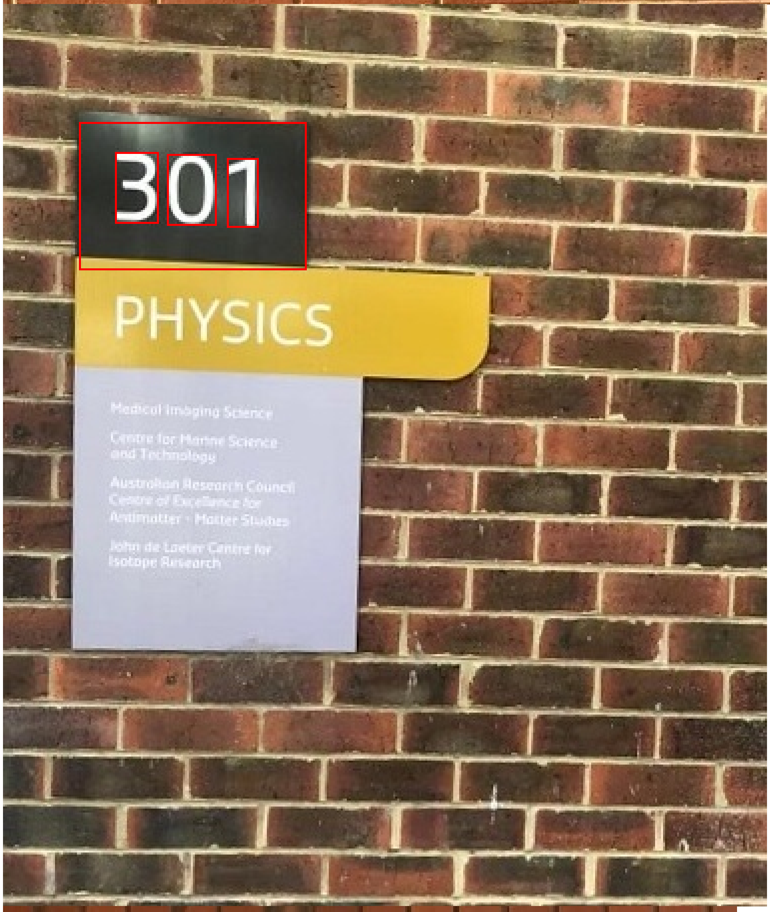

301
PHYSICS
Medical Imaging Science
Centre for Marine Science
and Technology
Australian Research Council
Centre of Excellence for
Antimatter - Matter Studies
John de Laeter Centre for
Isotope Research




314
NEW TECHNOLOGIES
LEVEL 1 - North Wing
Applied Geology
All general enquiries to be directed to
Administration Office in 8312 level 2

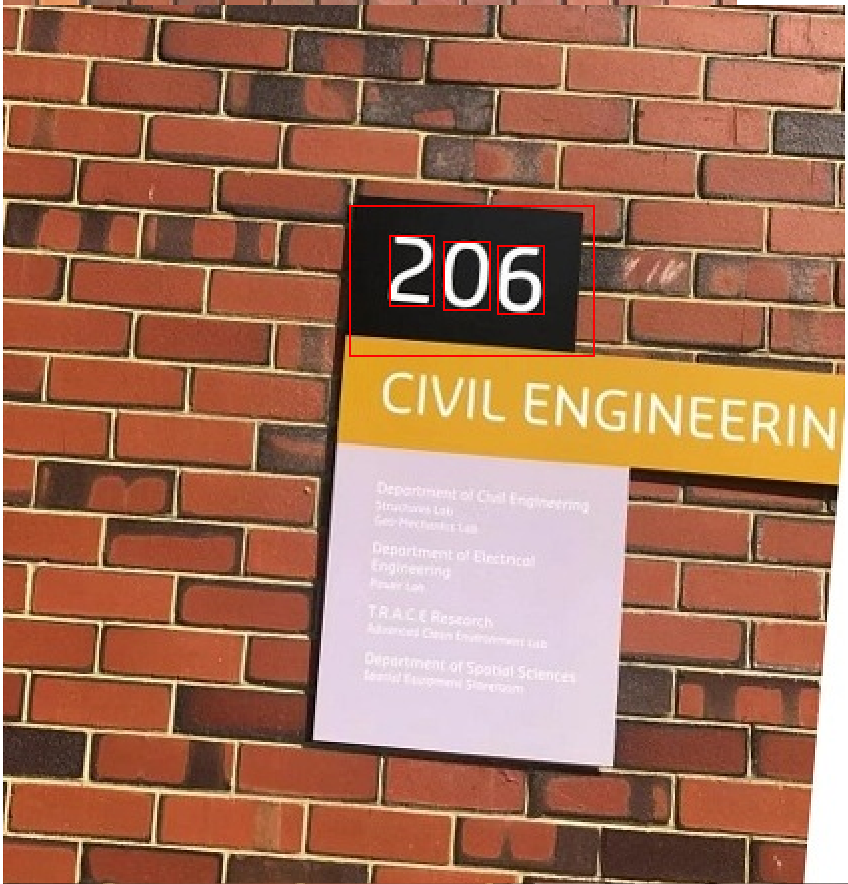

202
DESIGN AND ART
Art
Design
Communication Design
COUNCILLORS

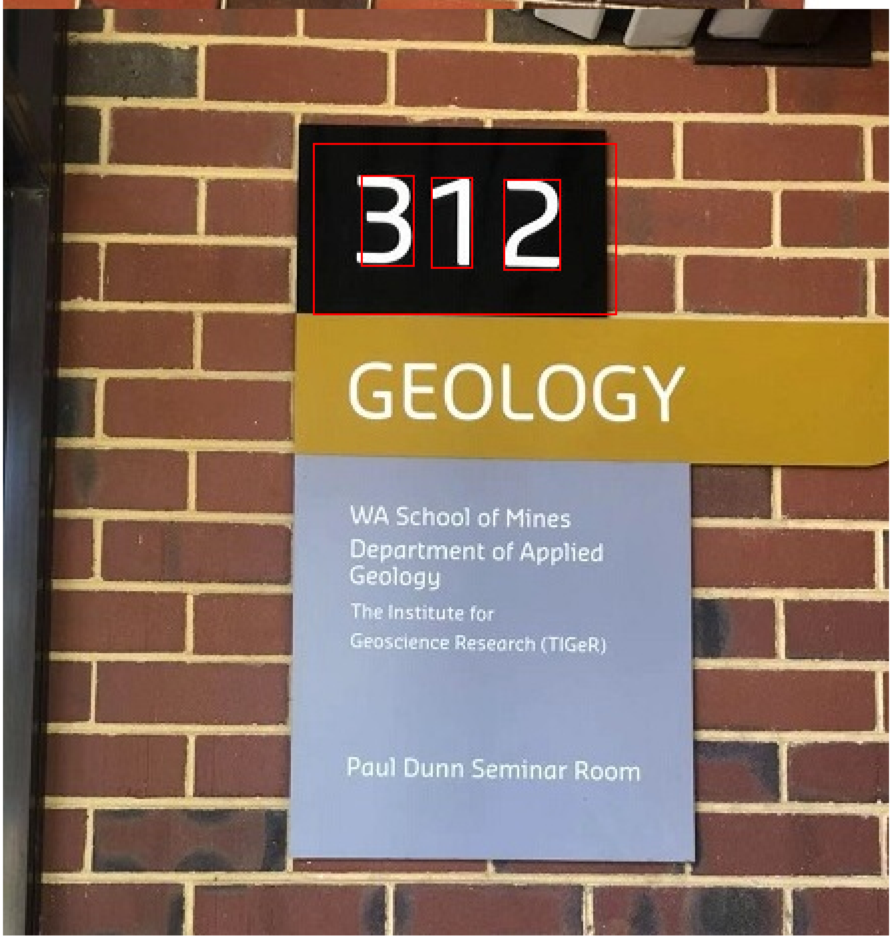

314
NEW TECHNOLOGIES
LEVEL 1 - North Wing
Applied Geology
All general enquiries to be directed to Administration Office in B312 level 2

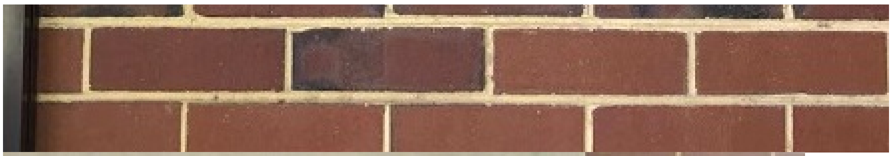

301
PHYSICS
Medical Imaging Science
Centre for Marine Science and Technology
Australian Research Council Centre of Excellence for Antimatter - Matter Studies
John de Laeter Centre for Isotope Research


109
COUNSELLING AND
Health Services
Psychological and Counselling Service


AccessAbility Services
Properties, Facilities
& Development

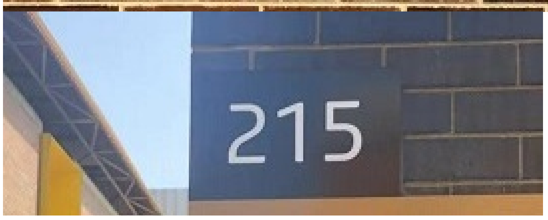

206
CIVIL ENGINEERIN
Department of Civil Engineering
Department of Electrical
Engineering
T.R.A.C.E Research
Department of Spatial Sciences

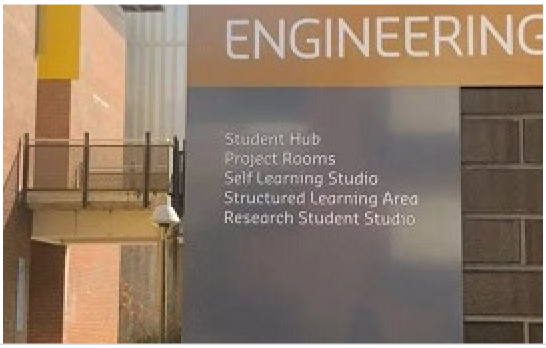

312
GEOLOGY
WA School of Mines
Department of Applied
Geology
The Institute for
Geoscience Research (TIGeR)
Paul Dunn Seminar Room




209
HUMANITIES
Faculty of Humanities
Pro Vice Chancellor's Office
Humanities & Science & Engineering
Student Service Office
Teaching and Learning Office


207
SPATIAL SCIENCES
Department of
Spatial Sciences
Western Australian Centre for Geodesy
Global Navigation Satellite
Systems Research Centre
Geographic Information and Remote
Sensing Research
Photogrammetry and Laser
Scanning Research
Department of Electrical and
Computer Engineering


215


ENGINEERING
Student Hub
Project Rooms
Self Learning Studio
Structured Learning Area
Research Student Studio

Assignment

Task 2. Coral Image Classification

```
"""
Adapted from Assignment (COMP3010)
    Tanaka Chitete
    Accessed 07/10/2022

Adapted from Image Data Loaders in PyTorch
    Shivam Chandhok
    Accessed 07/10/2022
""";
```

```
# Connect to my Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

```
!pip install matplotlib-inline
!pip install d2l==0.17.5
```

```
import torch
from torch import nn
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import torchvision.models as models
from torchvision import datasets, transforms as T
from d2l import torch as d2l
```

```
if torch.cuda.is_available():
  torch.cuda.empty_cache()
  device = torch.device('cuda')
else:
  device = torch.device('cpu')

alexnet = models.alexnet(pretrained=True).to(device)
```

```
torch.manual_seed(0)
```

Reading the Dataset

```
normalize = T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
transforms = T.Compose([T.Resize((224, 224)), T.ToTensor(), normalize])
```

```
batch_size = 256

train_path = "drive/MyDrive/Code/comp3007/assignment/task2/alexnet/resources/train"
train_data = ImageFolder(root=train_path, transform=transforms)

val_path = "drive/MyDrive/Code/comp3007/assignment/task2/alexnet/resources/val"
val_data = ImageFolder(root=val_path, transform=transforms)

# Create training and validation set dataloaders
train_data_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True)
val_data_loader = DataLoader(val_data, batch_size=batch_size, shuffle=True)
```

Configuring Output Dimensions

```
def init_weights(model):
  nn.init.normal_(model.weight, std=0.01)

out_layer = nn.Linear(4096, 2)
out_layer.apply(init_weights)


alexnet.classifier[6] = out_layer
```

Training

```
"""
Adapted from Dive into Deep Learning.
    Aston Zhang, Zack C. Lipton, Mu Li and Alex J. Smola.
    https://d2l.ai/
    Accessed 12/05/2022
"""

def train(net, train_iter, num_epochs, lr, device):
    print('training on', device)
    net.to(device)
    optimizer = torch.optim.SGD(net.parameters(), lr=lr)
    loss = nn.CrossEntropyLoss()
    timer, num_batches = d2l.Timer(), len(train_iter)
    for epoch in range(num_epochs):
        # Sum of training loss, sum of training accuracy, no. of examples
        metric = d2l.Accumulator(3)
        net.train()
        for i, (X, y) in enumerate(train_iter):
            timer.start()
            optimizer.zero_grad()
            X, y = X.to(device), y.to(device)
            y_hat = net(X)
            l = loss(y_hat, y)
```

```
            l.backward()
            optimizer.step()
            with torch.no_grad():
                metric.add(l * X.shape[0], d2l.accuracy(y_hat, y), X.shape[0])
            timer.stop()
            train_l = metric[0] / metric[2]
            train_acc = metric[1] / metric[2]
        print(f'epoch {epoch + 1}, loss {train_l:.3f}, train acc {train_acc:.3f}')
    print(f'{metric[2] * num_epochs / timer.sum():.1f} examples/sec '
          f'on {str(device)}')

learning_rate, num_epochs = 0.01, 10
train(alexnet, train_data_loader, num_epochs, learning_rate, d2l.try_gpu())
```

```
training on cuda:0
epoch 1, loss 1.167, train acc 0.487
epoch 2, loss 0.820, train acc 0.604
epoch 3, loss 0.578, train acc 0.791
epoch 4, loss 0.633, train acc 0.703
epoch 5, loss 0.405, train acc 0.854
epoch 6, loss 0.678, train acc 0.661
epoch 7, loss 0.401, train acc 0.828
epoch 8, loss 0.443, train acc 0.819
epoch 9, loss 0.264, train acc 0.890
epoch 10, loss 0.337, train acc 0.865
704.5 examples/sec on cuda:0
```

Evaluating

```
"""
Adapted from Dive into Deep Learning.
    Aston Zhang, Zack C. Lipton, Mu Li and Alex J. Smola.
    https://d2l.ai/
    Accessed 12/05/2022
"""

def evaluate(net, data_iter, device=None):
    if isinstance(net, nn.Module):
        net.eval()  # Set the model to evaluation mode
        if not device:
            device = next(iter(net.parameters())).device
    # No. of correct predictions, no. of predictions
    metric = d2l.Accumulator(2)

    with torch.no_grad():
        for X, y in data_iter:
            if isinstance(X, list):
                # Required for BERT Fine-tuning
                X = [x.to(device) for x in X]
            else:
                X = X.to(device)
            y = y.to(device)
            metric.add(d2l.accuracy(net(X), y), y.numel())
    test_acc = metric[0] / metric[1]
```

```
    test_acc = metric[0] / metric[1]
    print(f'test acc {test_acc:.3f}')


evaluate(alexnet, val_data_loader)
```

```
test acc 0.955
```

Assignment

Task 2. Coral Image Classification

```
"""
Adapted from Assignment (COMP3010)
    Tanaka Chitete
    Accessed 07/10/2022

Adapted from Image Data Loaders in PyTorch
    Shivam Chandhok
    Accessed 07/10/2022
""";
```

```
# Connect to my Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

```
!pip install matplotlib-inline
!pip install d2l==0.17.5
```

```
import torch
from torch import nn
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import torchvision.models as models
from torchvision import datasets, transforms as T
from d2l import torch as d2l
```

```
if torch.cuda.is_available():
  torch.cuda.empty_cache()
  device = torch.device('cuda')
else:
  device = torch.device('cpu')

vgg16 = models.vgg16(pretrained=True).to(device)
```

```
torch.manual_seed(0)
```

Reading the Dataset

```
normalize = T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
transforms = T.Compose([T.Resize((224, 224)), T.ToTensor(), normalize])
```

```
batch_size = 128

train_path = "drive/MyDrive/Code/comp3007/assignment/task2/vgg16/resources/train"
train_data = ImageFolder(root=train_path, transform=transforms)

val_path = "drive/MyDrive/Code/comp3007/assignment/task2/vgg16/resources/val"
val_data = ImageFolder(root=val_path, transform=transforms)

# Create training and validation set dataloaders
train_data_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True)
val_data_loader = DataLoader(val_data, batch_size=batch_size, shuffle=True)
```

Configuring Output Dimensions

```
def init_weights(model):
  nn.init.normal_(model.weight, std=0.01)

out_layer = nn.Linear(4096, 2)
out_layer.apply(init_weights)


vgg16.classifier[6] = out_layer
```

Training

```
"""
Adapted from Dive into Deep Learning.
    Aston Zhang, Zack C. Lipton, Mu Li and Alex J. Smola.
    https://d2l.ai/
    Accessed 12/05/2022
"""

def train(net, train_iter, num_epochs, lr, device):
    print('training on', device)
    net.to(device)
    optimizer = torch.optim.SGD(net.parameters(), lr=lr)
    loss = nn.CrossEntropyLoss()
    timer, num_batches = d2l.Timer(), len(train_iter)
    for epoch in range(num_epochs):
        # Sum of training loss, sum of training accuracy, no. of examples
        metric = d2l.Accumulator(3)
        net.train()
        for i, (X, y) in enumerate(train_iter):
            timer.start()
            optimizer.zero_grad()
            X, y = X.to(device), y.to(device)
            y_hat = net(X)
            l = loss(y_hat, y)
            if not torch.isnan(l):
                l.backward()
```

```
            l.backward()
            optimizer.step()
        with torch.no_grad():
            metric.add(l * X.shape[0], d2l.accuracy(y_hat, y), X.shape[0])
        timer.stop()
        train_l = metric[0] / metric[2]
        train_acc = metric[1] / metric[2]
    print(f'epoch {epoch + 1}, loss {train_l:.3f}, train acc {train_acc:.3f}')
print(f'{metric[2] * num_epochs / timer.sum():.1f} examples/sec '
      f'on {str(device)}')
```

```
learning_rate, num_epochs = 0.05, 10
train(vgg16, train_data_loader, num_epochs, learning_rate, d2l.try_gpu())
```

```
training on cuda:0
epoch 1, loss 0.621, train acc 0.686
epoch 2, loss 0.758, train acc 0.607
epoch 3, loss 0.596, train acc 0.677
epoch 4, loss 0.548, train acc 0.741
epoch 5, loss 0.427, train acc 0.809
epoch 6, loss 0.302, train acc 0.875
epoch 7, loss 0.227, train acc 0.919
epoch 8, loss 0.231, train acc 0.904
epoch 9, loss 0.078, train acc 0.978
epoch 10, loss 0.135, train acc 0.951
553.6 examples/sec on cuda:0
```

Evaluating

```
"""
Adapted from Dive into Deep Learning.
    Aston Zhang, Zack C. Lipton, Mu Li and Alex J. Smola.
    https://d2l.ai/
    Accessed 12/05/2022
"""

def evaluate(net, data_iter, device=None):
    if isinstance(net, nn.Module):
        net.eval()  # Set the model to evaluation mode
        if not device:
            device = next(iter(net.parameters())).device
    # No. of correct predictions, no. of predictions
    metric = d2l.Accumulator(2)

    with torch.no_grad():
        for X, y in data_iter:
            if isinstance(X, list):
                # Required for BERT Fine-tuning
                X = [x.to(device) for x in X]
            else:
                X = X.to(device)
            y = y.to(device)
            metric.add(d2l.accuracy(net(X), y), y.numel())
```

```
    test_acc = metric[0] / metric[1]
    print(f'test acc {test_acc:.3f}')
```

```
evaluate(vgg16, val_data_loader)
```

```
test acc 0.968
```

Assignment

Task 2. Coral Image Classification

```
"""
Adapted from Assignment (COMP3010)
    Tanaka Chitete
    Accessed 07/10/2022

Adapted from Image Data Loaders in PyTorch
    Shivam Chandhok
    Accessed 07/10/2022
""";
```

```
# Connect to my Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

```
!pip install matplotlib-inline
!pip install d2l==0.17.5
```

```
import torch
from torch import nn
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import torchvision.models as models
from torchvision import datasets, transforms as T
from d2l import torch as d2l
```

```
if torch.cuda.is_available():
  torch.cuda.empty_cache()
  device = torch.device('cuda')
else:
  device = torch.device('cpu')

resnet18 = models.resnet18(pretrained=True).to(device)
```

```
torch.manual_seed(0)
```

Reading the Dataset

```
normalize = T.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
transforms = T.Compose([T.Resize((224, 224)), T.ToTensor(), normalize])
```

```
batch_size = 256

train_path = "drive/MyDrive/Code/comp3007/assignment/task2/resnet18/resources/train
train_data = ImageFolder(root=train_path, transform=transforms)

val_path = "drive/MyDrive/Code/comp3007/assignment/task2/resnet18/resources/val"
val_data = ImageFolder(root=val_path, transform=transforms)

# Create training and validation set dataloaders
train_data_loader = DataLoader(train_data, batch_size=batch_size, shuffle=True)
val_data_loader = DataLoader(val_data, batch_size=batch_size, shuffle=True)
```

Configuring Output Dimensions

```
def init_weights(model):
  nn.init.normal_(model.weight, std=0.01)

out_layer = nn.Linear(512, 2)
out_layer.apply(init_weights)
```

```
resnet18.fc = out_layer
```

Training

```
"""
Adapted from Dive into Deep Learning.
    Aston Zhang, Zack C. Lipton, Mu Li and Alex J. Smola.
    https://d2l.ai/
    Accessed 12/05/2022
"""

def train(net, train_iter, num_epochs, lr, device):
    print('training on', device)
    net.to(device)
    optimizer = torch.optim.SGD(net.parameters(), lr=lr)
    loss = nn.CrossEntropyLoss()
    timer, num_batches = d2l.Timer(), len(train_iter)
    for epoch in range(num_epochs):
        # Sum of training loss, sum of training accuracy, no. of examples
        metric = d2l.Accumulator(3)
        net.train()
        for i, (X, y) in enumerate(train_iter):
            timer.start()
            optimizer.zero_grad()
            X, y = X.to(device), y.to(device)
            y_hat = net(X)
            l = loss(y_hat, y)
```

```
            l.backward()
            optimizer.step()
            with torch.no_grad():
                metric.add(l * X.shape[0], d2l.accuracy(y_hat, y), X.shape[0])
            timer.stop()
            train_l = metric[0] / metric[2]
            train_acc = metric[1] / metric[2]
        print(f'epoch {epoch + 1}, loss {train_l:.3f}, train acc {train_acc:.3f}')
    print(f'{metric[2] * num_epochs / timer.sum():.1f} examples/sec '
          f'on {str(device)}')

learning_rate, num_epochs = 0.05, 10
train(resnet18, train_data_loader, num_epochs, learning_rate, d2l.try_gpu())
```

```
training on cuda:0
epoch 1, loss 2.518, train acc 0.615
epoch 2, loss 0.087, train acc 0.970
epoch 3, loss 0.111, train acc 0.962
epoch 4, loss 0.030, train acc 0.995
epoch 5, loss 0.033, train acc 0.991
epoch 6, loss 0.011, train acc 1.000
epoch 7, loss 0.008, train acc 1.000
epoch 8, loss 0.007, train acc 1.000
epoch 9, loss 0.007, train acc 1.000
epoch 10, loss 0.005, train acc 1.000
307.3 examples/sec on cuda:0
```

Evaluating

```
"""
Adapted from Dive into Deep Learning.
    Aston Zhang, Zack C. Lipton, Mu Li and Alex J. Smola.
    https://d2l.ai/
    Accessed 12/05/2022
"""

def evaluate(net, data_iter, device=None):
    if isinstance(net, nn.Module):
        net.eval()  # Set the model to evaluation mode
        if not device:
            device = next(iter(net.parameters())).device
    # No. of correct predictions, no. of predictions
    metric = d2l.Accumulator(2)

    with torch.no_grad():
        for X, y in data_iter:
            if isinstance(X, list):
                # Required for BERT Fine-tuning
                X = [x.to(device) for x in X]
            else:
                X = X.to(device)
            y = y.to(device)
            metric.add(d2l.accuracy(net(X), y), y.numel())
```

```
    test_acc = metric[0] / metric[1]
    print(f'test acc {test_acc:.3f}')
```

```
evaluate(resnet18, val_data_loader)
```

```
test acc 0.995
```

Assignment

Task 2. Coral Image Classification

```
"""
Adapted from Recognizing hand-written digits
    Gael Varoquaux
    Accessed 15/10/2022
""";
```

```
# Connect to my Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

```
%matplotlib inline
```

```
import matplotlib.pyplot as plt
from sklearn import datasets, neighbors, metrics, linear_model
from PIL import Image
from keras.preprocessing.image import ImageDataGenerator
from skimage.transform import resize
import numpy as np
import os # Optional
import glob # Optional
```

```
train_files = datasets.load_files("drive/MyDrive/Code/comp3007/assignment/task2/svm
test_files = datasets.load_files("drive/MyDrive/Code/comp3007/assignment/task2/svm/
```

```
def read_images(filenames):
    n_samples = len(filenames)
    images = list()
    for filename in filenames:
        image = plt.imread(filename)
        image = resize(image, (224, 224, 1), preserve_range=True)
        images.append(image)
    return np.array(images).reshape((n_samples, -1))
```

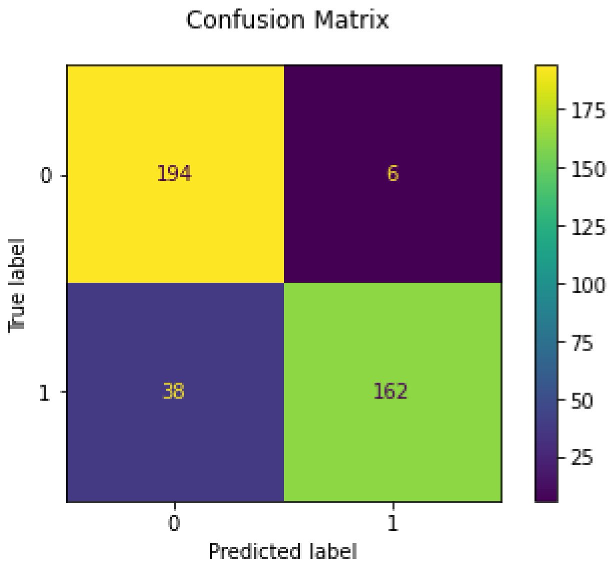
```
X_train = read_images(train_files.filenames)
y_train = train_files.target

X_test = read_images(test_files.filenames)
y_test = test_files.target

# Learn the images on the train subset
classifier = neighbors.KNeighborsClassifier(n_neighbors=5)
```

```
classifier.fit(X_train, y_train)

# Predict the class of the image on the test subset
predicted = classifier.predict(X_test)


print(
    f"Classification report for classifier {classifier}:\n"
    f"{metrics.classification_report(y_test, predicted)}\n"
)
```

```
Classification report for classifier KNeighborsClassifier():
              precision    recall  f1-score   support

           0       0.84      0.97      0.90       200
           1       0.96      0.81      0.88       200

    accuracy                           0.89       400
   macro avg       0.90      0.89      0.89       400
weighted avg       0.90      0.89      0.89       400
```

```
disp = metrics.ConfusionMatrixDisplay.from_predictions(y_test, predicted)
disp.figure_.suptitle("Confusion Matrix")
print(f"Confusion matrix:\n{disp.confusion_matrix}")

plt.show()
```

```
Confusion matrix:
[[194   6]
 [ 38 162]]
```



Assignment

Task 2. Coral Image Classification

```
"""
Adapted from Recognizing hand-written digits
    Gael Varoquaux
    Accessed 15/10/2022
""";
```

```
# Connect to my Google Drive
from google.colab import drive
drive.mount("/content/drive")
```

```
%matplotlib inline
```

```
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics, linear_model
from PIL import Image
from keras.preprocessing.image import ImageDataGenerator
from skimage.transform import resize
import numpy as np
import os # Optional
import glob # Optional
```

```
train_files = datasets.load_files("drive/MyDrive/Code/comp3007/assignment/task2/svm
test_files = datasets.load_files("drive/MyDrive/Code/comp3007/assignment/task2/svm/
```

```
def read_images(filenames):
    n_samples = len(filenames)
    images = list()
    for filename in filenames:
        image = plt.imread(filename)
        image = resize(image, (224, 224, 1), preserve_range=True)
        images.append(image)
    return np.array(images).reshape((n_samples, -1))
```

```
X_train = read_images(train_files.filenames)
y_train = train_files.target

X_test = read_images(test_files.filenames)
y_test = test_files.target

# Learn the images on the train subset
classifier = svm.SVC(gamma=0.001)
```

```
classifier.fit(X_train, y_train)

# Predict the class of the image on the test subset
predicted = classifier.predict(X_test)


print(
    f"Classification report for classifier {classifier}:\n"
    f"{metrics.classification_report(y_test, predicted)}\n"
)
```

```
    Classification report for classifier SVC(gamma=0.001):
                  precision    recall  f1-score   support

               0       0.95      0.94      0.95       200
               1       0.94      0.95      0.95       200

        accuracy                           0.95       400
       macro avg       0.95      0.95      0.95       400
    weighted avg       0.95      0.95      0.95       400
```

```
disp = metrics.ConfusionMatrixDisplay.from_predictions(y_test, predicted)
disp.figure_.suptitle("Confusion Matrix")
print(f"Confusion matrix:\n{disp.confusion_matrix}")

plt.show()
```

```
    Confusion matrix:
    [[188  12]
     [  9 191]]
```