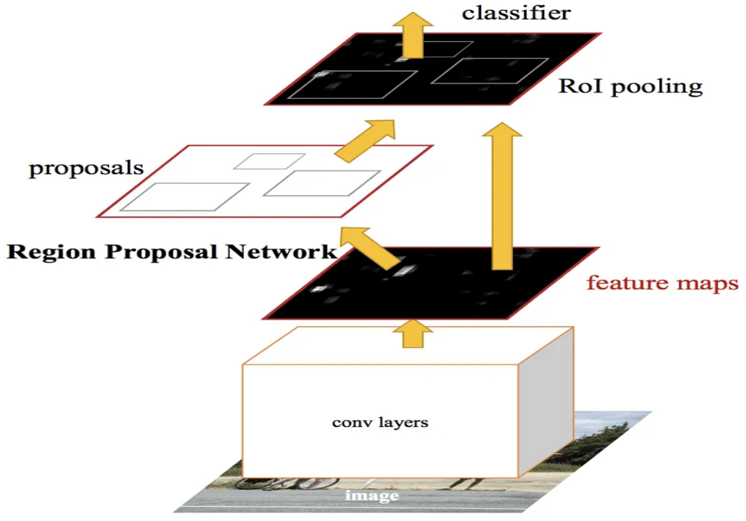
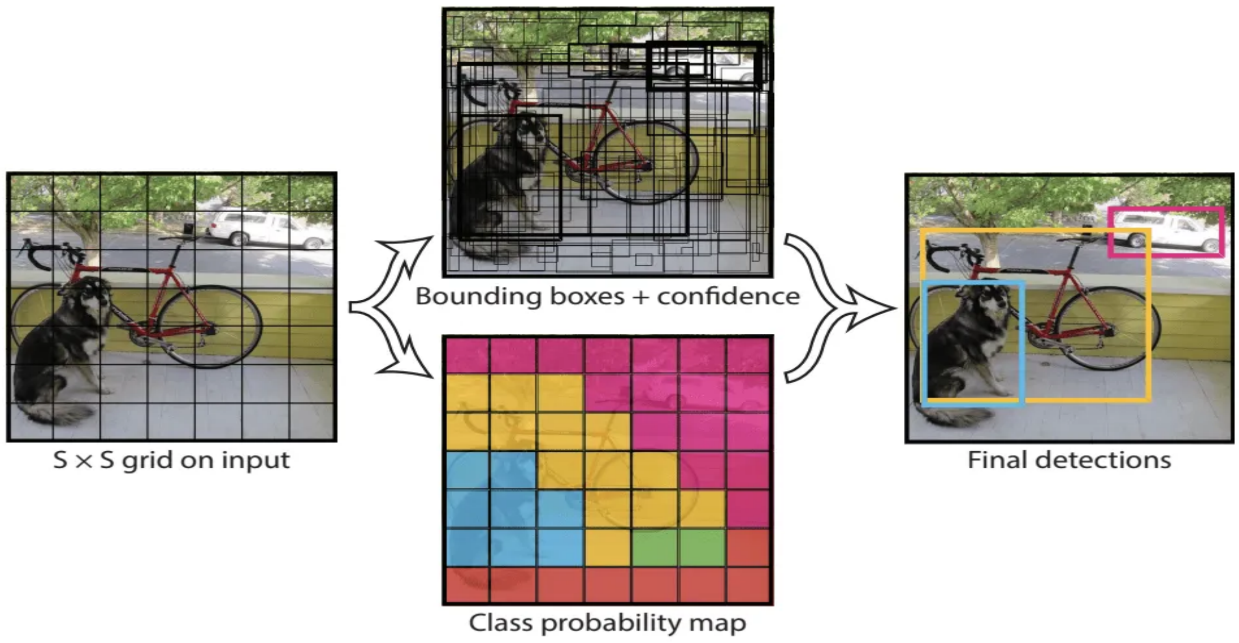
Task 3

Comparison between YOLO and CNN

The best way to utilize CNN is to use Faster R-CNN
Faster R-CNN is a deep convolutional network used for object detection, that appears to the user as a single, end-to-end, unified network. The network can accurately and quickly predict the locations of different objects.

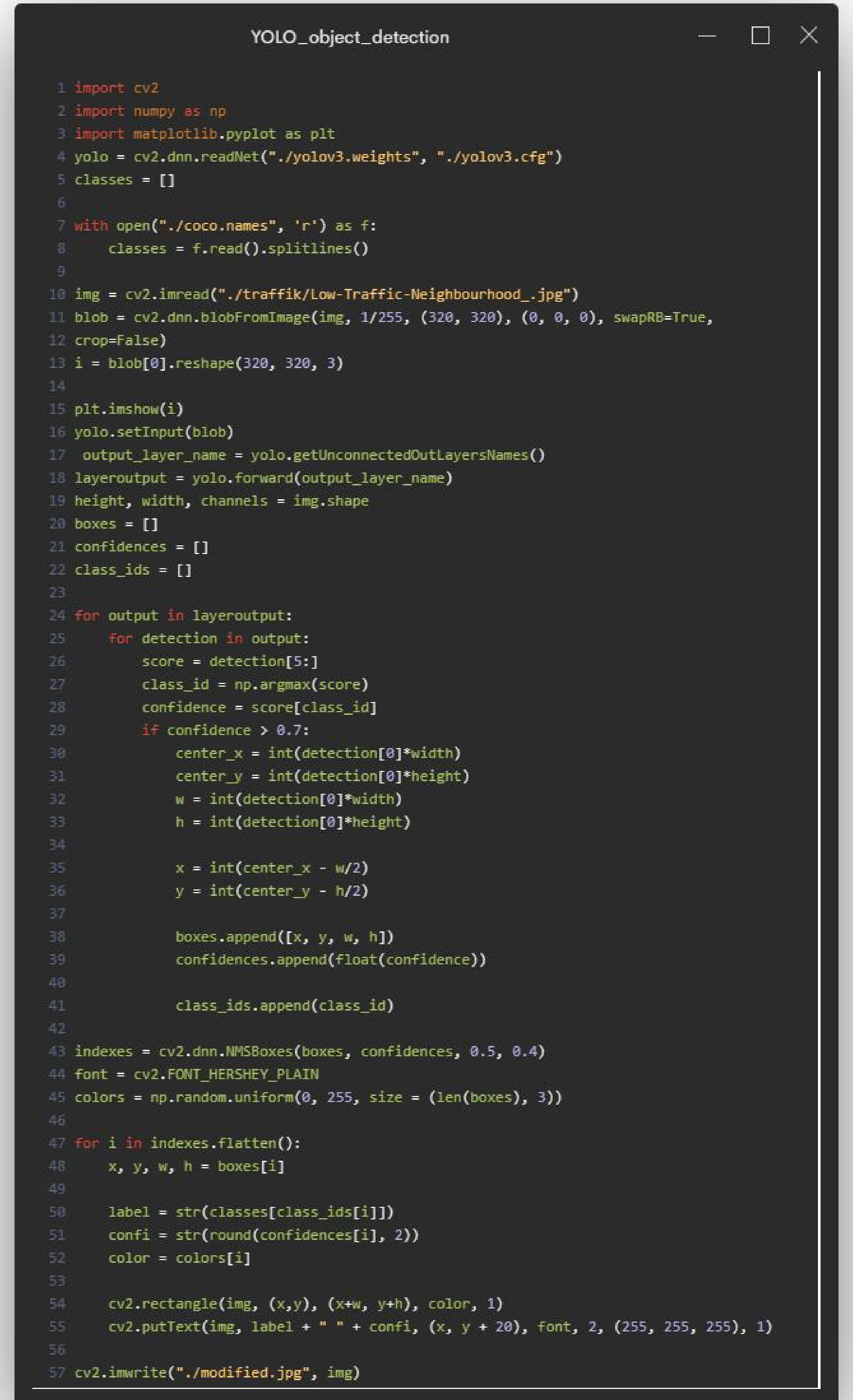


Yolo divides the image into a grid. For each grid, some values like class probabilities and the bounding box parameters are calculated. The model works by first splitting the input image into a grid of cells, where each cell is responsible for predicting a bounding box if the center of a bounding box falls within the cell. Each grid cell predicts a bounding box involving the x, y coordinate and the width and height and the confidence. A class prediction is also based on each cell.



Different Layers in CNN:

- Convolutional (CONV) - consist of a set of K learnable filters (i.e., "kernels"), where each filter has a width and a height, and are nearly always square.
- Activation (ACT or RELU, where we use the same or the actual activation function) - we apply a nonlinear activation function, such as ReLU, ELU, or any of the other Leaky ReLU variants.
- Pooling (POOL) - reduce the spatial size (i.e., width and height) of the input volume. Doing this allows us to reduce the amount of parameters and computation in the network — pooling also helps us control overfitting.
- Fully connected (FC) - Neurons in FC layers are fully connected to all activations in the previous layer, as is the standard for feedforward neural networks. FC layers are always placed at the end of the network
- Dropout - Dropout is actually a form of regularization that aims to help prevent overfitting by increasing testing accuracy, perhaps at the expense of training accuracy


YOLO_object_detection

```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
yolo = cv2.dnn.readNet("./yolov3.weights", "./yolov3.cfg")
classes = []

with open("./coco.names", 'r') as f:
    classes = f.read().splitlines()

img = cv2.imread("./traffik/Low-Traffic-Neighbourhood_.jpg")
blob = cv2.dnn.blobFromImage(img, 1/255, (320, 320), (0, 0, 0), swapRB=True,
crop=False)
i = blob[0].reshape(320, 320, 3)

plt.imshow(i)
yolo.setInput(blob)
 output_layer_name = yolo.getUnconnectedOutLayersNames()
layeroutput = yolo.forward(output_layer_name)
height, width, channels = img.shape
boxes = []
confidences = []
class_ids = []

for output in layeroutput:
    for detection in output:
        score = detection[5:]
        class_id = np.argmax(score)
        confidence = score[class_id]
        if confidence > 0.7:
            center_x = int(detection[0]*width)
            center_y = int(detection[0]*height)
            w = int(detection[0]*width)
            h = int(detection[0]*height)

            x = int(center_x - w/2)
            y = int(center_y - h/2)

            boxes.append([x, y, w, h])
            confidences.append(float(confidence))

            class_ids.append(class_id)

indexes = cv2.dnn.NMSBoxes(boxes, confidences, 0.5, 0.4)
font = cv2.FONT_HERSHEY_PLAIN
colors = np.random.uniform(0, 255, size = (len(boxes), 3))

for i in indexes.flatten():
    x, y, w, h = boxes[i]

    label = str(classes[class_ids[i]])
    confi = str(round(confidences[i], 2))
    color = colors[i]

    cv2.rectangle(img, (x,y), (x+w, y+h), color, 1)
    cv2.putText(img, label + " " + confi, (x, y + 20), font, 2, (255, 255, 255), 1)

cv2.imwrite("./modified.jpg", img)
```