Selected Topics in Visual Recognition in Deep Learning Exercise 3

Student name: Fu-sung Kim-Benjamin Tang

Student ID: 0845058

Github link of Homework 3: https://github.com/kimbold/VRDL_2019/tree/master/HW3 (https://github.com/kimbold/VRDL_2019/tree/master/HW3)

References: The code I used in this work reuses a lot of partly modified code from this repository: https://github.com/penny4860/Yolo-digit-detector (https://github.com/penny4860/Yolo-digit-detector)

Speed benchmark: Processing one image for digit prediction and localization takes 114 ms as tested on Google Colabs TPU which processed the data in general faster than the GPU in my tests:

Introduction:

Methodology:

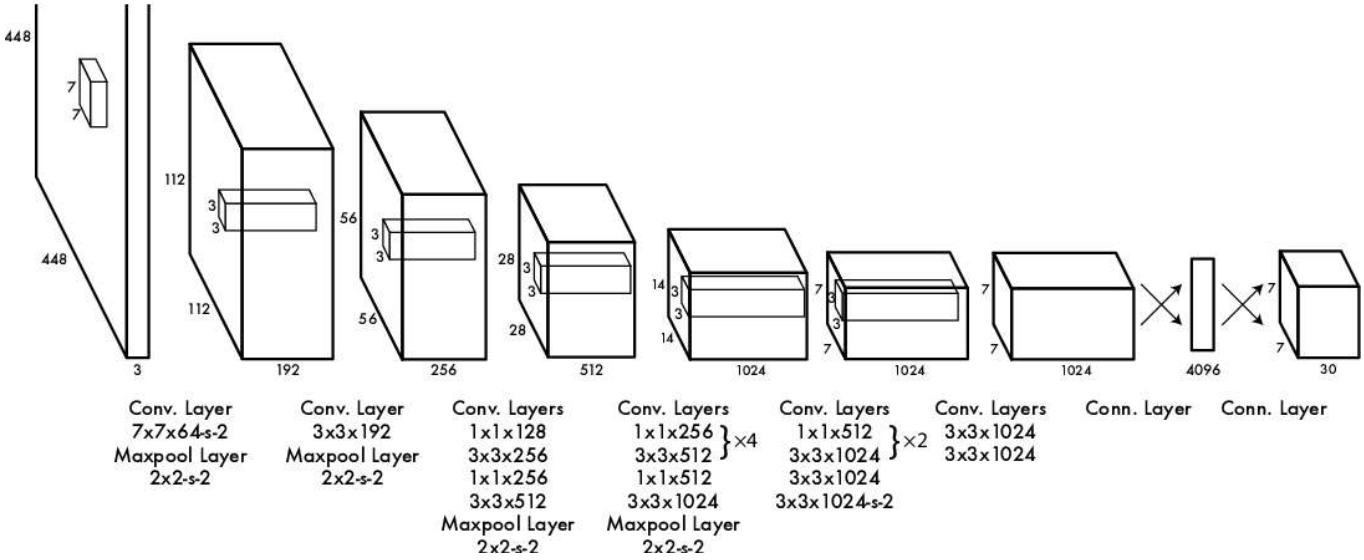
To solve this digit detection exercise, the YOLO v2 network was trained on the provided svhn data and then used to classify the test dataset. For the computation the weights from the training were loaded into google colab and the speed was benchmarked and results stored into the .json file for the submission.

Data preprocess:

The datasets have been loaded into Google Colab and were extracted. Then the .mat/h5 file was processed to create an xml file for each image for the annotations (label and positions). Based on that it was possible to train the network and store the weights for later usage afterwards.

Model architecture:

For this task I utilized the darknet yolo v2 architecture. As described in the yolo9000 (yolo v2) paper here (https://arxiv.org/pdf/1612.08242v1.pdf), the YOLO framework uses a custom network based on the Googlenet architecture. It is a fully connected convolutional neural network and can be visually represented:



(This is a figure from the YOLO paper here (https://www.semanticscholar.org/paper/You-Only-Look-Once%3A-Unified%2C-Real-Time-Object-Redmon-Divvala/f8e79ac0ea341056ef20f2616628b3e964764cfd)

Hyperparameters:

- "architecture": "ResNet50"
- "input_size": 416
- "anchors": [0.57273, 0.677385, 1.87446, 2.06253,3.33843, 5.47434, 7.88282, 3.52778, 9.77052, 9.16828]
- "labels": ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9"]
- "coord_scale" : 1.0
- "class_scale" : 1.0
- "object_scale" : 5.0
- "no_object_scale" : 1.0
- weights: "svhn/weights.h5"
- "actual_epoch": 25,
- "train_times": 5,
- "valid_times": 1,
- "batch_size": 16,
- "learning_rate": 1e-4,
- "saved_folder": "svhn",
- "jitter": true,
- "first_trainable_layer": "input_1",
- "is_only_detect" : false

Summary:

Overall the model performed quite well and managed to detect and a lot of images correctly.

But there are also several instances when the network could not detect any images at all.

In [ ]:

```
!git clone https://github.com/kimbold/VRDL_2019
```

In [0]:

```
######################################################
# Authenticate and create the PyDrive client.
# This only needs to be done once in a notebook.
######################################################

from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials

auth.authenticate_user()
gauth = GoogleAuth()
gauth.credentials = GoogleCredentials.get_application_default()
drive = GoogleDrive(gauth)
```

In [3]:

```
######################################################
# Switch to HW3 folder in Google Colab after
# cloning my github repository
######################################################

%cd VRDL_2019/HW3
```

```
/content/VRDL_2019/HW3
```

In [4]:

```
#####################################################
# Download weights that have been stored after training
#####################################################

!pip install gdown
!gdown https://drive.google.com/uc?id=1DryVeYEC5mlo0YFHltN6EhO4zFuTIvNn
```

```
Requirement already satisfied: gdown in /usr/local/lib/python3.6/dist-pack
ages (3.6.4)
Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-p
ackages (from gdown) (2.21.0)
Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packa
ges (from gdown) (4.28.1)
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packag
es (from gdown) (1.12.0)
Requirement already satisfied: chardet<3.1.0,>=3.0.2 in /usr/local/lib/pyt
hon3.6/dist-packages (from requests->gdown) (3.0.4)
Requirement already satisfied: urllib3<1.25,>=1.21.1 in /usr/local/lib/pyt
hon3.6/dist-packages (from requests->gdown) (1.24.3)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python
3.6/dist-packages (from requests->gdown) (2019.9.11)
Requirement already satisfied: idna<2.9,>=2.5 in /usr/local/lib/python3.6/
dist-packages (from requests->gdown) (2.8)
Downloading...
From: https://drive.google.com/uc?id=1DryVeYEC5mlo0YFHltN6EhO4zFuTIvNn
To: /content/VRDL_2019/HW3/weights.h5
285MB [00:01, 234MB/s]
```

In [5]:

```
#####################################################
# Download test data as specified by HW3
#####################################################

!gdown https://drive.google.com/uc?id=1GV4TLocxFcsR8QPdM-rP8RoOB2vjs_E_
```

```
Downloading...
From: https://drive.google.com/uc?id=1GV4TLocxFcsR8QPdM-rP8RoOB2vjs_E_
To: /content/VRDL_2019/HW3/test.zip
272MB [00:01, 224MB/s]
```

In [ ]:

```
#####################################################
# Install required libraries via pip in Google colab
#####################################################

!pip install tensorflow==1.14.0
!pip install keras==2.1.1
!pip install imgaug==0.2.6
!pip install opencv-python
!pip install Pillow
!pip install requests
!pip install tqdm
!pip install sklearn
!pip install pytest-cov
!pip install codecov
!pip install matplotlib
```

Load svhn dataset

In [0]:

```
######################################################
# If training is enabled, also load training data
# But currently training is disabled
######################################################

training=0
if(training==1):
  !wget http://ufldl.stanford.edu/housenumbers/train.tar.gz
```

Extract dataset files

In [3]:

```
!unzip test.zip
```

Import required modules

In [ ]:

```
import numpy as np
import argparse
import os
import json
from yolo.frontend import create_yolo, get_object_labels
import xml.etree.cElementTree as ET
import h5py
from lxml import etree
import tables
import argparse
import json
import cv2
import numpy as np
from yolo.frontend import create_yolo
from yolo.backend.utils.box import draw_scaled_boxes
from yolo.backend.utils.annotation import parse_annotation
from yolo.backend.utils.eval._box_match import BoxMatcher

import os
import yolo
import os

os.environ["CUDA_DEVICE_ORDER"]="PCI_BUS_ID"
os.environ["CUDA_VISIBLE_DEVICES"]="0"
```

Define methods to load annotations

In [0]:

```
def get_name(index, hdf5_data):
    name = hdf5_data['/digitStruct/name']
    return ''.join([chr(v[0]) for v in hdf5_data[name[index][0]].value])

def keys(f):
    return [key for key in f.keys()]

def get_bbox(index, hdf5_data):
  """
  The box data contains width and height of the box as well as the upper left corner.
  By supplying the distance from the left border and top border, the upper left point c
an be identified.
  """
  attrs = {}
  item = hdf5_data['digitStruct']['bbox'][index].item()
  for key in ['label', 'left', 'top', 'width', 'height']:
      attr = hdf5_data[item][key]
      values = [hdf5_data[attr.value[i].item()].value[0][0]
                for i in range(len(attr))] if len(attr) > 1 else [attr.value[0][0]]
      attrs[key] = values
  return attrs

def prettyPrintXml(xmlFilePathToPrettyPrint):
    parser = etree.XMLParser(resolve_entities=False, strip_cdata=False)
    document = etree.parse(xmlFilePathToPrettyPrint, parser)
    document.write(xmlFilePathToPrettyPrint, pretty_print=True, encoding='utf-8')


# Define replacement strings for different operating systems
WINDOWS_LINE_ENDING = b'\r\n'
UNIX_LINE_ENDING = b'\n'

def replace_Unix_with_Windows_in_XLM(xmlFilePath):
    with open(xmlFilePath, 'rb') as open_file:
      content = open_file.read()

    content = content.replace(UNIX_LINE_ENDING,WINDOWS_LINE_ENDING)

    with open(xmlFilePath, 'wb') as open_file:
        open_file.write(content)
```

In [0]:

```
#######################################################
# If training is enabled, the test annotations have
# to be created to train the model
#######################################################
if(training==1):
  directory = "tests/dataset/svhn/test_anns"
  if not os.path.exists(directory):
      os.makedirs(directory)
```

Load annotations for training

The annotations are read from the h5 file and then written to xml files for each picture.

In [0]:

```
if(training==1):
  # Need to define method to read .mat data and then write annotation files for each pi
cture
  filename = "train/digitStruct.mat"

  f = h5py.File(filename,'r')

  # Iterate through all images to create the annotation file
  for image in range(f['digitStruct']['bbox'].shape[0]):
    #annotation
    root = ET.Element("annotation")

    #Get image annotation data
    size_data = get_bbox(image, f)

    #filename
    filename = get_name(image,f)
    doc = ET.SubElement(root, "filename").text = filename

    # For each detected digit, add the data for the box for it
    object_list = [[] for _ in range(len(size_data['label']))]
    bndbox = [[] for _ in range(len(size_data['label']))]

    for number in range(len(size_data['label'])):
      object_list[number] = ET.SubElement(root, "object")
      ET.SubElement(object_list[number], "name").text = str(int(size_data['label'][numb
er]))

      bndbox[number] = ET.SubElement(object_list[number], "bndbox")
      ET.SubElement(bndbox[number], "xmin").text = str(int(size_data['left'][number]))
      ET.SubElement(bndbox[number], "ymin").text = str(int(size_data['top'][number]))
      ET.SubElement(bndbox[number], "xmax").text = str(int(size_data['width'][number]+s
ize_data['left'][number]))
      ET.SubElement(bndbox[number], "ymax").text = str(int(size_data['height'][number]+
size_data['top'][number]))

    tree = ET.ElementTree(root)

    #Write xml
    tree.write("tests/dataset/svhn/anns/"+str(image+1)+".xml")
    prettyPrintXml("tests/dataset/svhn/anns/"+str(image+1)+".xml")
    replace_Unix_with_Windows_in_XLM("tests/dataset/svhn/anns/"+str(image+1)+".xml")

  # For some reason google colab always showed an error with this checkpoint directory
 after this operation
  # So I had to delete it
  if os.path.exists('tests/dataset/svhn/anns/.ipynb_checkpoints'):
    os.rmdir('tests/dataset/svhn/anns/.ipynb_checkpoints')
```

Train yolo

In [0]:

```
#Train and validate YOLO_v2 model on any dataset
def setup_training(config_file):
    """make directory to save weights & its configuration """
    import shutil
    with open(config_file) as config_buffer:
        config = json.loads(config_buffer.read())
    dirname = config['train']['saved_folder']
    if os.path.isdir(dirname):
        print("{} is already exists. Weight file in directory will be overwritten".form
at(dirname))
    else:
        print("{} is created.".format(dirname, dirname))
        os.makedirs(dirname)
    print("Weight file and Config file will be saved in \"{}\"".format(dirname))
    shutil.copyfile(config_file, os.path.join(dirname, "config.json"))
    return config, os.path.join(dirname, "weights.h5")


def train(conf="configs/from_scratch.json"):
  #path to configuration file

    config, weight_file = setup_training(conf)

    if config['train']['is_only_detect']:
        labels = ["object"]
    else:
        if config['model']['labels']:
            labels = config['model']['labels']
        else:
            labels = get_object_labels(config['train']['train_annot_folder'])
    print(labels)

    # 1. Construct the model
    yolo = create_yolo(config['model']['architecture'],
                       labels,
                       config['model']['input_size'],
                       config['model']['anchors'],
                       config['model']['coord_scale'],
                       config['model']['class_scale'],
                       config['model']['object_scale'],
                       config['model']['no_object_scale'])

    # 2. Load the pretrained weights (if any)
    yolo.load_weights(config['pretrained']['full'], by_name=True)

    # 3. actual training
    yolo.train(config['train']['train_image_folder'],
               config['train']['train_annot_folder'],
               config['train']['actual_epoch'],
               weight_file,
               config["train"]["batch_size"],
               config["train"]["jitter"],
               config['train']['learning_rate'],
               config['train']['train_times'],
               config['train']['valid_times'],
               config['train']['valid_image_folder'],
               config['train']['valid_annot_folder'],
               config['train']['first_trainable_layer'],
```

```
                config['train']['is_only_detect'])
    # loss: 2.1691, train batch jitter=False
```

In [0]:

```
#######################################################
# Here the entire network is trained and it takes
# a long time depending on the configurations
#######################################################

if(training==1):
  train(conf="configs/from_scratch_custom.json")
```

In [0]:

```
#######################################################
# Here only the last layer is fine tuned
#######################################################

if(training==1):
  train(conf="configs/from_scratch2_custom.json")
```

After training the weights are stored in .h5 file

So now it is necessary to ensure that it is closed properly to avoid corruption. Not sure why exactly but it caused errors.

In [0]:

```
# Closing all .hd5 files: https://stackoverflow.com/questions/29863342/close-an-open-h5
py-data-file against corruption
tables.file._open_files.close_all()
```

Evaluate trained yolo digit detector

In [5]:

```
#######################################################
# Define parameters
#######################################################

DEFAULT_CONFIG_FILE = os.path.join(yolo.PROJECT_ROOT, "svhn", "config.json")
DEFAULT_WEIGHT_FILE='/content/VRDL_2019/HW3/weights.h5'
DEFAULT_THRESHOLD = 0.3
```

Predict test data

In [0]:

```
import json

######################################################
# Define method to just build the network and load the weights
# so that the speed can be benchmarked with just the prediction
# and not the model loading as well
######################################################

def create_yolo_instance(conf=DEFAULT_CONFIG_FILE, weights=DEFAULT_WEIGHT_FILE):
  with open(conf) as config_buffer:
    config = json.loads(config_buffer.read())

  # 2. create yolo instance & predict
  yolo = create_yolo(config['model']['architecture'],
                     config['model']['labels'],
                     config['model']['input_size'],
                     config['model']['anchors'])
  yolo.load_weights(weights)
  return yolo


######################################################
# Define method to predict the 13068 test images
######################################################

def predict_testdata(image_folder="test/",conf=DEFAULT_CONFIG_FILE, weights=DEFAULT_WEI
GHT_FILE, threshold=DEFAULT_THRESHOLD):

      # Create list of dictionaries for submission
      prediction_dictionaries = [{} for _ in range(1,13069)]


      with open(conf) as config_buffer:
          config = json.loads(config_buffer.read())

      # 2. create yolo instance & predict
      yolo = create_yolo(config['model']['architecture'],
                         config['model']['labels'],
                         config['model']['input_size'],
                         config['model']['anchors'])
      yolo.load_weights(weights)

      # 3. read image
      write_dname = "detected"
      if not os.path.exists(write_dname): os.makedirs(write_dname)

      for i in range(1,13069):
          # For each image, get the predicted labels, probabilities and boxes

          img_path=image_folder+str(i)+".png"
          print(img_path)
          img_fname = os.path.basename(img_path)
          image = cv2.imread(img_path)
          boxes = [[]]
          probs = []
          labels = []
          boxes, probs = yolo.predict(image, float(threshold))
          labels = np.argmax(probs, axis=1) if len(probs) > 0 else []
```

```
            try:
              prediction_dictionaries[i-1]['bbox']=boxes.tolist()
            except:
              prediction_dictionaries[i-1]['bbox']=boxes

            prediction_dictionaries[i-1]['score']=[np.max(probs[i]) for i in range(len(pr
obs))]

            try:
              prediction_dictionaries[i-1]['label']=labels.tolist()
            except:
              prediction_dictionaries[i-1]['label']=labels

            #print(prediction_dictionaries)
            # 4. save detection result
            image = draw_scaled_boxes(image, boxes, probs, config['model']['labels'])
            output_path = os.path.join(write_dname, os.path.split(img_fname)[-1])

            cv2.imwrite(output_path, image)
            #print("{}-boxes are detected. {} saved.".format(len(boxes), output_path))

        #5. Write the list of dictionaries to a .json file for the submission
        with open("0845058.json", 'w') as f:
          f.write(str(prediction_dictionaries))

        #6. Upload the .json file to google drive
        file1 = drive.CreateFile()
        file1.SetContentFile('0845058.json')
        file1.Upload()
```

Timing for one picture

In [2]:

```
########################################################
#Create yolo instance
########################################################

conf = os.path.join(yolo.PROJECT_ROOT, "config_customized.json")
yolo = create_yolo_instance(conf=conf, weights=DEFAULT_WEIGHT_FILE)


########################################################
#Load image from test folder (unseen)
########################################################

image_folder="test/"
i=1 # load the first image
img_path=image_folder+str(i)+".png"
img_fname = os.path.basename(img_path)
image = cv2.imread(img_path)
```

In [22]:

```
#######################################################
# Benchmark the speed on google colab with TPU
#######################################################

%%timeit
yolo.predict(image, float(0.3))
```

```
The slowest run took 7.99 times longer than the fastest. This could mean t
hat an intermediate result is being cached.
1 loop, best of 3: 114 ms per loop
```

In [1]:

```
#######################################################
# Dtect digits of the test data
#######################################################

predict_testdata(conf="/content/VRDL_2019/HW3/testing.json",image_folder=image_folder,
weights=DEFAULT_WEIGHT_FILE)
```