CS1390: Introduction to Machine Learning

Big Brother isn't Watching You: Crowd Detection and Obfuscation

Soham De, Tanvi Roy

December 12, 2021

1 Introduction

Our project aims to develop a pipeline for Real-Time Image obfuscation from Video Feeds, using state-of-the-art Object Detection Models. We thus explore the possibility of using such a pipeline to implement Crowd Detection and Obfuscation. We accomplish the same by conducting various experiments using a pre-trained Single Shot Multibox Detector model and Facebook's RetinaNet dense detector. Additionally, we introduce a custom curated and annotated 'Ashokan Objects in the Wild' Dataset on objects and places native to Ashoka University. We hence put our models to the test in real Ashokan crowds and conducted various experiments. We provide extensive analyses for the performance of our model in these multiple different scenarios. While we conclude that true Crowd Detection is not feasible with the methods we explore, we believe our experiments have given us valuable insights on the subject.

2 Literature Review

Our preliminary literature review on object detection began with looking at One Stage Object Detection versus Multi Stage Object Detection.

One Stage Object Detectors which directly, from the given image, classify or find bounding boxes are far more faster and easier to use. This is in comparison to Two Stage Object Detectors which extract and review the object proposals, resulting in better accuracy. For the purposes of our project, we chose to experiment with One Stage detectors.

2.1 Faster R-CNN

Initially proposed by Ross Girshick in 2014, R-CNN (Regions with CNN features)[2] obtained a mean Average Precision (mAP) of 53.3%.[2] However, R-CNNs would lack efficiency as the convolution networks were applied on every possible object proposal.

Following this, in 2015 Girshick proposed a Fast Region-based Convolutional Network method (Fast R-CNN)[1] for object detection. Fast R-CNN introduced region of interest (RoI) pooling and the training of all network layers to be processed in a single stage, which greatly improved its efficiency. Compared to SPPnet[3], Fast R-CNN trained VGG16[7] 3x faster, tested 10x faster, and

proved to be more accurate.[1]

In 2016, Ren. et al introduced Faster R-CNN[6] which enabled nearly cost-free region proposals by sharing full-image convolutional features with the detection network. Faster R-CNN proposed a new Region Proposal Network (RPN) that simultaneously predicts object bounds and objectness scores at each position.

2.2 Single Shot Multibox Detector (SSD)

Introduced in 2016, SSD[5] discretizes the output space of bounding boxes into a set of default boxes over different aspect ratios and scales per feature map location. SSD combines inputs from feature maps with various resolutions allowing it to predict objects at different convolutional scales. The architecture of SSD achieves 72.1% mAP on VOC2007 test at 58 FPS on a Nvidia Titan X and for 500×500 input.[5]

2.3 RetinaNet

Prior to RetinaNet[4], the fastest object detectors were primarily two stage detectors stemming from the R-CNN family of models. In 2018, Facebook's RetinaNet matched the speed of popular one stage detectors, while surpassing the accuracy of two stage detectors. A primary reason attributed to the lower accuracy observed in fine one stage detectors is the extreme foreground-background class imbalance during training. RetinaNet solves this problem by reshaping the standard cross entropy loss such that it down-weights the loss assigned to well-classified examples.[4] It does so by introducing a Focal Loss.

3 Project Description

Our goals for the project were as follows:

1. A comprehensive **Literature Review** on Object Detection Architectures
2. A novel Ashoka Object Detection Dataset (**Ashokan Objects in the Wild**)
3. **Fine-Tuned** Object Detection Network on Ashokan Objects in the Wild
4. A **Crowd-Sensing** and **Image-Processing Pipeline for obfuscating Video Feed**

In our original project pitch, we had the additional final goal of experimenting with YOLO as a region proposer for Mask-RCNN. However, we have decided against going ahead with that following Prof. Banerjee's advice to keep the region proposals as generous as possible.

4 Dataset Specifications

Ashokan Objects in the Wild is a custom dataset that we manually curated and annotated. Pictures were taken on a 12MP camera, and downsized to a resolution of (853 x 640). This is sufficient enough to run inference on full-HD video feed. This dataset has 10 classes, with 25 images each. All images are manually annotated with bounding boxes in the format described below. This dataset is available here.

MSCOCO 2017 is a standard dataset for object detection models. It has over 330,000 unique images with 80 categories. It has over 1.5 million object instances annotated. The large networks we used for our project ([5] and [4]) were all pre-trained on MSCOCO.

5 Methods

Models and Architectures used

1. **Faster RCNN**[6]: After running preliminary inference on pre-trained Faster RCNN, even on a GPU we experienced inference times of 13.6 seconds on 640x 640 images. Since our objective was real-time video obfuscation, this model was infeasible for our use.

2. **SSD**[5]: We used a SSD (with a MobileNet V2 backend), pre-trained on MSCOCO for preliminary inference. The results were satisfactory although accuracy wasn't upto the mark. We discuss some of these issues in the Observations in the report. This motivated us to Fine-Tune a detection network using our own dataset.

3. **RetinaNet**[4]: We used a pre-trained RetinaNet and fine-tuned it on a custom dataset of Ashokan objects. This was as fast as SSD (inference times of about 0.34s on a GPU per frame) while at the same time,

6 Implementation Details

6.1 Notebooks and Custom Datasets

- **Video Obfuscation Pipeline** - Google Colab Notebook
- **Few Shot Object Detection** - Google Colab Notebook
- **Annotations Pipeline** - Google Colab Notebook
- **Ashoka Objects in the Wild Dataset** - Google Drive

6.2 Image Processing Pipeline

For the object detection, we use a bounding box to describe the spatial location of the object. The bounding box is rectangular and we describe it with x, y, w, and h. We remark that this is different from the traditional VOC formats for annotations, although they are inter-convertible.

The Image Obfuscation Pipeline, as illustrated in 1 was implemented from scratch. It goes as follows:

1. Bounding Box coordinates and classes are obtained from the detection model.

2. Only classes Person, Man, and Woman are considered. Bounding boxes corresponding to theses are fed to the `draw_bounding_box_on_image` function, which was then modified to blur the box. This function is listed below in the report.

3. This process is repeated for every such box

4. The final count is added on the image after the return of the aforementioned function.

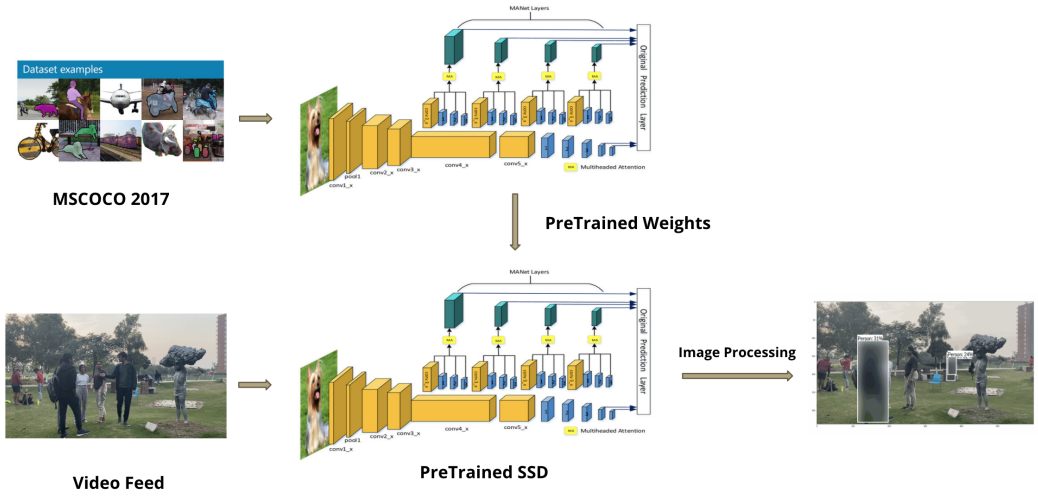


Figure 1: Image Obfuscation Pipeline in our Project

```python
from PIL import Image, ImageFilter

def draw_bounding_box_on_image(image,
                               ymin,
                               xmin,
                               ymax,
                               xmax,
                               color,
                               font,
                               thickness=2,
                               display_str_list=()):

  draw = ImageDraw.Draw(image)
  im_width, im_height = image.size
  (left, right, top, bottom) = (xmin * im_width,
                                xmax * im_width,
                                ymin * im_height,
                                ymax * im_height)

  box = (int(left), int(top), int(right), int(bottom))
  crop_img = image.crop(box) # Crops the section within the bounding box
  blur_image = crop_img.filter(ImageFilter.GaussianBlur(radius=5)) # Blurs the
    cropped section
  image.paste(blur_image, box) # Pastes the blurred section back
```

Listing 1: Obfuscating People in a Frame

6.3 RetinaNet Fine-Tuning

For the final experiment, we used our custom dataset to Fine-Tune the RetinaNet. The RetinaNet is pre-trained on the MSCOCO dataset, which has 80 classes. However, all the classes in our dataset are unique to Ashoka, and therefore not present in MSCOCO. The RetinaNet was trained for 40 epochs, the details of which are available in 2. We also present a single frame post inference, to illustrate the succesful training of our model in 3

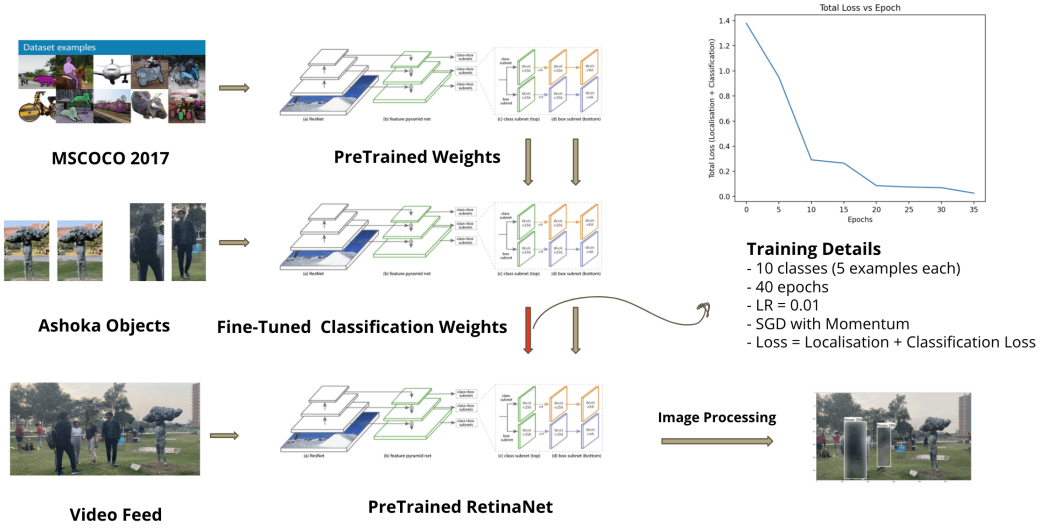


Figure 2: RetinaNet FineTuning Pipeline in our Project

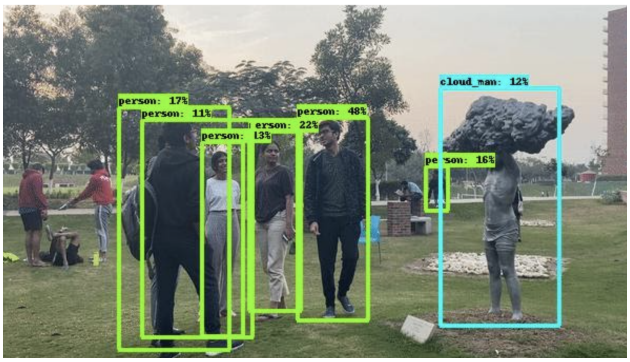


Figure 3: RetinaNet Inference with novel classes

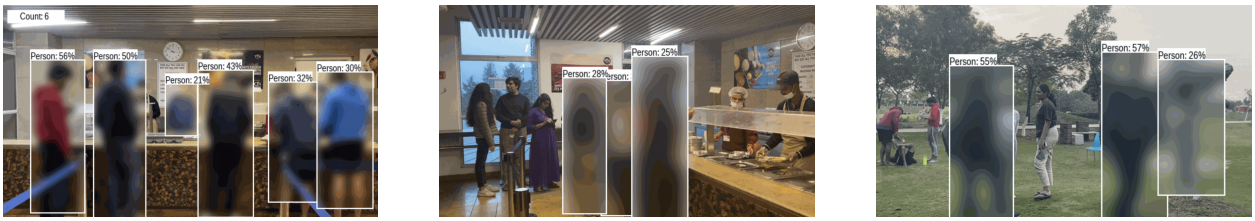
7 Observations

We extensively experimented with object detection inferences on real-life video feed from the Ashoka campus and have made the following observations:

- **Scene 1:** This is the ideal scenario, when there is no overlapping objects and there is good background-foreground contrast
- **Scene 2:** This is when there is a foreground object blocking an object - this leads to a mis-classification (clothing instead of person)
- **Scene 3:** Poor background separation. We overcome this by fine-tuning a RetinaNet on custom dataset, as demonstrated earlier.

8 Conclusion and Future Work

- Naive Object Detection Algorithms on raw video feed is not good enough for crowd detection. More over, crowd detection is not equivalent to the count of the number of people in the frame



(a) Scene 1: Ideal Case (b) Scene 2: Foreground Object (c) Scene 3: Poor Contrast

Figure 4: SSD Inference Failure Instances

- it depends on the context of the scene, which isn't being revealed by the detection model.

- This model is more suited for person detecting as opposed to crowd detection
- We may use an RCNN model on top of the bounding boxes to get a segmentation, which can then be used for more precise blurring/ obfuscation of the video feed.

References

[1] Ross Girshick. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV].

[2] Ross Girshick et al. *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2014. arXiv: 1311.2524 [cs.CV].

[3] Kaiming He et al. "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition". In: *Lecture Notes in Computer Science* (2014), pp. 346–361. ISSN: 1611-3349. DOI: 10.1007/978-3-319-10578-9_23. URL: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[4] Tsung-Yi Lin et al. *Focal Loss for Dense Object Detection*. 2018. arXiv: 1708.02002 [cs.CV].

[5] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[6] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV].

[7] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

Listings