HW2-18-794: Introduction to Deep Learning and Pattern Recognition for Computer Vision

Assignment 2: Face Detection

Instructor: Marios Savvides

Due Date: Oct 19, 2023

Total Points: 100

Submission: Submit your solutions and plots on Gradescope.

START HERE: Instructions

- **Collaboration policy:** All are encouraged to work together BUT you must do your own work (code and write up). If you work with someone, please include their name in your write-up and cite any code that has been discussed. If we find highly identical write-ups or code or lack of proper accreditation of collaborators, we will take action according to strict university policies. See the Academic Integrity Section detailed in the initial lecture for more information.
- **Late Submission Policy:** There are a **total of 5** late days across all homework submissions. Submissions more than 5 days after the deadline will receive a 0.

  For those students taking Mini-1 (semi-semester), you cannot submit it later than Oct 21, 2023, as the grading deadline of mini-1 is due on Oct 23, 2023.
- **Submitting your work:**
  - We will be using Gradescope (`https://gradescope.com/`) to submit the Problem Sets. Please use the provided template only. Submissions must be written in LaTeX. All submissions not adhering to the template will not be graded and receive a zero.
  - **Deliverables:** Please submit all the `.py` files. Add all relevant plots and text answers in the boxes provided in this file. TO include plots you can simply modify the already provided latex code. Submit the compiled `.pdf` report as well.

NOTE: Partial points will be given for implementing parts of the homework even if you don't get the mentioned numbers as long as you include partial results in this pdf.

Overview

In this assignment, you will train and test a state-of-the-art face detector that is deployed in the real world applications.

1 Understand your data (POINTS : 30)

Understanding your data is always the first step in training deep learning models, which will greatly ease the following works. This step usually involves statistical analysis, preprocessing, and visualization for detection datasets.

Data path:

https://drive.google.com/file/d/1799456S54_M7CS9Gom1xrD9_tIKavPuB/view?usp=sharing

Note 1. The bounding box is indicated by the first 4 floats and is in (x1, y1, w, h). Where x1, y1 is the top-left corner, followed by the 5 landmarks points (L_x, L_y), each landmark point is followed by a 0/1 that you can ignore. The landmarks might be -1 indicating no landmark available for this face.

1.1 Write code to do the following statistical analysis

1. Total number of ground truths (GT), i.e., faces, for training and validation.

2. Average area, width, and height of bounding boxes

3. Are faces uniformly distributed on images? Give your analysis.

4. Check if any GT exceeds the range of the image. If yes, how do you plan to solve it?

5. How serious is the mutual overlap among GTs? Calculate the percentage of GTs that overlap with one another (i.e., overlapped GT vs total number of GT). Hint: check the overlap using IOU > 0.

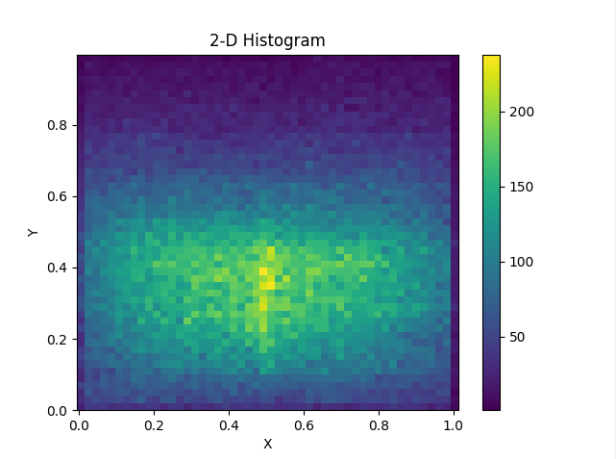
Solution

1. Faces found in widerface_homework/train: 159424
Faces found in widerface_homework/val: 39708
Found 199132 faces in total

All subsequent answers are for considering the training and validation set together

2. Average width: 29.00, Average height: 37.44, Average area: 3847.96

3. No they are not uniformly distributed on the images. Here's a scaled 2d histogram of face location frequencies. Clearly you can see that the bottom-middle-center location has the most concentration of faces.

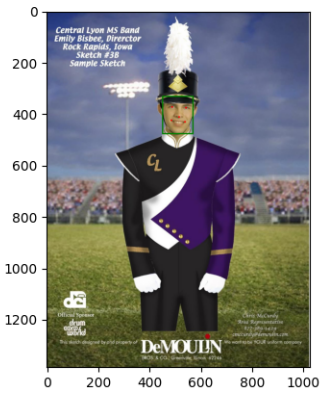


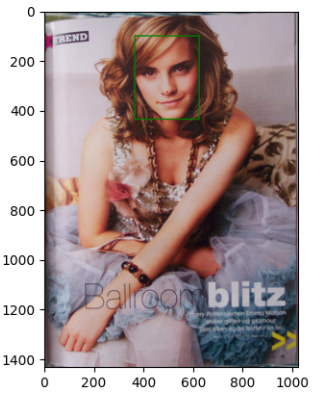
4. I haven't been able to find any GT that exceeds the range of the image. If it did, I would simply bound the GT to the dimensions of the image.

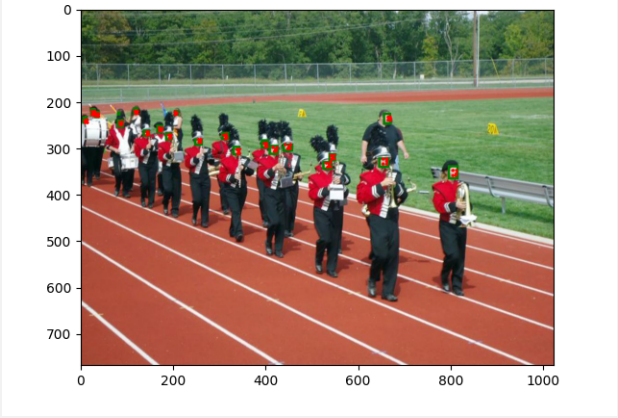
5. When calculating GTs, I got 23241 overlaps with 199132 total making for 11.67% of GTs overlapping. Is this serious? I don't think so, there might be confuounding if there is too much overlap, but given it's only 10% of the dataset, I am not too concerned, but a more experienced researcher might know better.

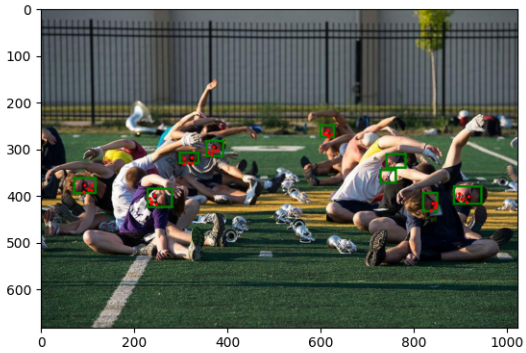
1.2 Visualize the first 4 images and their ground truths, including the faces (in green) and landmarks (in red).

Solution









2 Initialize your detector and build up the forward path (./detector/mydetector.py).

This face detector has 4 components: 1. ResNet 50 as Backbone. 2. Feature pyramid network (FPN). 3. A special module named SSH. 4. Three heads for classification, bounding box regression, and landmarks, respectively.

Note 1: Since the FPN needs a feature pyramid extracted from the backbone, you will need to use

`torchvision.models._utils.IntermediateLayerGetter`

to gather these intermediate layers from backbone.

Note 2: The classification head aims to classify each anchor as positive or negative, because we are doing face/not face here. Regression aims to regress a bounding box for each anchor, it requires 4 degrees of freedom to do so. Landmark head aims to regress 5 landmark points, it requires 10 degrees of freedom to do so.

2.1 Initialize backbone and gather intermediate layers, the layers to-be-gather are defined in cfg['return layers']

Note: cfg is "configuration" that is specified in config.py

Solution

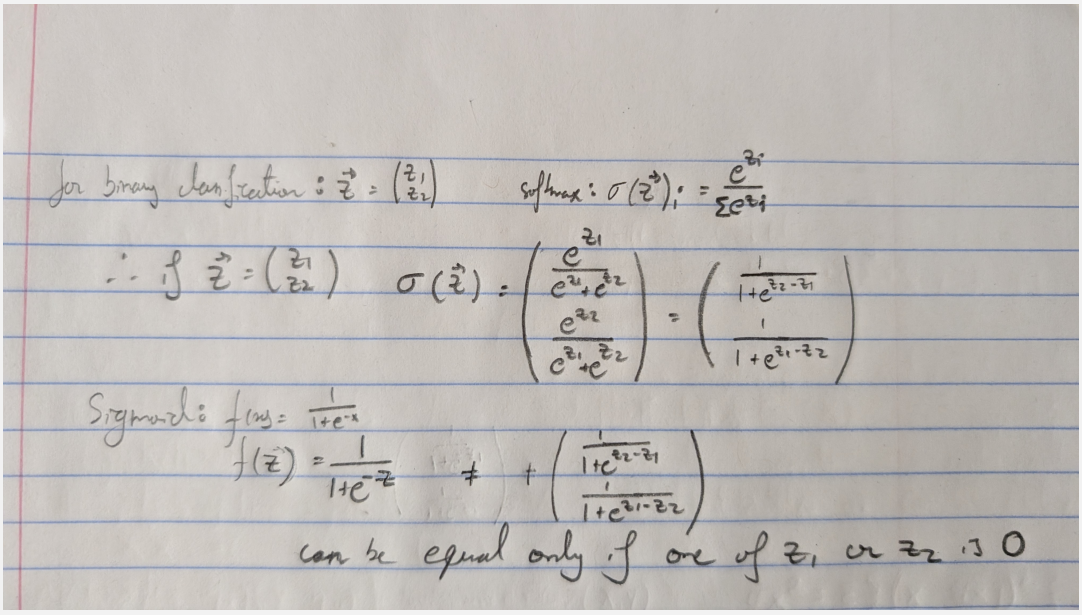
```
backbone = models.resnet50()
self.body = IntermediateLayerGetter(backbone, cfg['return_layers'])
```

2.2 Understand how we make the classification head. What is the output dimension of the head? And, for the binary classification, is sigmoid mathematically equivalent to softmax? Show your proof.

Solution

There are three classification heads, each takes in a feature map of different size. Each classification head will output a shape of dimension "<batch_size, size_of_feature_map, degrees_of_freedom>"

The bath_size and degrees_of_freedom are set to 2. The size_of_feature_map varies between 22050, 5618, and 1458.



Its important to note that softmax typically takes in a vector of values to calulate the relative probabilities. Sigmoid can only take in one value, which it then converts into a probability from 0 to 1.

Thus, during binary classification, we would have a situation where we would be comparing a vector and a scalar. The to resolve this, is to realize that the value provided by the sigmoid must (in an ideal world) be equal to one of the values of the softmax (since the 2nd value of the softmax is just one minus the first value). This match can only happen if the input to the sigmoid, z_s is equal to the difference between the inputs to the softmax $z_1 - z_2$ or $z_2 - z_1$

Although, they follow a very similar structure, they cannot be conisered equivalent.

2.3 Following the code of the classification head, finish the regression head and landmarks head.

Solution

```
class BboxHead(nn.Module):
    def __init__(self, inchannels=512, num_anchors=3):
        super(BboxHead,self).__init__()
        self.num_anchors = num_anchors
        self.conv1x1 = nn.Conv2d(inchannels,
                        self.num_anchors*4,kernel_size=
                        (1,1),stride=1,padding=0)

class LandmarkHead(nn.Module):
    def __init__(self, inchannels=512, num_anchors=3):
        super(LandmarkHead, self).__init__()
        self.num_anchors = num_anchors
        self.conv1x1 = nn.Conv2d(inchannels,
                       self.num_anchors*10,kernel_size=
                       (1,1),stride=1,padding=0)
```

2.4 Complete the forward function of your detection. The process is Input image $->$ gather feature maps $->$ FPN $->$you got 3 feature pyramids $->$ for each feature pyramid, apply a predefined SSH module to it $->$for each feature pyramid, apply three heads to it.

3 Dataloader (data/dataloader.py)

Your dataloader feeds images, bounding boxes, and landmarks to the detector for training.

3.1 initialize dataloader

```
In __init__, load the widerface/train/label.txt,
and store the path of images in self.imgs_path as a list.
Store the annotations in self.words. The structure must be
self.imgs_path: [path1, path2, path3]
self.words: [[img1_anno1, img1_anno2, ...]
             [img2_anno1, img2_anno2, img2_anno3, ...]
             [img3_anno1, ...]
                 ...]
Make sure you are storing numbers in float.
```

3.2 getitem

Getitem is a commonly seen function in pytorch based dataloader. It aims to get/organize/process the training data from the pool prepared for each iteration.

Read carefully:

```
In __getitem__, complete the annotation reorganization. Each "annotation"
should be in the following format:
np.array(x1, y1, x2, y2, L0_x, L0_y, L1_x, L1_y, ..., L4_x, L4_y, M)
Where (x2, y2) is the bottom-right corner of the bounding box,
```

```
L stands for the landmark points.
M is an indicator showing whether landmark points are
available,
Yes for 1 and No for -1. The "annotation" is in shape (1, 15)
```

4 Anchor(anchor.py)

Anchor has been an important concept for detection since 2015 (Faster RCNN).

We define "Anchor", in this homework, as "all reference boxes located in every feature point"

Understand the code, and answer the following

4.1 Name all hyperparameters that correspond to the number of anchors. I.e., which hyperparameters in config will affect the number of anchors? Briefly describe their physical meanings in your words

Solution

The two hyper parameters to talk about in that regard are **min_sizes** and **steps**. The steps parameter partitions the image into several x, y coordinates each **steps** distance away from each other. Since several steps parameters may be passed in, (for example, $8, 12$), we will first partition the image into a grid of points 8 pixels away from each other and then 12 pixels away from each other.

For each partitioned grid, each point of the grid will be the top left corner of out anchor box. A natural question is on how big the anchor box should be. This is what the **min_sizes** parameter will provide us. Since it contains an array of tuples (e.g. $[(2, 4), (4, 8)]$), from a single point it will create bounding boxes of all those sizes.

So if we were to start with a point (x, y), we shall receive bounding boxes of $(x, y, 2, 4)$ and $(x, y, 4, 8)$.

4.2 If we enlarge the width and height of the image by x, how would the number of anchors change?

Solution

We should see x^2 times the number of anchor boxes. This is because we did not change the size of the steps or feature maps. The sizes of the feature map increases proportionally and since we are doing a range over the dimensions, the number of anchor boxes will increase as well

4.3 For anchors on different feature pyramids, are they the same size? Explain your observations and explain why it is designed in such a way.

Solution

They will be different, each anchor will be of different sizes to detect objects of different sizes and with different levels of detail

5 Calculate the Loss

Anchor plays a key role in anchor-based object detection for deciding which feature point on the feature pyramid should be responsible for learning a certain ground-truth. Let's do a quick review:

On your multi-level feature pyramid, each pixel is a feature point. Each feature point is mapped to one or multiple anchors depending on the detector's setting. We use an anchor to compare with ground-truth via "jaccard index", i.e., Intersection over Union. If the IoU is larger than a threshold (0.2), then the feature point corresponds to that anchor would be assign to learn the ground-truth, the learning includes positive/negative classification, bounding box regression, and landmark. If an anchor matches zero ground-truth, the corresponding feature point needs to be classified as negative.

5.1 In utils/boxutils.py, complete jaccard function.

5.2 Explain the following code in your words:

In "match", what is the following code used for?

```
best_prior_overlap, best_prior_idx = overlaps.max(1, keepdim=True)
```

Solution

This returns the tensor and the index of the tensor where the resulting IOU between the ground truth and prior was the highest. Since there are num_obj independent ground truth boxes, for each ground truth box, we find which prior gave us the greatest IOU. The overlaps should be of dimensions num_obj by num_prior, and after this, the return values should be of shape num_obj. My doing overlaps.max(1...) , we are finding which prior best fit with each of the ground truth box. The tensor itself represent IOU (i.e. jaccard value)

In "match", what is the following code used for?

```
valid_gt_idx = best_prior_overlap[:, 0] >= 0.2
```

Solution

This gives us the index of the priors that have a overlap/IOU of at least 0.2. Since this has not breached the threshold, we do not use the feature point to learn that ground truth

In "match", what is the following code used for?

```
best_truth_overlap, best_truth_idx = overlaps.max(0, keepdim=True)
```

Solution

For each prior, this find the best indices and overlaps for each of the ground truth boxes. I.e. gives the best GT for each prior

6 NMS

Non-maximum suppression is a must-ask question for those interviews seeking computer vision engineer positions. Prior to 2020, it is an essential component for object detection and many other tasks. Write function of NMS in nms.py

```
input: dets: ndarray (nx5), where n is number of predictions,
and each prediction is in (x1, y1, x2, u2, confidence)
```

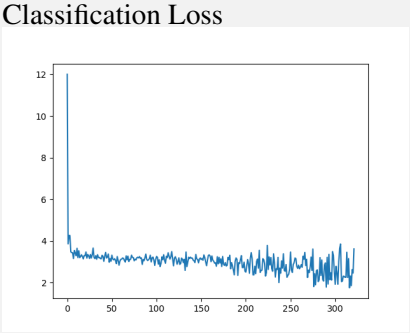
```
        thres: float, which is IOU threshold.
output: list, the index of the predictions in dets, which passed NMS.
```

7 Training

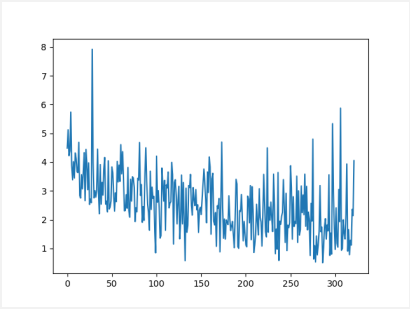
Train your detector with 1 epoch, and plot each loss curve, the plotting step is per 20 iterations.

Solution

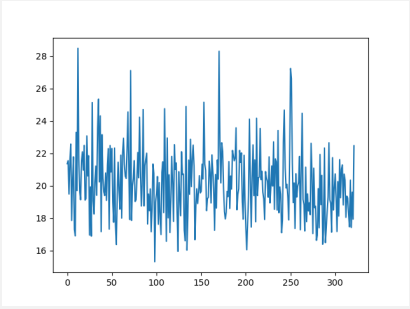
Classification Loss



Location Loss



Landmark Loss



Use any methods you feel comfortable with to remove the landmark supervision. Explain your method. And what do you expect the face detector would perform (better or worse) after the removal? Explain your answer.

Solution

I multiplied the landmark supervision code by 0 so that section of the model is no longer being trained. To verify, I checked the loss curves of the landmark head and saw that it did not decrease.

I expect the face detector to perform about equal if not slightly better. They should be relatively independent since all the heads are separated and are not affecting each other too much. But since landmark is no longer being trained, I would expect the others (detector and classifier) to perform much better since they won't be fighting each other's changes.

8 Testing

Load the checkpoint we provided, it is the same as yours but trained with larger batchsize and with 180 epochs. Plot the first 4 results of your detector, with confidence, face bounding box, and landmarks.

Solution

Enjoy your face detector and feel free to deploy it for your home security.