Course Project – Object Detection

Tutorial of EE4211

TA: YANG Chen
Email: cyang53-c@my.cityu.edu.hk

Department of Electrical Engineering
City University of Hong Kong
03/29/2022

Outline

- Course Project
  - Objective
  - Kaggle Competition
- Machine Learning for Object Detection
  - Feature extraction by HOG
  - Classification by SVM
- Deep Learning for Object Detection
  - Network Architecture (RCNN)
  - Implement RCNN with Pytorch
- Further Extensions

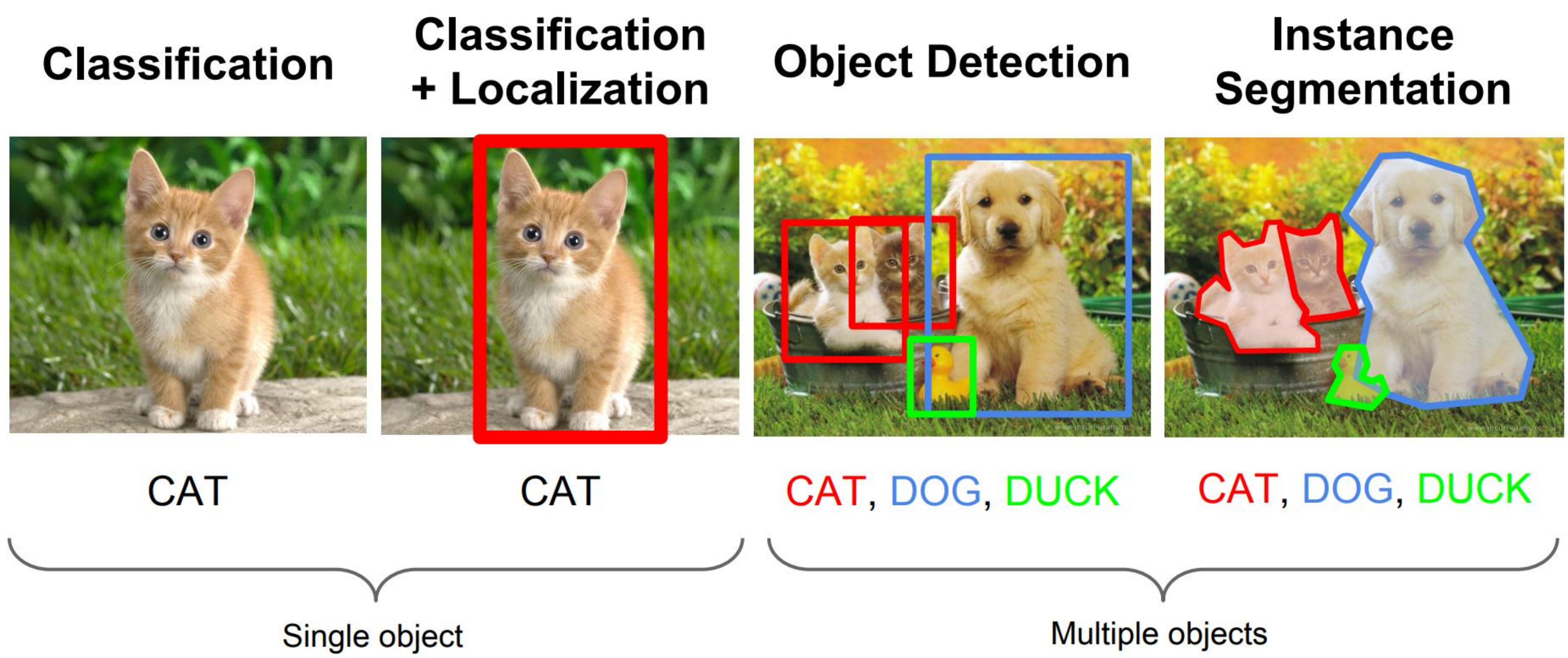
Outline

- Course Project
  - Objective
  - Kaggle Competition
- Machine Learning for Object Detection
  - Feature extraction by HOG
  - Classification by SVM
- Deep Learning for Object Detection
  - Network Architecture (RCNN)
  - Implement RCNN with Pytorch
- Further Extensions

Course Project - Objective

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

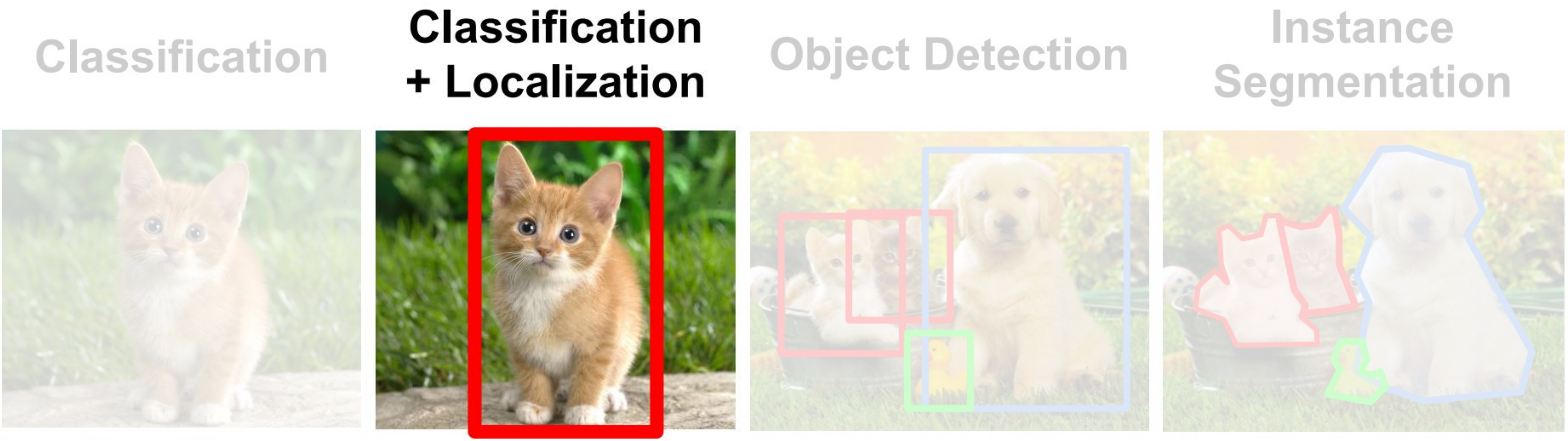
- Task –Object Detection



Course Project - Objective

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Task –Object Detection



Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Task –Object Detection

Classification + Localization: Task

Classification: C classes
  Input: Image
  Output: Class label
  Evaluation metric: Accuracy

→ CAT

Localization:
  Input: Image
  Output: Box in the image (x, y, w, h)
  Evaluation metric: Intersection over Union

→ (x, y, w, h)

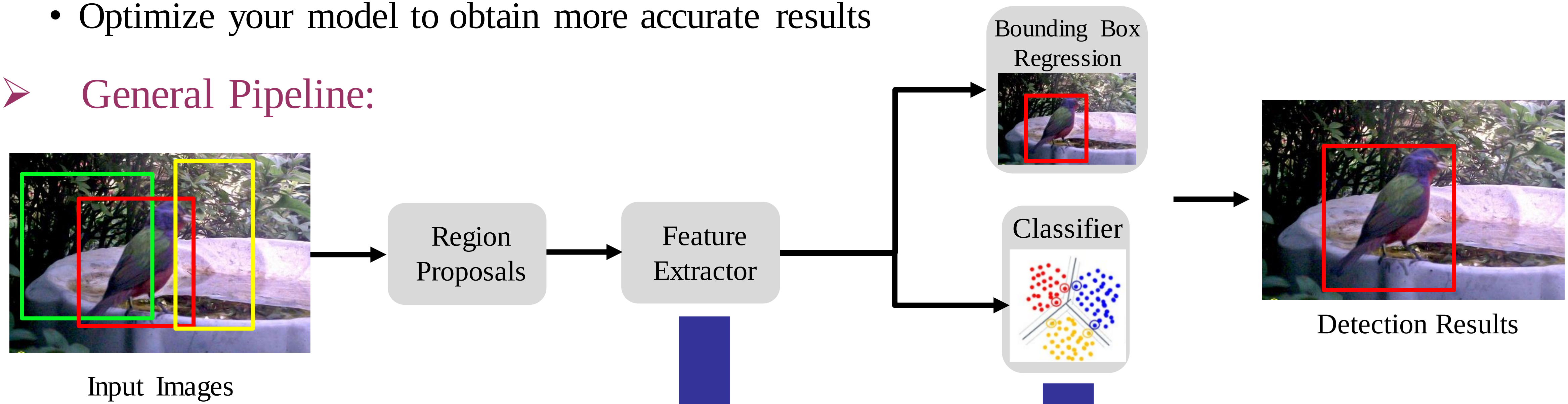
Classification + Localization: Do both

Course Project - Objective

➢ Project objective:

- Using machine learning or deep learning methods to solve object detection problem
- Optimize your model to obtain more accurate results

➢ General Pipeline:



- Handcrafted:
  - HOG, SIFT
- ConvNets:
  - VGG, ResNet

- Linear classifier
  - SVM
- Conv classifier:
  - CNN

Course Project - Kaggle Competition

- How to participate:

  URL: https://www.kaggle.com/c/ee4211-object-detection/overview

- Data Information
  - Train/Test input images
  - Train bounding boxes (csv file)

- Data
  - Test
  - Train
  - train.csv

- Quick start
  - A baseline folder helping you quickly get start.

- Deadline:
  - 04/05/2022

Course Project - Kaggle Competition

CityU Department of Electrical Engineering
香港城市大學
City University of Hong Kong

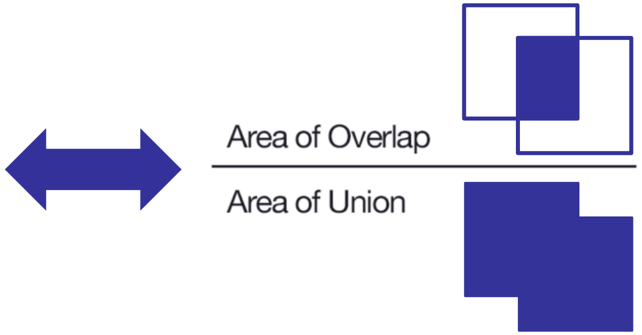
- Result submission
  - Generate two-column results (.csv):
    Image ID (referred to image name)
    Corresponding Bounding Boxed of Predictions. (Arbitrary order is OK)

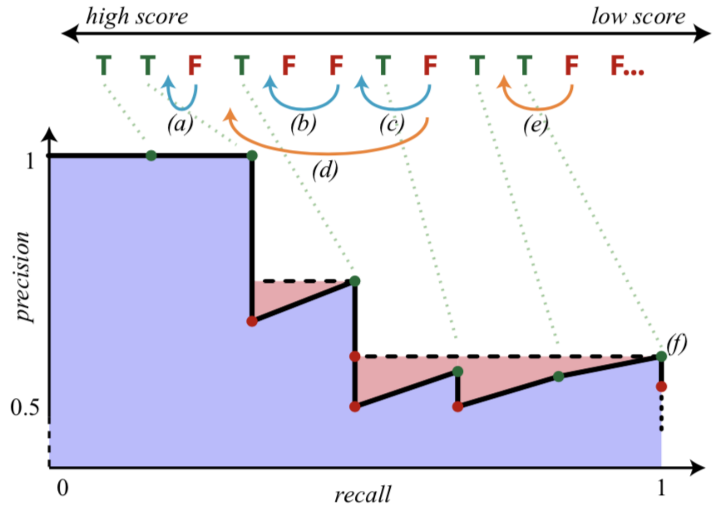
ImageId	PredictionString
0.jpg	1 1.0 0.21 0.23 0.41 0.65
1.jpg	1 1.0 0.43 0.35 0.5 0.47
2.jpg	1 1.0 0.18 0.14 0.66 0.76
3.jpg	1 1.0 0.1 0.25 0.65 0.61
4.jpg	1 1.0 0.22 0.21 0.52 0.66
5.jpg	1 1.0 0.01 0.18 0.79 0.8
6.jpg	1 1.0 0.27 0.27 0.48 0.52
7.jpg	1 1.0 0.37 0.3 0.52 0.51
8.jpg	1 1.0 0.14 0.06 0.52 0.71
9.jpg	1 1.0 0.14 0.14 0.83 0.85
10.jpg	1 1.0 0.12 0.47 0.88 0.39
11.jpg	1 1.0 0.25 0.15 0.3 0.81

  - PredictionString-definition of category id, prediction probability, Xmin, Ymin, Xmax, Ymax

- Evaluation matrix: mean average precision (mAP)

$IoU(A,B)=(A\cap B)/(A\cup B)$.





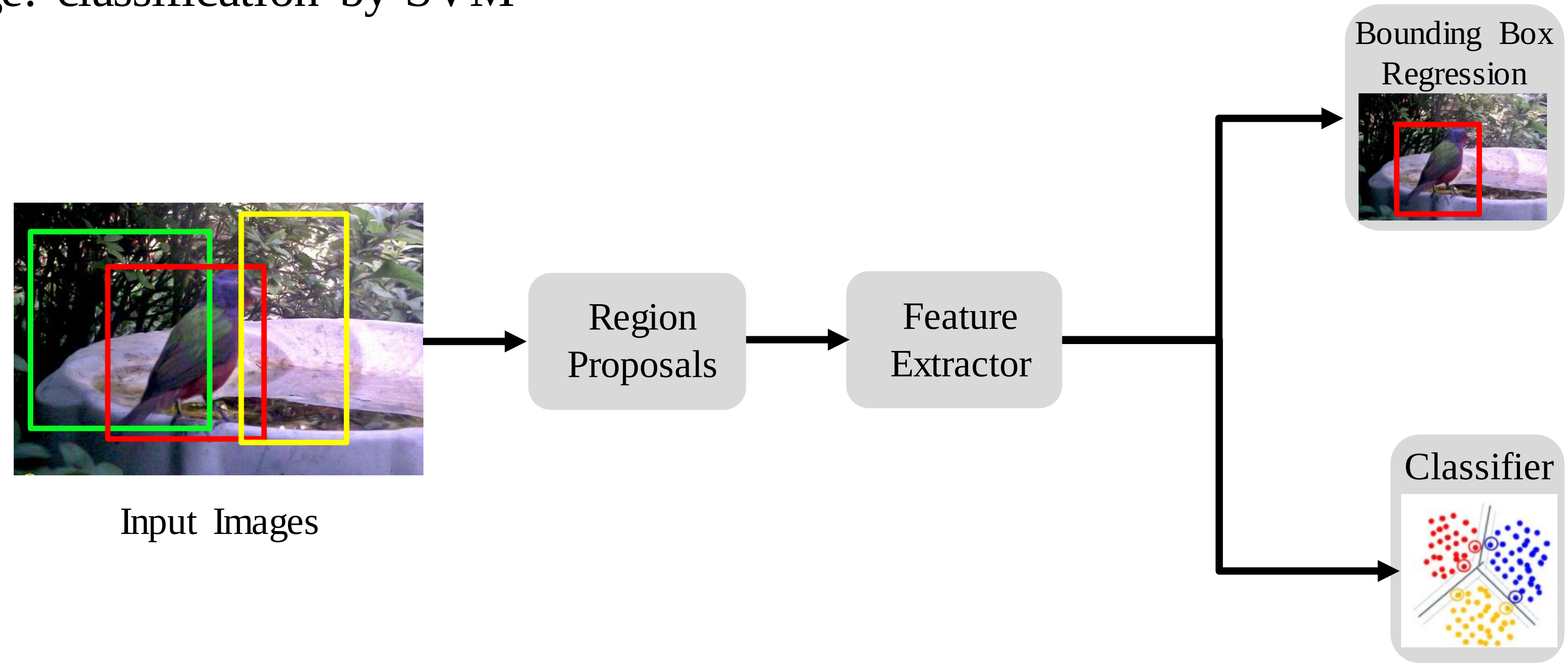
Outline

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Course Project
  - Objective
  - Kaggle Competition
- Machine Learning for Object Detection
  - Feature extraction by HOG
  - Classification by SVM
- Deep Learning for Object Detection
  - Network Architecture (RCNN)
  - Implement RCNN with Pytorch
- Further Extensions

General Pipeline – two stages

- First stage: feature extraction by HOG
- Second stage: classification by SVM



Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- General Pipeline – 6 steps

1. Sampling positive images
2. Sampling negative images
3. Training a Linear SVM
4. Performing hard-negative mining
5. Re-training your Linear SVM using the hard-negative samples
6. Evaluating your classifier on your test dataset, utilizing non-maximum suppression to ignore redundant, overlapping bounding boxes

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

General Pipeline – code structure

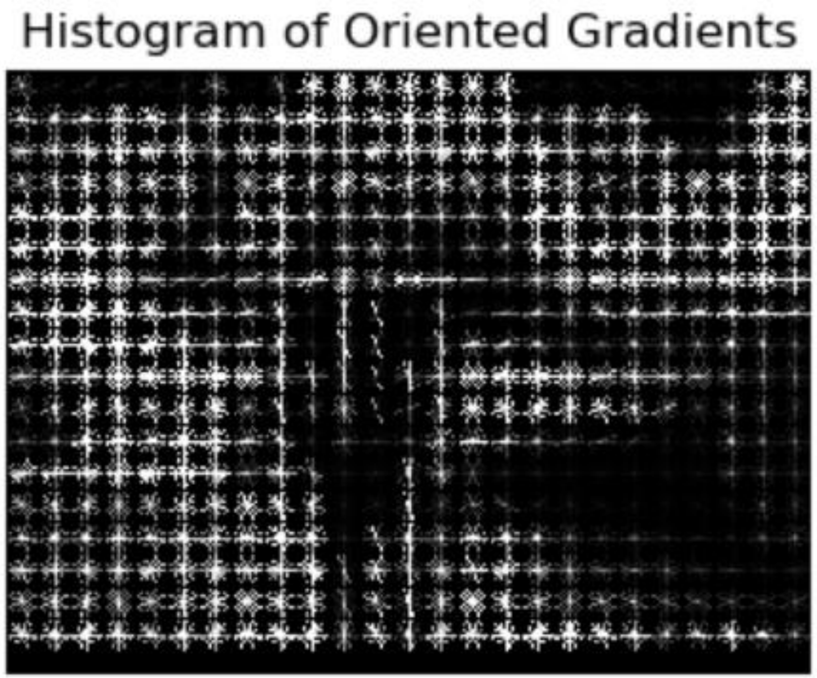
```
EE4211-DET [SSH: HPC.EE.CITYU.EDU.HK]
  ML-based
    bin
    data
    object-detector
      __init__.py
      config.py
      extract-features.py
      nms.py
      test-classifier.py
      train-classifier.py
    .gitignore
    LICENSE
    README.md
```

About the modules

- `extract-features.py` -- This module is used to extract HOG features of the training images.
- `train-classifier.py` -- This module is used to train the classifier.
- `nms.py` -- This module performs Non Maxima Suppression.
- `test-classifier.py` -- This module is used to test the classifier using a test image.
- `config.py` -- Imports the configuration variables from `config.cfg`.

First stage: feature extraction by HOG

- An example of obtaining HOG feature vectors.



Department of Electrical Engineering
香港城市大學
City University of Hong Kong

➢ First stage: feature extraction by HOG

- [code]: *https://github.com/aarcosg/object-detector-svm-hog-python*

```
for im_path in glob.glob(os.path.join(pos_im_path, "*")):
    im = imread(im_path, as_grey=True)
    if des_type == "HOG":
        fd = hog(im, orientations, pixels_per_cell, cells_per_block, visualize, normalize)
    fd_name = os.path.split(im_path)[1].split(".")[0] + ".feat"
    fd_path = os.path.join(pos_feat_ph, fd_name)
    joblib.dump(fd, fd_path)
print("Positive features saved in {}".format(pos_feat_ph))

print("Calculating the descriptors for the negative samples and saving them")
for im_path in glob.glob(os.path.join(neg_im_path, "*")):
    im = imread(im_path, as_grey=True)
    if des_type == "HOG":
        fd = hog(im,  orientations, pixels_per_cell, cells_per_block, visualize, normalize)
    fd_name = os.path.split(im_path)[1].split(".")[0] + ".feat"
    fd_path = os.path.join(neg_feat_ph, fd_name)
    joblib.dump(fd, fd_path)
print("Negative features saved in {}".format(neg_feat_ph))

print("Completed calculating features from training images")
```

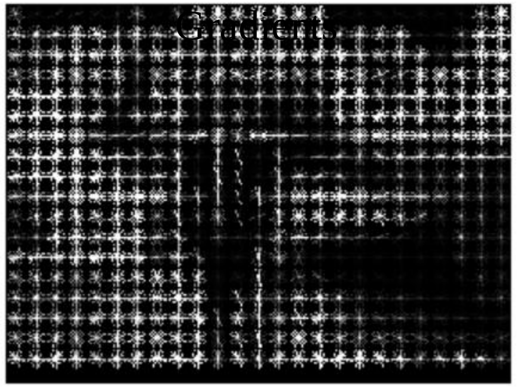
➢ Second stage: classification by SVM

- After classifying with a trained SVM model and applying NMS the following result is achieved:

Detection Prediction:

Input image

Histogram of Oriented Gradients



detection result

Department of Electrical Engineering
香港城市大學 City University of Hong Kong

Second stage: classification by SVM

- [code]: *https://github.com/aarcosg/object-detector-svm-hog-python*

```
pos_feat_path =  args["posfeat"]
neg_feat_path = args["negfeat"]

# Classifiers supported
clf_type = args['classifier']

fds = []
labels = []
# Load the positive features
for feat_path in glob.glob(os.path.join(pos_feat_path,"*.feat")):
    fd = joblib.load(feat_path)
    fds.append(fd)
    labels.append(1)

# Load the negative features
for feat_path in glob.glob(os.path.join(neg_feat_path,"*.feat")):
    fd = joblib.load(feat_path)
    fds.append(fd)
    labels.append(0)

if clf_type is "LIN_SVM":
    clf = LinearSVC()
    print("Training a Linear SVM Classifier")
    clf.fit(fds, labels)
    # If feature directories don't exist, create them
    if not os.path.isdir(os.path.split(model_path)[0]):
        os.makedirs(os.path.split(model_path)[0])
    joblib.dump(clf, model_path)
    print("Classifier saved to {}".format(model_path))
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

Test stage: test image by SVM

- [code]: *https://github.com/aarcosg/object-detector-svm-hog-python*

```python
# Read the image
im = imread(args["image"], as_grey=False)
min_wdw_sz = (100, 40)
step_size = (10, 10)
downscale = args['downscale']
visualize_det = args['visualize']

# Load the classifier
clf = joblib.load(model_path)

# List to store the detections
detections = []
# The current scale of the image
scale = 0
# Downscale the image and iterate
for im_scaled in pyramid_gaussian(im, downscale=downscale):
    # This list contains detections at the current scale
    cd = []
    # If the width or height of the scaled image is less than
    # the width or height of the window, then end the iterations.
    if im_scaled.shape[0] < min_wdw_sz[1] or im_scaled.shape[1] < min_wdw_sz[0]:
        break
    for (x, y, im_window) in sliding_window(im_scaled, min_wdw_sz, step_size):
        if im_window.shape[0] != min_wdw_sz[1] or im_window.shape[1] != min_wdw_sz[0]:
            continue
        # Calculate the HOG features
        fd = hog(im_window, orientations, pixels_per_cell, cells_per_block, visualize, normalize)
        pred = clf.predict(fd)
        if pred == 1:
            print("Detection:: Location -> ({}, {})".format(x, y))
            print("Scale ->  {} | Confidence Score {} \n".format(scale,clf.decision_function(fd)))
            detections.append((x, y, clf.decision_function(fd),
                int(min_wdw_sz[0]*(downscale**scale)),
                int(min_wdw_sz[1]*(downscale**scale))))
            cd.append(detections[-1])
```

```python
        # If visualize is set to true, display the working
        # of the sliding window
        if visualize_det:
            clone = im_scaled.copy()
            for x1, y1, _, _, _  in cd:
                # Draw the detections at this scale
                cv2.rectangle(clone, (x1, y1), (x1 + im_window.shape[1], y1 +
                    im_window.shape[0]), (0, 0, 0), thickness=2)
            cv2.rectangle(clone, (x, y), (x + im_window.shape[1], y +
                im_window.shape[0]), (255, 255, 255), thickness=2)
            cv2.imshow("Sliding Window in Progress", clone)
            cv2.waitKey(30)
    # Move the the next scale
    scale+=1

# Display the results before performing NMS
clone = im.copy()
for (x_tl, y_tl, _, w, h) in detections:
    # Draw the detections
    cv2.rectangle(im, (x_tl, y_tl), (x_tl+w, y_tl+h), (0, 0, 0), thickness=2)
cv2.imshow("Raw Detections before NMS", im)
cv2.waitKey()

# Perform Non Maxima Suppression
detections = nms(detections, threshold)

# Display the results after performing NMS
for (x_tl, y_tl, _, w, h) in detections:
    # Draw the detections
    cv2.rectangle(clone, (x_tl, y_tl), (x_tl+w,y_tl+h), (0, 0, 0), thickness=2)
cv2.imshow("Final Detections after applying NMS", clone)
cv2.waitKey()
```

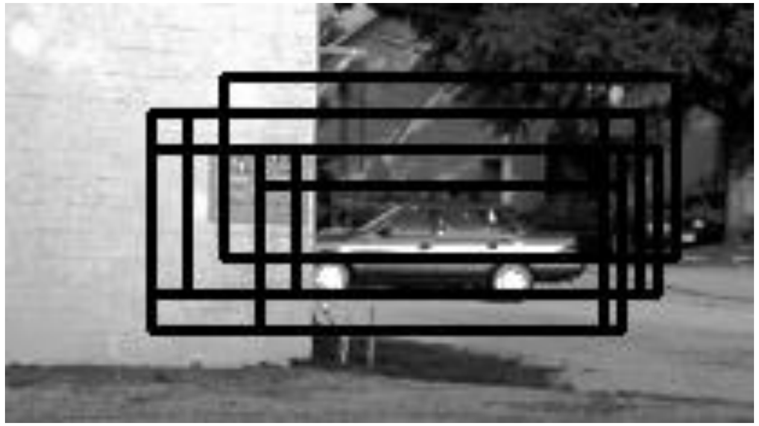
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

➢ Test stage: Non-Max Suppression (NMS)

- [code]: *https://github.com/aarcosg/object-detector-svm-hog-python*

```
def nms(detections, threshold=.5):
    '''
    This function performs Non-Maxima Suppression.
    `detections` consists of a list of detections.
    Each detection is in the format ->
    [x-top-left, y-top-left, confidence-of-detections, width-of-detection, height-of-detection]
    If the area of overlap is greater than the `threshold`,
    the area with the lower confidence score is removed.
    The output is a list of detections.
    '''
    if len(detections) == 0:
        return []
    # Sort the detections based on confidence score
    detections = sorted(detections, key=lambda detections: detections[2],
            reverse=True)
    # Unique detections will be appended to this list
    new_detections=[]
    # Append the first detection
    new_detections.append(detections[0])
    # Remove the detection from the original list
    del detections[0]
    # For each detection, calculate the overlapping area
    # and if area of overlap is less than the threshold set
    # for the detections in `new_detections`, append the
    # detection to `new_detections`.
    # In either case, remove the detection from `detections` list.
    for index, detection in enumerate(detections):
        for new_detection in new_detections:
            if overlapping_area(detection, new_detection) > threshold:
                del detections[index]
                break
        else:
            new_detections.append(detection)
            del detections[index]
    return new_detections
```

Detections before NMS

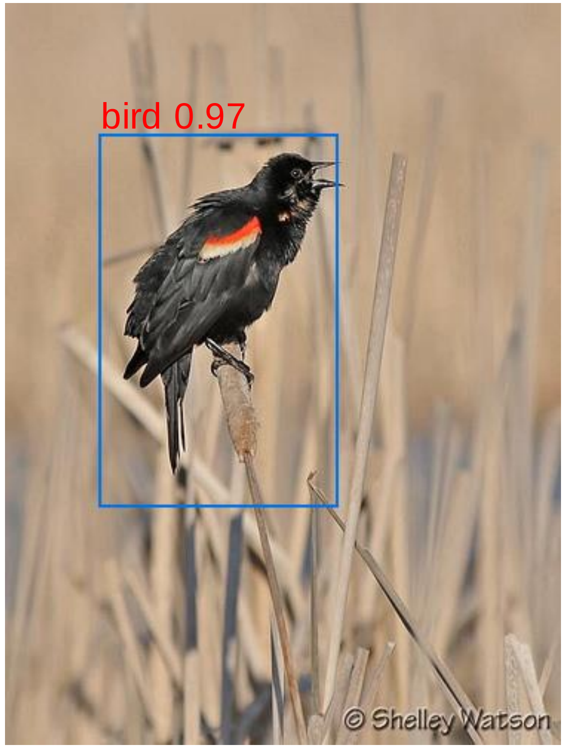


Detections after NMS

Machine Learning – Results

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

➢ Results on bird detection



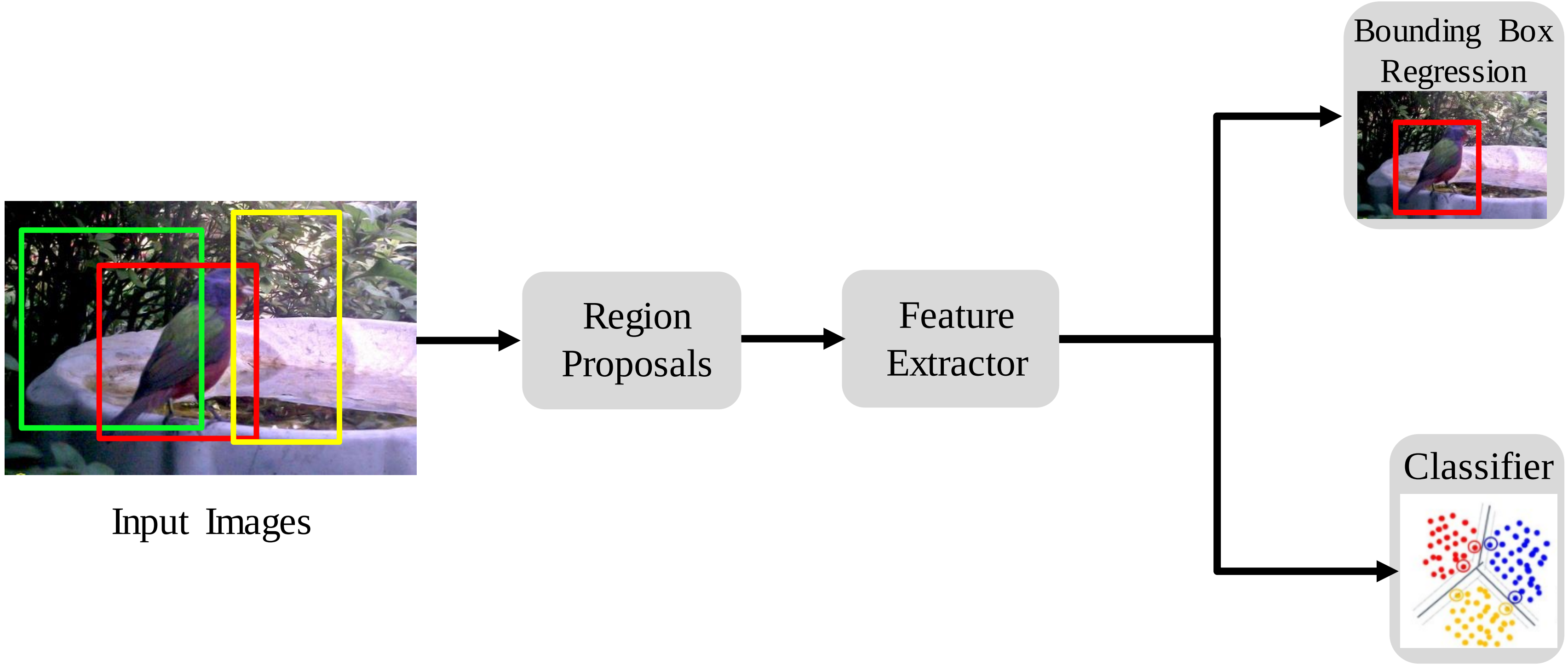
Outline

- Course Project
  - Objective
  - Kaggle Competition
- Machine Learning for Object Detection
  - Feature extraction by HOG
  - Classification by SVM
- Deep Learning for Object Detection
  - Network Architecture (RCNN)
  - Implement RCNN with Pytorch
- Further Extensions

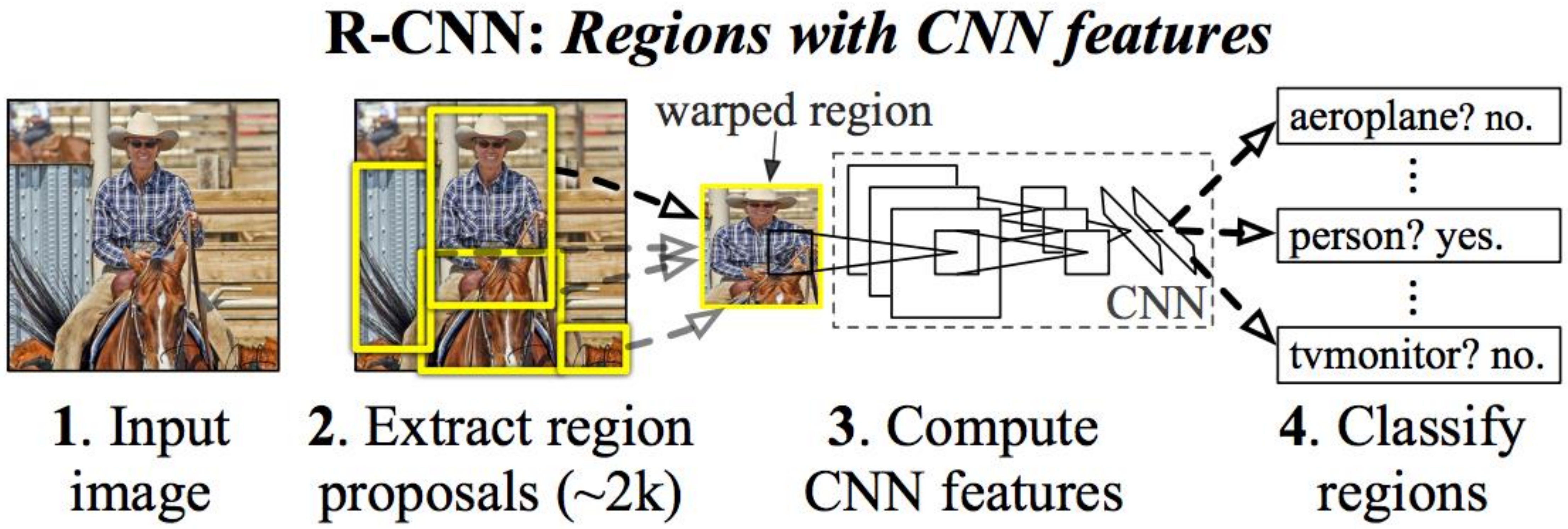
Department of Electrical Engineering
香港城市大學
City University of Hong Kong

➢ General Pipeline – one stage

- End-to-end manner: joint feature extraction and classification



Deep Learning – Network Architecture (RCNN)

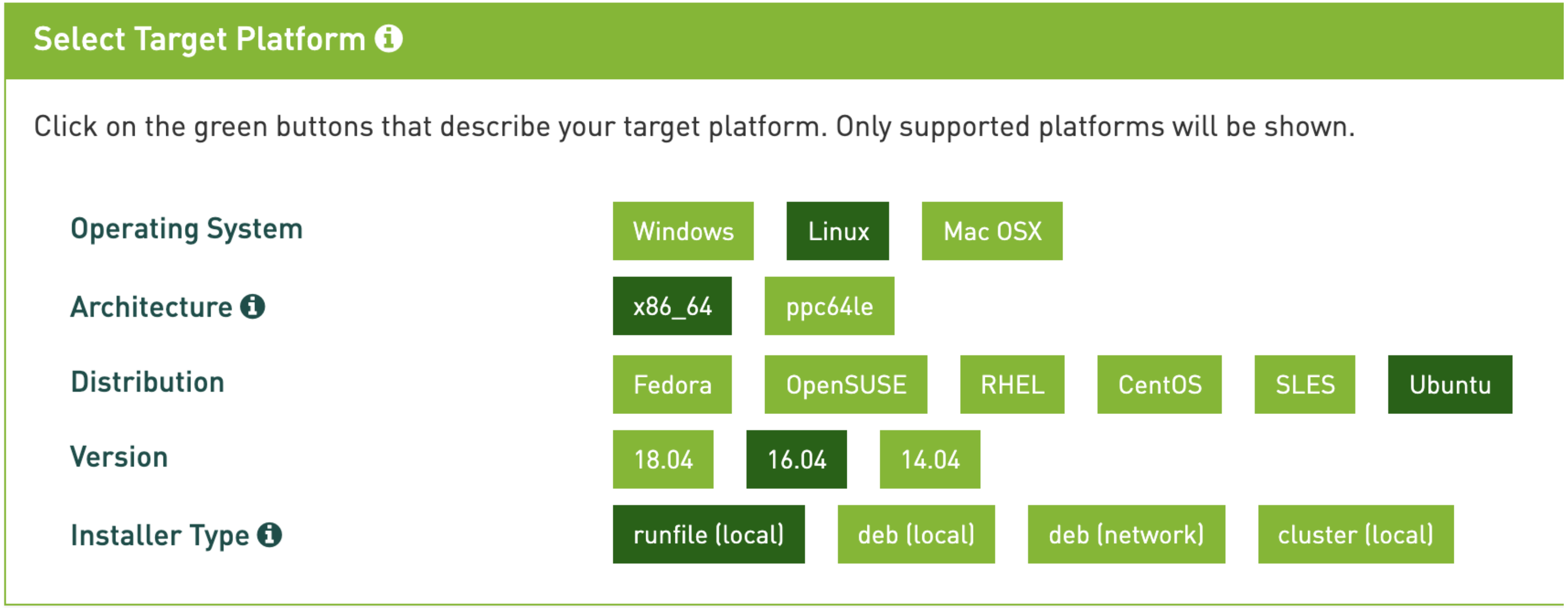


Girshick, Ross, et al. "Rich feature hierarchies for accurate object detection and semantic segmentation." CVPR. 2014.

➢ Install CUDA

NVIDIA DEVELOPER

- Download: https://developer.nvidia.com/cuda-10.0-download-archive



Installation Instructions:

1. Run `sudo sh cuda_10.0.130_410.48_linux.run`
2. Follow the command-line prompts

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

➢ Install cuDNN

NVIDIA. DEVELOPER

- Download: https://developer.nvidia.com/rdp/cudnn-archive

Download cuDNN v8.0.1 RC2 (June 26th, 2020), for CUDA 10.2

Download cuDNN v7.6.5 (November 18th, 2019), for CUDA 10.2

Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 10.1

Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 10.0

Download cuDNN v7.6.5 (November 5th, 2019), for CUDA 9.2

- You should choose the matched version of cuDNN with the installed CUDA
- Installation instruction: https://docs.nvidia.com/deeplearning/cudnn/install-guide/index.html#installlinux

Department of Electrical Engineering
香港城市大學 City University of Hong Kong

➢ Install Anaconda

ANACONDA

- Download: https://www.anaconda.com/products/individual#linux

Anaconda Installers

Windows

Python 3.8

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (397 MB)

MacOS

Python 3.8

64-Bit Graphical Installer (462 MB)

64-Bit Command Line Installer (454 MB)

Linux

Python 3.8

64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

➢ Install Anaconda

ANACONDA

- Install Steps:

```
cd /home  # The path you download Anaconda
bash Anaconda3-2020.07-Linux-x86_64.sh  # Enter,
          # Follow the command-line prompts and answer each question

# Once install successfully, test with
python  # If there is an error indicating ‘cannot find python’
sudo vim /etc/profile  # Configure and edit the environment variables
PATH=/home/user/anaconda3/bin:$PATH  # Add this line into the file
           # Save this configure file with ‘:wq!’
source /etc/profile  # Update the environment variables

# Test again
python
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

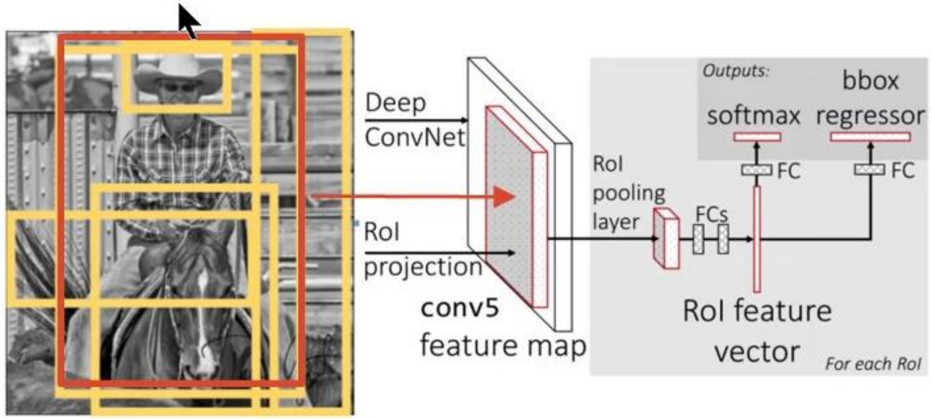
- Install torch, numpy, opencv-python, scikit-image, scipy

PyTorch NumPy scikit-image image processing in python SciPy library Fundamental library for scientific computing

```
conda create -n torch020 python=3.6  # construct virtual environment named pytorch
conda activate torch020  # activate virtual environment
Pip list  # show the installed library

pip install torch==1.7.0
pip install numpy
pip install opencv-python
pip install scikit-image
pip install scipy
# The resting required libraries can be installed following the reported error when debugging
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- RCNN
  ./lib/net/data.py

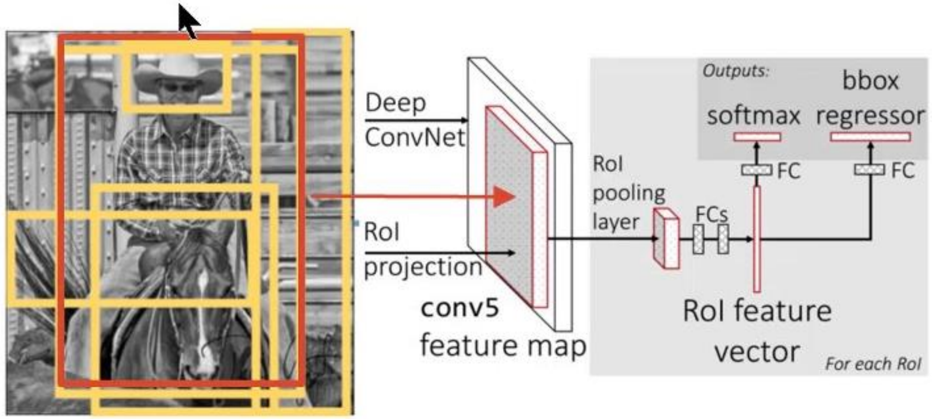



```python
train_imgs = []
train_img_info = []
train_roi = []
train_cls = []
train_tbbox = []

N_train = len(data)
for i in trange(N_train):
    img_path = data['image_name'][i]
    gt_boxs = data['boxes'][i]
    gt_classes = data['gt_classes'][i]
    nobj = data['num_objs'][i]
    bboxs = data['selective_search_boxes'][i]
    nroi = len(bboxs)

    img = Image.open('data/JPEGImages/' + img_path)
    img_size = img.size
    img = img.resize((224, 224))
    img = np.array(img).astype(np.float32)
    img = np.transpose(img, [2, 0, 1])

    rbboxs = rel_bbox(img_size, bboxs)
    ious = calc_ious(bboxs, gt_boxs)
    max_ious = ious.max(axis=1)
    max_idx = ious.argmax(axis=1)
    tbbox = bbox_transform(bboxs, gt_boxs[max_idx])
```

```python
train_imgs = np.array(train_imgs)
train_img_info = np.array(train_img_info)
train_roi = np.array(train_roi)
train_cls = np.array(train_cls)
train_tbbox = np.array(train_tbbox).astype(np.float32)
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- RCNN
  ./lib/net/RCNN.py




```python
class RCNN(nn.Module):
    def __init__(self):
        super().__init__()

        rawnet = torchvision.models.vgg16_bn(pretrained=True)
        self.seq = nn.Sequential(*list(rawnet.features.children())[:-1])
        self.roipool = SlowROIPool(output_size=(7, 7))
        self.feature = nn.Sequential(*list(rawnet.classifier.children())[:-1])

        _x = Variable(torch.Tensor(1, 3, 224, 224))
        _r = np.array([[0., 0., 1., 1.]])
        _ri = np.array([0])
        _x = self.feature(self.roipool(self.seq(_x), _r, _ri).view(1, -1))
        feature_dim = _x.size(1)
        self.cls_score = nn.Linear(feature_dim, N_CLASS+1)
        self.bbox = nn.Linear(feature_dim, 4*(N_CLASS+1))

        self.cel = nn.CrossEntropyLoss()
        self.sl1 = nn.SmoothL1Loss()

    def forward(self, inp, rois, ridx):
        res = inp
        res = self.seq(res)
        res = self.roipool(res, rois, ridx)
        res = res.detach()
        res = res.view(res.size(0), -1)
        feat = self.feature(res)

        cls_score = self.cls_score(feat)
        bbox = self.bbox(feat).view(-1, N_CLASS+1, 4)
        return cls_score, bbox

    def calc_loss(self, probs, bbox, labels, gt_bbox):
        loss_sc = self.cel(probs, labels)
        lbl = labels.view(-1, 1, 1).expand(labels.size(0), 1, 4)
        mask = (labels != 0).float().view(-1, 1).expand(labels.size(0), 4)
        loss_loc = self.sl1(bbox.gather(1, lbl).squeeze(1) * mask, gt_bbox * mask)
        lmb = 1.0
        loss = loss_sc + lmb * loss_loc
        return loss, loss_sc, loss_loc
```

Department of Electrical Engineering
香港城市大學
City University of Hong Kong

- Loss calculation
  ./experiment/rcnn/train.py

Detection Results

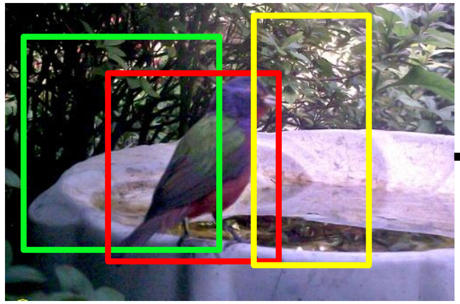
```python
def train_epoch(run_set, is_val=False):
    for i in trange(0, Nimg, I):
        lb = i
        rb = min(i+I, Nimg)
        torch_seg = torch.from_numpy(perm[lb:rb])
        img = Variable(train_imgs[torch_seg], volatile=is_val).cuda()
        ridx = []
        glo_ids = []

        for j in range(lb, rb):
            info = train_img_info[perm[j]]
            pos_idx = info['pos_idx']
            neg_idx = info['neg_idx']
            ids = []

            if len(pos_idx) > 0:
                ids.append(np.random.choice(pos_idx, size=POS))
            if len(neg_idx) > 0:
                ids.append(np.random.choice(neg_idx, size=NEG))
            if len(ids) == 0:
                continue
            ids = np.concatenate(ids, axis=0)
            glo_ids.append(ids)
            ridx += [j-lb] * ids.shape[0]

        if len(ridx) == 0:
            continue
        glo_ids = np.concatenate(glo_ids, axis=0)
        ridx = np.array(ridx)
        rois = train_roi[glo_ids]
        gt_cls = Variable(torch.from_numpy(train_cls[glo_ids]), volatile=is_val).cuda()
        gt_tbbox = Variable(torch.from_numpy(train_tbbox[glo_ids]), volatile=is_val).cuda()

        loss, loss_sc, loss_loc = train_batch(img, rois, ridx, gt_cls, gt_tbbox, is_val=is_val)
        losses.append(loss)
        losses_sc.append(loss_sc)
        losses_loc.append(loss_loc)
    avg_loss = np.mean(losses)
    avg_loss_sc = np.mean(losses_sc)
    avg_loss_loc = np.mean(losses_loc)
    print(f'Avg loss = {avg_loss:.4f}; loss_sc = {avg_loss_sc:.4f}, loss_loc = {avg_loss_loc:.4f}')
    return losses, losses_sc, losses_loc
```

- Results on bird detection

Outline

- Course Project
  - Objective
  - Kaggle Competition
- Machine Learning for Object Detection
  - Feature extraction by HOG
  - Classification by SVM
- Deep Learning for Object Detection
  - Network Architecture (RCNN)
  - Implement RCNN with Pytorch
- Further Extensions

Further Extensions



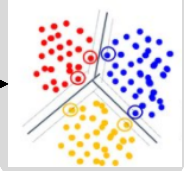
Input Images

Region Proposals

Feature Extractor

Bounding Box Regression

Classifier



- Better data
  - Data augmentation
  - Color space transformation
- Better feature vectors
  - Combine ML hand-crafted features with DL features
- Better supervisions
  - Contrastive learning

Thanks for listening!