Detecting Tear Gas Canisters With Limited Training Data

Ashwin D'Cruz [*]
ashwin.d.dcruz@gmail.com

Christopher Tegho [*]
christegho@gmail.com

Sean Greaves [*]
greaves.sean@btinternet.com

Lachlan Kermode
lk@forensic-architecture.org
Forensic Architecture
London, UK

Abstract

Human rights investigations often entail triaging large volumes of open source images and video in order to find moments that are relevant to a given investigation and warrant further inspection. Searching for instances of tear gas usage online manually is laborious and time-consuming. In this paper, we study various object detection models for their potential use in the discovery and identification of tear gas canisters for human rights monitors. CNN based object detection typically requires large volumes of training data, and prior to our work, an appropriate dataset of tear gas canisters did not exist. We benchmark methods for training object detectors using limited labelled data: we fine-tune different object detection models on the limited labelled data and compare performance to a few shot detector and augmentation strategies using synthetic data. We provide a dataset for evaluating and training tear gas canister detectors and indicate how such detectors can be deployed in real-world contexts for investigating human rights violations. Our experiments show that various techniques can improve results, including fine-tuning state of the art detectors, using few shot detectors, and including synthetic data as part of the training set.

1. Introduction

Human rights investigations often entail triaging large volumes of open source images and video in order to find moments that are relevant to a given investigation and warrant further inspection. Searching for instances of tear gas usage online manually is laborious and time-consuming. Techniques that automate either the discovery, the identification, or the archiving of online material is thus hugely beneficial for human rights monitors. One heuristic that can identify possible instances where tear gas is or has recently been use is to detect the presence of tear gas canisters as they appear within images. Automatic detector can be applied either directly to images available online, or to videos by first converting them to a series of frames and then applying the detector to each frame.

Convolutional neural networks (CNNs) have been successfully used for object detection, but it typically requires large volumes of annotated training data. A large annotated dataset of tear gas canisters currently does not exist. Our goal is thus to train detectors that generalize well to real-world cases given the limited available training data for tear gas canisters.

To that end, our first contribution is a dataset of tear gas canisters consisting of a small number of real images depicting tear gas canisters. We have curated this dataset to cover as wide a range of scenes and settings as was possible using images in the wild.

Our second contribution consists in the exploration of data augmentation methods to produce additional training examples of tear gas canisters. In [21], the authors improved the performance of a canister detector by training with real and synthetic data. In our work, we adopt this approach, and build upon it by testing classifiers for a wider range of tear gas canister types using synthetic data. Our third contribution is to investigate the performance of standard pre-trained object detectors that are fine-tuned on our dataset of both synthetic and/or real tear gas canisters. We show that weights learned from other real life images do help for the detection of tear gas canisters. In our fourth contribution, we try to address the limited data issue through the use of fewshot learning, that is the problem of learning from few labeled training examples. We specifically apply the few shot learning method via reweighting features by [20].

Finally, we carry out ablation studies to investigate the

[*]equal contribution

impact of different classes on the dataset. Specifically, we examine how useful real images are compared to synthetic images. In the case of the few-shot detector, we also investigate the use of images containing objects that are similar in appearance to tear gas canisters but are in fact not tear gas canisters.

Our experiments show that fine-tuned state of the art detectors perform as well as the few shot detector, and including synthetic data can improve results for certain detectors. Additionally, fine-tuning the few-shot detector benefits from including classes for canister-like objects.

2. Related Work

2.1. Object Detection

Object detection is the task of localizing and classifying objects of interest within an image. Concretely, an object detection model would take as input an image and output the coordinates of bounding boxes for each object of interest as well as the category or class that each object belongs to.

Prior to deep learning, an object detection pipeline consisted of identifying regions within an image to consider, extracting relevant features using techniques such as SIFT [26] and Haar [41], and then feeding these features into a decision algorithm such as SVMs [9] and Deformable Parts Models [14]. While models that used this pipeline were able to detect a variety of objects, they were limited by the richness of the features. These features didn't generalize well to different settings and backgrounds from the domain they were developed in. Additionally, features well suited for localization may not have worked as well for classification so a combination of features might have been required to achieve good results.

Convolutional neural networks (CNNs) [23] were able to overcome several of these limitations. Like all deep learning techniques, these models could learn a feature representation that generalized across a range of image backgrounds. The ability to stack multiple layers also meant that a rich hierarchy of features could be learned. Furthermore, CNNs could be set up to optimize solving multiple sub-tasks simultaneously so a single network and a single set of features could be utilized for both localization and classification.

Within the domain of CNNs for object detection, there are two main frameworks. The first is the region proposal based framework including models such as Faster R-CNN [34]. Region proposal methods follow this pipeline roughly: generate regions of interest from an image, extract features for these regions, and predict bounding box coordinate and class confidence scores for objects of interest within each region. Models of this family have good performance with regards to localization and classification. However, the different parts of the pipeline in these models can contribute to a large training and inference overhead. The second framework is the single-shot detector framework. Models within this framework go directly from pixels to bounding boxes and include works such as YOLO [31] and DSOD [35]. While there are slight variations, models within this family work by starting with a set of anchor boxes and then regress deviations to these boxes such that the resulting boxes encompass objects of interest. In parallel to this, the model also assigns class confidence scores to each of these boxes with a special 'background' class for boxes that do not contain any objects of interest. As there are fewer moving parts in the pipelines of these models, they tend to be faster to train and run inference with.

For a more detailed review of deep learning based object detectors, we refer the reader to [43].

2.2. Few Shot Object Detection

Few-shot learning refers to learning from just a few training examples. One approach is to train a recurrent model [29] to take in the dataset sequentially and then process new inputs from the task. The meta-learner uses gradient descent, whereas the learner simply rolls out the recurrent network. Another approach concerns metric learning, where a learner compares datapoints in a learned metric space in which learning is particularly efficient. This metric space can be learned using VAEs as in DARLA [17], using Siamese networks [22], or neural networks with external memories [40]. Another approach treats model adaptation as optimization [29, 15]. MAML [15] encourages a global model to learn a representation that can quickly be adapted to a new task via few gradient steps and training examples. During adaptation, the parameters of the model are close to the optimal parameters for a new task. More recent work that specifically focus on few-shot object detection learns the matching metric between image pairs based on the Faster R-CNN framework equipped with an attention RPN and a multi-relation detector trained using a contrastive training strategy [13]. Our paper uses a different approach that utilizes a meta feature learner and a reweighting module within a one-stage detection architecture, to quickly adapt to novel classes [20]. The feature learner extracts meta features that are generalizable to detect novel object classes, using training data from base classes with sufficient samples.

3. Dataset

We present a new dataset of images containing tear gas canisters to encourage the deployment of computer vision classifiers in the aim to help human rights investigations and monitoring at large. The dataset is publicly available here.

3.1. Image Collections

The dataset comprises four different collections of images:

1. Images that contain one or more canisters, or part of canisters of type Triple Chaser, and of type 34-60mm. These images are collected from publicly available sources, such as news articles and publicly accessible videos. The canisters vary in shape, view angle, size, colors, and states (used or unused canisters). The images represent various scenes in which a canister can appear in: scenes of street violence, protests, bombed areas, abandoned or torn buildings, natural landscapes (where used canisters are left behind, or disposed).

2. Images that contain objects that are not tear gas canisters but resemble them. These objects are bottles, bins, cans and other metallic and cylindrical objects.

3. Images that contain scenes in which canisters have appeared in, but that do not contain canisters or objects that resemble canisters.

4. Synthetic images of canisters, to augment the training set, given the limited amount of real images with canisters we have labelled. These synthetic images recreate the visual contexts in which the object of interest have been previously documented [21], and are generated with Unity [3] and Unreal Engine [27]. The 3D model of the Triple Chaser is inserted into a scene that randomly varies the render camera's position, as well as modifying settings such as exposure, focal length, and depth of field, between frames. After rendering each frame, post-effects such as LUTs for color correction, variations in the film grain, and different vignettes are applied [21].

Examples of these images can been seen in Figure 1.

Split	Nb.	Subset	Subset Nb.
Training	1434	Triple Chaser	219
		Canister-like Objects	16
		Synthetic Triple Chaser	1199
Validation	125	Triple Chaser	45
		Canister-like Objects	80
Testing	268	37-40mm Canisters	165
		No Canisters	103

Table 1: Summary of data splits.

3.2. Dataset Format

When annotating the locations of target canisters, we followed the PASCAL VOC [12] format, and the format for YOLO [32] based detectors.

The PNG masks for image segmentation and the corresponding bitmap objects are available through Supervise.ly [1], which we used for labelling our data, but are currently not available in this current version of the dataset.

4. Methods

4.1. YOLOv5

One of the object detection models we consider is YOLOv5 [38]. YOLO models [32] frame object detection as a regression problem and are trained end-to-end, which make them fast to train and run inference with, while providing competitive performance. The main contribution of YOLOv5 is to port YOLO from the Darknet neural network framework [30] to PyTorch [28], which allows for the use of 16 bit floating point computations instead of 32 bits, significantly reducing inference time.

YOLO has three main components: a backbone network that generates image features at different granularity, a neck that mixes and combines these different features, and a head that consumes these mixed features to predict boxes and classes. The loss consists of a combination of generalized intersection over union (GIoU), a class loss, and an object loss. Of note, a bottleneck based on the Cross Stage Partial Network (CSPNet) [42] is used for the backbone of YOLOv5. CSPNet is designed to minimize the amount of duplicated gradient computations observed in networks based on DenseNets [19]. This allows for faster training times and models can be made smaller with less impact to performance.

The family of YOLO models treats detection as a regression problem. Specifically, each anchor box has an x and y coordinate, a width and a height. The model predicts how much change is required to each of these attributes so that the box encompasses an object of interest if there is one close to the anchor box. In this implementation, the anchor boxes are learned initially based on the distribution of boxes in the training dataset using k-means and genetic learning algorithms. This is helpful for our use case as tear gas canisters have a limited range of appearances. By using anchor boxes that are more suited to the dimensions of tear gas canisters, the model can learn more quickly how to predict correct bounding boxes for tear gas canisters.

Aside from the use of CSPNet and learned anchor boxes, YOLOv5 also utilizes certain data augmentation techniques that are useful for our application. The most noteworthy data augmentation is the mosaic data augmentation, first introduced in YOLOv4 [8]. For this augmentation technique, four different images are combined into a single one by tiling them using different ratios. Similarly to random cropping, this helps the network better deal with issues of occlusion which is beneficial for our application as a lot of tear gas canisters are often occluded. Additionally, this data augmentation method also combines objects of different classes into a single image. This is useful for us as real

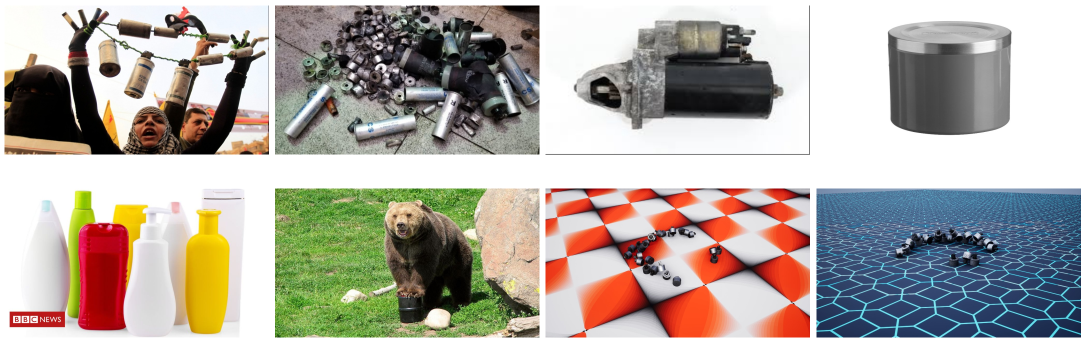
Figure 1: Example images from the dataset. In the top row, two images on the left that contain one on more canisters. The other two images contain objects that look like canisters but are not canisters. On the bottom row, the images on the left do no contain any objects that look like canisters. The remaining images are synthetic images of canisters.

world images may sometimes contain a variety of tear gas canisters. Finally, by tiling different images together, the background of the image becomes more varied. This could be viewed as a form of domain randomisation which previous work [21] has shown helps for this application.

4.2. RetinaNet

RetinaNet is a popular single-stage detector that demonstrates comparable accuracy to two-stage detectors without sacrificing the fast inference speed characteristic of other single-stage detectors [24]. Prior to RetinaNet, the accuracy of single-shot detectors trailed behind the accuracy of two-stage detectors for many tasks. It has been shown that single-stage detectors suffer from extreme foreground-background class imbalance during training resulting in easily classified negatives making up the majority of the loss. This background-foreground imbalance during training has been identified as the main factor contributing towards lower accuracy of single-stage detectors. This observation is addressed within RetinaNet primarily through a novel loss function termed 'focal loss' that focuses training on sparse hard examples. Focal loss therefore mitigates the contribution of the vastly greater number of easy negative samples from overwhelming the detector during training leading to improvements in accuracy.

RetinaNet is a simple detector comprised of a ResNet-FPN backbone network and two parallel subnetworks for classification and bounding box regression. The feature pyramid network (FPN) is an important architectural design choice as it allows for multi-scale object classification.

4.3. Faster R-CNN

Faster RCNN is a popular two-stage detector and was the first end-to-end trainable, near-realtime deep learning detector [25]. Faster RCNN significantly improves the inference speed of Fast RCNN by replacing the costly region proposal algorithm used to generate object location proposals with a region proposal network, a small CNN that enables almost cost-free proposal generation. The architecture is comprised of an RPN and Fast R-CNN within a single network.

4.4. Few-Shot Detector

Another approach to deal with lack of training data is meta-learning, an increasingly popular solution for the task of few shot learning. Meta-learning algorithms train a learner, which is a parameterized function that maps labeled training sets to classifiers [10]. Meta-learners are trained by sampling a collection of learning tasks in a few-shot setting from base classes with lots of available training data [40, 16, 37]. The model is optimized to perform well over these few shot tasks so it is able to tackle the recognition of unseen or novel classes from few training examples.

We specifically apply the few shot learning method via reweighting features by [20]. In this work, meta features are extracted from a meta feature learner $\mathcal{D}$. The backbone of YOLOv2 [33] with its anchor settings is used for implementing the meta feature extractor and the detection prediction module $\mathcal{P}$. The extracted features are generalizable and are used to detect objects from both novel and base classes. The meta learner uses few support examples to identify features that are important and discriminative for detecting novel classes.

A reweighting module $\mathcal{M}$ provides a global vector for each class, which indicates the importance or relevance of the meta features for detecting the corresponding objects. The meta features are adapted to be suitable for the novel object detection by reweighting them using the feature vectors [20]. A detection prediction layer then predicts the classes scores and the coordinates for the detection bounding boxes using the adapted meta-features.

Formally, given an input query image $\mathcal{I}$, meta features $\mathcal{F} \in \mathbb{R}^{w \times h \times m}$ are extracted by $\mathcal{D}$ with $\mathcal{F} = \mathcal{D}(\mathcal{I})$. The reweighting model $\mathcal{M}$ takes as input a given support image

$\mathcal{I}_i$ and its associated bounding box annotation, and target class to detect, denoted as $\mathcal{M}_i$ for class i, $i = 1, ..., N$; and outputs class-specific reweighting coefficients w_i. Class-specific features $\mathcal{F}_i$ for novel class i are then obtained by:

$$\mathcal{F}_i = \mathcal{F} \bigotimes w_i, i = 1, ..., N \quad (1)$$

where $\bigotimes$ denotes channel-wise multiplication, implemented through 1×1 depth-wise convolution. The prediction module $\mathcal{P}$ outputs an objectness score o, bounding box location offsets (x, y, h, w), and a classification score c_i for each of the predefined anchors:

$$\{o_i, x_i, y_i, h_i, w_i, c_i\} = \mathcal{P}(\mathcal{F}_i), i = 1, ..., N. \quad (2)$$

All three modules, the meta features, the reweighting, and the prediction modules are trained end to end, with abundant training images from bases classes in a first stage, then with k training images from novel and base classes, where k is typically a small number (1 to 10), though in this paper, we experiment with a larger k as well.

5. Experiments

5.1. Experimental Setup

For YOLOv5, we use an existing PyTorch implementation of YOLOv5[38]. This repository provides different model sizes offering a trade-off between inference speed and accuracy. For our work, we consider the small, medium, and large models. The differences between these models are listed in the original repository [38]. For our experiments, we use checkpoints of models that were previously trained on the 2017 COCO dataset. All YOLOv5 models are fine-tuned with a batch size of 16 for 50 epochs. The best checkpoint is chosen based on the validation loss. We experiment with fine-tuning all three model sizes on both synthetic images and real images. All three model sizes are trained on the training subset with real images of canisters. In addition, we experiment with fine-tuning the best performing model with regards to mAP synthetic images only, and on real images only.

Faster R-CNN and RetinaNet are implemented in PyTorch within [39] and trained with ResNet-50FPN backbone networks. When fine-tuning models, images are reshaped such that the larger dimension is equal to 1000 pixels whilst maintaining aspect ratio. Suitable learning rates are set uniquely for each training run and chosen through conducting a learning rate range test as described in [36]. The test involves conducting a mock training run for several epochs during which the learning rate is increased linearly from low to high. Analysing how loss varies with learning rate throughout training allows for a learning rate to be estimated that could result in a large initial decrease in loss without training diverging. This test is implemented within a learning rate finder method in the fastai library which is used to train both models [18]. Models are fine tuned for 10 epochs with a batch size of 6 on a single NVIDIA Quatro P5000. Fine tuning involves training the randomly initialised layers for one epoch and then unfreezing the remaining layers and training all layers for the remaining epochs.

For the few shot detector via feature reweighting, we use the PyTorch implementation by the original authors [20]. We use the same base model checkpoint which was trained on the VOC 2007 [12] and the VOC 2012 [11] datasets. For the base model, we use the same 15 object categories from COCO used in [20]. We treat the object label for canisters as a novel class. The fewshot finetuning is done using a batch size of 20, for 20 epochs, on one GTX 1080. We also experiment settings where we finetune the detector on canister-like object classes ”bin”, ”can” and ”cylinder” which are treated as novel classes.

5.2. Performance Metrics

To evaluate object detectors on tear-gas canisters, we report mAP50 which reflects precision and recall on the bounding box level for a given evaluation set. However, in production, we are only interesting in flagging frames within a video which might contain a tear-gas canister. It is enough for the detector to detect one of many tear gas canisters in a given image, regardless of whether the bounding box is accurate or not. To reflect this, we report two additional performance measures on an image level: recall to measure the number of missed frames which contain tear gas canisters, and reduction to measure the number of frames which are wrongly flagged as having a tear gas canister. It is more important to avoid missing true positives over false positives so we combine both reduction and recall with a F2 score as follows:

$$\text{recall} = \text{tp}/(\text{tp} + \text{fn}) \quad (3)$$

$$\text{reduction} = \text{tn}/(\text{tn} + \text{fp}) \quad (4)$$

$$F_\beta = (1 + \beta^2) \cdot \frac{\text{reduction} \cdot \text{recall}}{(\beta^2 \cdot \text{reduction}) + \text{recall}}, \beta = 2 \quad (5)$$

We finetune hyperparameters based on mAP for the validation set and we set the decision thresholds based on the best F2 score obtained on the validation set.

6. Results

Results for all experiments carried for this paper are summarised in Table 2 for the validation set and Table 3 for the test set.

For both the validation and the test sets, we did not observe a significant improvement in mAP and F2 when using synthetic data for finetuning models compared to the

Method	Real Nb.	Synthetic Nb.	”Similar”	mAP	Recall	Reduction	F2 Score
fewshot	219	N	Y	18.28	0.7609	0.2785	0.5651
fewshot	219	N	N	23.04	0.6304	0.4625	0.5878
fewshot	219	219	Y	**24.76**	0.7609	0.4	0.6446
fewshot	219	219	N	14.38	0.6739	0.3625	0.5751
fewshot	10	N	Y	12.98	0.7174	0.4068	0.6223
fewshot	10	N	N	10.66	0.75	0.3239	0.5938
fewshot	10	10	Y	21.42	0.75	0.5286	**0.692**
fewshot	10	10	N	17.15	0.6591	0.4675	0.6092
yolov5-small	219	1199	N	16.12	0.619	0.524	0.597
yolov5-medium	219	1199	N	17.16	0.595	0.512	0.576
yolov5-large	219	1199	N	19.21	0.548	0.524	0.543
yolov5-large	219	N	N	**20.16**	0.690	0.369	0.588
yolov5-large	0	1199	N	15.76	0.810	0.429	**0.687**
faster r-cnn	219	N	N	40.91	0.870	0.215	**0.537**
faster r-cnn	219	1199	N	50.39	0.913	0.133	0.377
faster r-cnn	0	1199	N	**50.73**	0.913	0.2	0.533
retinanet	219	N	N	47.61	0.587	0.6	0.590
retinanet	219	1199	N	51.38	0.717	0.4	0.620
retinanet	0	1199	N	**55.56**	0.739	0.5	**0.675**

Table 2: Detection performance (mAP and F2-Score) on the validation set. The validation set include images which have tear gas canisters, ones which have objects resembling canisters but are not canisters, and ones which do not have any canister or object resembling canisters. The ”Similar” column indicates whether the detector’s training set includes objects that resemble tear-gas canisters. Given the low number of images which contain these objects, we only included these classes during training for some few shot detectors.

same settings without including the synthetic data, and for Faster R-CNN and RetinaNet, using synthetic data resulted in worse performance.

For the fewshot detector, the best mAP and F2 scores are obtained when using 10 images with canisters, 10 synthetic images and images with canister-like objects for fine-tuning. Slightly lower F2 and higher mAP are obtained when we use all 219 training images of canisters and 219 synthetic images. Using images with canister-like classes for fine-tuning the fewshot detector has consistently improved performance for both the validation and the test sets. This improved performance is attributed to obtaining vectors (during the fewshot fine-tuning stage) which are much further in the feature space which allows better classification and differentiation between canisters and other objects whether they look like canisters or not.

For YOLOv5, when training on the dataset containing both real and synthetic images, we notice that mAP increases as the model size increases which is expected since with an increased model capacity, we would expect better performance. As the large YOLOv5 model has the best performance, we also use this model to investigate the contribution of the real and synthetic data. For mAP, it appears that the real data is more important than the synthetic data as the model trained with just synthetic data has the worst mAP of the yolov5 models by a large margin. Our experiment with using just the real image dataset shows that even though the model was fine-tuned on just 219 images, the weights learned from COCO2017 are suitable for the task at hand. The F2 scores don’t show as clear a trend with regards to model capacity and data effects. We can attribute this noise to the fact that the models were optimized for mAP and not F2. While the mAP results from the YOLOv5 model are not as high as the results from Faster R-CNN and RetinaNet, they are roughly on par with the fewshot detector models. Additionally, their more streamlined structure means that it is faster to run inference with these models compared to the others.

Faster R-CNN exhibits high mAP and recall across validation and test sets. Highest mAP on the test set is achieved through training Faster R-CNN on only real images demonstrating the generally higher accuracy of two-stage detectors in comparison to single-stage detectors.

RetinaNet achieves high mAP on both validation and test sets demonstrating similar mAP to Faster R-CNN. RetinaNet achieves highest reduction and second highest mAP of all models on the test set which could be attributed to the contribution of focal loss in focusing the training on hard

Method	Real Nb.	Synthetic Nb.	"Similar"	mAP	Recall	Reduction	F2 Score
fewshot	219	N	Y	29.16	0.7484	0.807	0.7594
fewshot	219	N	N	**62.94**	0.6516	0.9035	0.6901
fewshot	219	219	Y	50.84	0.6516	0.7895	0.6752
fewshot	219	219	N	35.96	0.6774	0.6228	0.6657
fewshot	10	N	Y	43.72	0.7568	0.6901	0.7424
fewshot	10	N	N	50.04	0.7324	0.7679	0.7392
fewshot	10	10	Y	50.81	0.7917	0.7258	**0.7776**
fewshot	10	10	N	58.85	0.7078	0.8214	0.7279
yolov5-small	219	1199	N	37.04	0.523	0.728	0.554
yolov5-medium	219	1199	N	46.45	0.852	0.605	0.787
yolov5-large	219	1199	N	**55.43**	0.89	0.588	**0.807**
yolov5-large	219	N	N	55.29	0.606	0.860	0.644
yolov5-large	0	1199	N	13.24	0.587	0.640	0.597
faster r-cnn	219	N	N	**66.73**	0.806	0.816	**0.808**
faster r-cnn	219	1199	N	55.60	0.852	0.5	0.747
faster r-cnn	0	1199	N	13.22	0.852	0.193	0.506
retinanet	219	N	N	**66.10**	0.516	0.940	0.567
retinanet	219	1199	N	51.48	0.581	0.807	0.615
retinanet	0	1199	N	27.41	0.710	0.482	**0.649**

Table 3: Detection performance (mAP and F2-Score) on the test set. The test set include images of tear gas canisters only. The "Similar" column indicates whether the detector's training set includes objects that resemble tear-gas canisters.

negative examples resulting in fewer false positives.

Faster R-CNN and RetinaNet demonstrate higher recall and lower reduction when trained on more synthetic data. To avoid generating large numbers of false positives, a calibrated blend of real and synthetic images is required to manage the trade-off of improved recall and decayed reduction.

Overall, fine-tuning state of the art object detection models perform comparatively well to fine-tuning a model with a fewshot method. It is enough to fine-tune the fewshot detector with as little as 10 images which have tear gas canisters, though results show that including canister-like objects classes is essential to get competitive results for the fewshot method. YOLOv5 and RetinaNet on the other hand benefit from other methods designed for faster convergence when using small training datasets such as the mosaic data augmentation in YOLOv5 and focal loss in RetinaNet. Results on the test set are higher than the validation set because of the presence of images with canister-like objects in the validation set.

7. Use in Human Rights Investigation

Due to the increasingly mediatized nature of protest and political action in general, a wide range of human rights violations are caught on camera in some shape or form during such events. Claims that violations occurred can now be investigated by way of online media evidence, either 'live' as the media becomes available online, or after-the-fact if the material has been archived [5, 7, 6]. In many cases, however, sifting through the video material manually is infeasible due to both its volume and unstructured nature [2]. Algorithmic methods can identify objects that strongly signify potential violations, such as tear gas canisters. Tear gas can be misused by police forces in a range of ways, such as being fired directly at the body or being used excessively in a confined space, which constitute infringements on civilian rights [4]. Tear gas canisters can therefore be used as a visual search term that structure video archives to optimize for the discovery of human rights violations [21]. While this approach will inevitably not surface footage that captures other violations, speculatively ordering media archives that are otherwise unstructured, even suboptimally, is a useful method that can significantly expedite the discovery of violations in certain cases.

8. Conclusion

We demonstrate that existing object detection models can be leveraged for the important work of investigating human rights violation, specifically the identification of tear gas canisters. Data for this domain has been lacking and we address this, beginning with our contribution of a dataset of real and synthetic tear gas canisters. Through this work, we also show several ways to overcome a lack of real world data when working in a new domain. Specifically, we successfully apply data augmentation techniques as well as using synthetic data. This can then be paired with a pretrained model for good performance. Furthermore, we investigate instances with very few examples, tens instead of hundreds, and show that the use of a few shot detector can provide good performance here.

For future work, we would like to focus on better incorporating metrics such as recall, reduction, and F2 directly into the optimization process. By aligning what users care about and the training procedure, our models can be better suited to their real world applications. Additionally, we want to do a more detailed analysis of the impact of synthetic data, focusing on aspects such as scene complexity and resolution, which can easily be controlled by the rendering engine.

We intend to evaluate the detectors within real-world deployment scenarios and explore how useful they might be to different actors, such as research organisations, citizen data scientists and activists. The detectors evaluated in this paper demonstrate variations in performance and inference time therefore further work around deployment within a variety of contexts is required to access their efficacy.

References

[1] Supervisely - web platform for computer vision.

[2] The syrian archive. https://syrianarchive.org/. Accessed: 2021-03-14.

[3] Unity real-time development platform — 3d, 2d vr ar.

[4] Did israel use excessive force at gaza protests? *BBC News*, 2018.

[5] The battle of ilovaisk. *Forensic Architecture*, 2019.

[6] The extrajudicial execution of ahmad erekat. *Forensic Architecture*, 2021.

[7] How open source experts identified the us capitol rioters. *Global Investigative Journalism Network*, 2021.

[8] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[9] Bernhard E Boser, Isabelle M Guyon, and Vladimir N Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152, 1992.

[10] Zitian Chen, Yanwei Fu, Yu-Xiong Wang, Lin Ma, Wei Liu, and Martial Hebert. Image deformation meta-networks for one-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8680–8689, 2019.

[11] Mark Everingham, SM Ali Eslami, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, 111(1):98–136, 2015.

[12] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[13] Qi Fan, Wei Zhuo, Chi-Keung Tang, and Yu-Wing Tai. Few-shot object detection with attention-rpn and multi-relation detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4013–4022, 2020.

[14] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[15] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[16] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[17] Irina Higgins, Arka Pal, Andrei A Rusu, Loic Matthey, Christopher P Burgess, Alexander Pritzel, Matthew Botvinick, Charles Blundell, and Alexander Lerchner. Darla: Improving zero-shot transfer in reinforcement learning. *arXiv preprint arXiv:1707.08475*, 2017.

[18] Jeremy Howard et al. fastai. https://github.com/fastai/fastai, 2020.

[19] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[20] Bingyi Kang, Zhuang Liu, Xin Wang, Fisher Yu, Jiashi Feng, and Trevor Darrell. Few-shot object detection via feature reweighting. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 8420–8429, 2019.

[21] Lachlan Kermode, Jan Freyberg, AI Element, Alican Akturk, Robert Trafford, Denis Kochetkov, Rafael Pardinas, Eyal Weizman, and Julien Cornebise. Objects of violence: synthetic data for practical ml in human rights investigations.

[22] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks for one-shot image recognition. In *ICML Deep Learning Workshop*, volume 2, 2015.

[23] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[24] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2999–3007, 2017.

[25] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep Learning for Generic Object Detection: A Survey. *International Journal of Computer Vision (2020)*, 128:261–318.

[26] G Lowe. Sift-the scale invariant feature transform. *Int. J*, 2(91-110):2, 2004.

[27] Sebastien Miglio. Ai in unreal engine: Learning through virtual simulations., 2019.

[28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[29] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. 2016.

[30] Joseph Redmon. Darknet: Open source neural networks in c. `http://pjreddie.com/darknet/`, 2013–2016.

[31] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[32] Joseph Redmon and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[33] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[34] Shaoqing Ren, Kaiming He, Ross B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:1137–1149, 2015.

[35] Zhiqiang Shen, Zhuang Liu, Jianguo Li, Yu-Gang Jiang, Yurong Chen, and Xiangyang Xue. Dsod: Learning deeply supervised object detectors from scratch. In *Proceedings of the IEEE international conference on computer vision*, pages 1919–1927, 2017.

[36] L. N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472, 2017.

[37] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 403–412, 2019.

[38] Ultralytics. Yolov5, 2020.

[39] L Vazquez and F Hassainia. Icevision: An agnostic object detection framework, 2020.

[40] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. In *Advances in neural information processing systems*, pages 3630–3638, 2016.

[41] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001*, volume 1, pages I–I. IEEE, 2001.

[42] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hau Yeh. Cspnet: A new backbone that can enhance learning capability of cnn. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 390–391, 2020.

[43] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30(11):3212–3232, 2019.