Comparison of Different Methods to Detect Prohibited Objects in X-Ray Image

Jerry Peng
New York University
jp4906@nyu.edu

Abstract

Security inspection is very important in maintaining social security. Thus, in order to improve its performance, we design this research project based on object detection and transfer learning techniques. We explore the differences of models base on machine learning algorithms and models base on deep learning architechture. Notably, we separately provide both pre-trained and non pretrained models for deep learning model. We utilize PIDray dataset, which contains X-Ray images that include objects prohibited by Transportation Security Administration (TSA), to train the models. Finally, we evaluate and compare these models, and find that these two types models can have similar performance in general, but can have distant performance on classification tasks for some classes. Code repository can be found at here: https://github.com/JerryPeng0201/TSA-Prohibited-Objects-Detection

1. Introduction

Security inspection, as an important process to ensure the safety of public life and property, is widely used in public, especially in public transport stations and large event sites where massive numbers of people gather. But because current machine detection is not perfect, many occasions still arrange many staff to assist the machine or even completely replace the machine for manual inspection. Traditionally, most security inspections are performed by manual inspection. However, this method is subject and therefore very susceptible to external factors. Thus, the staff are likely to give unfair or inaccurate assessments. Therefore, it's still necessary and important to improve the performance of machine detection.

In this paper, we propose a research about using deep learning and machine learning learning methods to perform detection of prohibited items from X-Ray images. In this study, we utilize two object detection methods: Faster R-CNN [14], and Histogram of Oriented Gradients (HOG) [4] with Support Vector Machine (SVM) [1]. With a security dataset PIDray [17], we build, train, evaluate, and compare these methods. We expect to contribute to improve the performance of security detection. Specifically, the contribuions of this paper are:

1. Propose a system that use Faster R-CNN to detect prohibited items from X-Ray images. We use pre-trained Faster R-CNN model and not pre-trained Faster R-CNN model to build the system and compare their performance.

2. Propose a system that use HOG and SVM to perform prohibited item detection from X-Ray images. The HOG features are displayed.

3. Compare these methods' detection performance base on multiple factors.

2. Related Works

2.1. Object Detection

Object detection is a part of Computer Vision techniques. It can identify and locate the instance of the target object(s) in an input image or video by drawing bounding box(es) or image mask(s) around them. Object detection has multiple possible applications, including but not limited to autonomous vehicles, robotics, image retrieval, and augmented reality (AR).

There are various methods to implement object detection tasks in Computer Vision. Based on the model structures, we can divide them into two categories: 1) methods utilize deep learning models, and 2) methods not utilize deep learning models.

Deep-Learning Methods Methods that utilize deep learning models usually use Convolutional Neural Network (CNN) [10] as their basic structure to learn features and perform the object detection tasks. The representative models include 1) R-CNN [6], 2) Fast R-CNN [5], 3) Faster R-CNN [14], 4) Mask R-CNN [8], 5) YOLO models [**?**], and others.

Non-Deep-Learning Methods This type of methods usually utilize feature extractions and/or descriptions plus machine learning algorithms to perform object detection tasks. The representative methods include 1) Scale-Invariant Feature Transform (SIFT) [12], and 2) Histogram of Oriented Gradients (HOG) [4]

2.2. Transfer Learning

Transfer learning is a machine learning technique. It utilizes a pre-trained deep learning models to solve a similar problem. The key is to use the weights and knowledge learned from one task and apply them to another one, usually with the goal of reducing training time or improve prediction performance. [18] provides a comprehensive survey of transfer learning.

3. Dataset

In this project, we utilize the dataset **PIDray** [17]. PIDray contains 47,677 slices of X-Ray images collected in different scenarios (such as airport, subway stations, and railway stations). It covers total 12 categories of prohibited items, namely guns, knife, wrench, pliers, scissors, hammer, handcuffs, baton, sprayer, power-bank, lighter and bullet. The distribution of object is shown in Fig. 1 and examples are shown in Fig. 2. Each image is provided with image-level and instance-level annotation. The annotation contains labeled bounding box and segmentation mask. Images are split into 29,457 (around 60%) and 18,220 (around 40%) images as train and test sets. Meanwhile, based on the difficulty degree of prohibited item detection, the test images are grouped into three subsets: *easy*, *hard* and *hidden*. Specifically, images in the subset marked as *easy* only contain one prohibited item. Images in the subset named *hard* contain more than one prohibited items. The subsets marked as *hidden* contains images that have deliberately hidden prohibited items.

4. Methods

4.1. Faster R-CNN

Faster R-CNN [14], illustrated in Fig. 3, is a two-stage object detection architecture presented in 2015. It is composed from four parts. The first part is the backbone which is used to extract image feature maps. The second component is the Region Proposal Network placed upon the feature map and the third part is the RoIPolling layer which is used to extract accurate feature maps for proposals. The last part is the detection head that classifies the proposal as well as regresses the bounding box.

RPN is a small neural network sliding on the last feature map of the convolution layers. The main purpose of this network is to output proposals that might cover an object. It

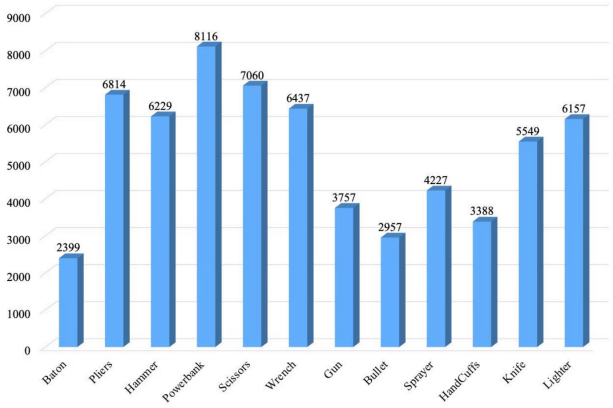


Figure 1. Distribution of objects. Image from [17]

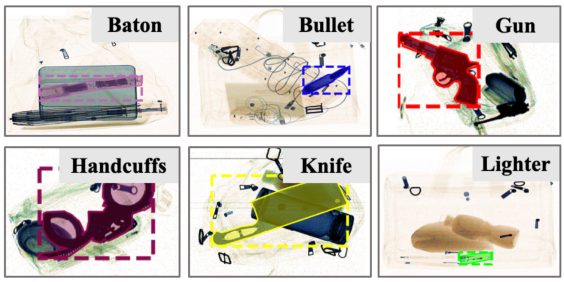


Figure 2. Dataset object examples. Image from [17]

uses N anchor boxes at each location. Anchors are translation invariant, which means the same ones is used for every location. Regression, which is one of convolutional layer in RPN, can give offsets from anchor boxes, and classification, which is another convolutional layer, gives probability of the anchors that show objects.

After output all predicted boxes, Non-Maximum Suppression (NMS) [9] is applied over the predicted bounding boxes using their predicted scores as criteria for filtration. Before processed by NMS, the Regions of Interest (ROI) are preprocessed by clipping and removing those with height, width beyond a threshold value. The top ROIs alone are taken and sorted by their confidence scores. After this process, NMS enumerates and compares all ROIs. If the IOU of this comparison results in a value greater than a predefined threshold, then that latter ROI is popped from the list.

4.2. Histograms of Oriented Gradients (HOG)

Histograms of Oriented Gradients, or HOG [4], is a feature extractor in Computer Vision and image processing. It is known for effectively detection of objects from the input images. The HOG descriptor focuses on the structure or the shape of an object, which means it mainly identifies if a pixel is a part of a edge or not. By calculating the gradient

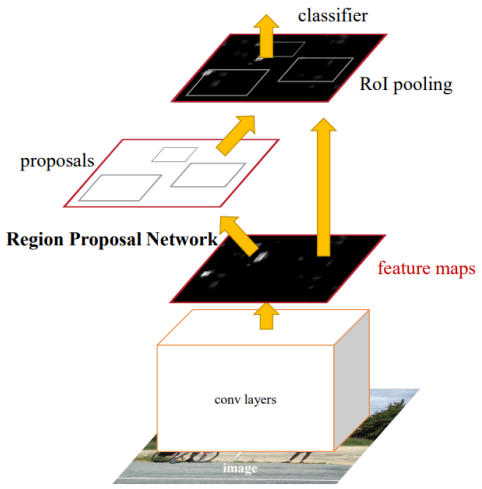


Figure 3. Illustration of RPN structure. Image from [14]

and orientation, HOG can also detect edge direction.

After processing the input image, HOG calculates the gradient magnitudes and directions for each pixel in both the x and y directions in the image. One common method for gradient computation is applying image filter or using convolution with specific kernels, such as Sobel [7]. The Sobel operator contains two convolution kernels for the horizontal gradient (Gx) and vertical gradient (Gy), respectively. These are designed to approximate the derivate of the image in both x and y directions. The magnitude can be calculate by formula 1, and the angle can be calculated by formulat 2

$$Magnitude(\mu) = \sqrt{G_x^2 + G_y} \tag{1}$$

$$Angle(\theta) = tan^{-1}(\frac{G_y}{G_x}) \tag{2}$$

After calculating the gradient for each pixel, the image is divided into cells (usually each cell has size of 8×8 or 16×16) to form blocks. For each block, a 9-point histogram is calculated. This histogram contains 9 bins of gradient directions, and the magnitudes are accumulated in the corresponding bins.

4.3. Support Vector Machine (SVM)

Support Vector Machines (SVM) [1] is a supervised machine learning algorithm used for classification and regression tasks. In [15], SVM is considered in binary classification setting. Hence, the main goal of SVM is to find the best decision boundary, or *hyperplane*, that can seperate two classes of data points. Considering we have a dataset of n labeled samples (x_i, y_i), where $x_i \in \mathbb{R}^d$ is a feature vector and $y_i \in \{-1, 1\}$ is the corresponding label. The hyperplane can be calculated by maximizing margin, which is the minimum distance between the *hyperplane* and data points from two classes. A simple SVM is shown in 4.

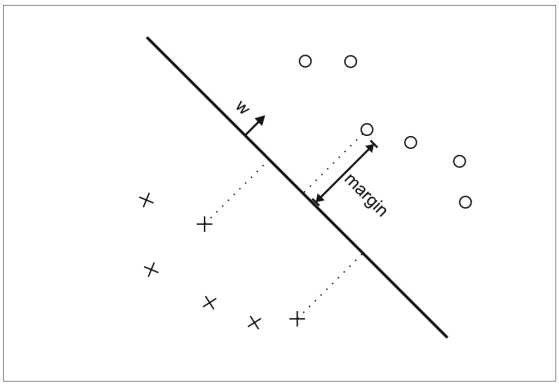


Figure 4. A Simple Linear SVM. Image from [15]

The linear hyperplane can be calculated by 3, where $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$. The distance between the hyperplane and a data point from anywhere can be represented as 4.

$$f(x) = w^T x + b \tag{3}$$

$$\frac{|w^T x + b|}{\|w\|} \tag{4}$$

For non-linearly separable data, we can use the kernel K to map the data points into a higher-dimensional space where they can be linearly separated. In [15], considering efficiency of the algorithm, researchers utilize radial basis function kernel $K(\mathbf{u}, \mathbf{v}) = (e^{-\gamma(\mathbf{u}-\mathbf{v})\cdot(\mathbf{u}-\mathbf{v})})$ which induces boundaries by placing weighted Gaussians up on key training instances.

4.4. HOGgles

HOGles [2] is a visualization tool proposed by researchers from MIT in 2013. This tool can visualize the HOG features in human-eye-interpretable form. Since features are too high dimensional for humans to directly inspect, this visualization algorithms work by inverting features back to natural images. These inversions provide an intuitive and accurate visualization of the feature spaces used by object detectors.

5. Experiment

For Faster R-CNN, we design experiments to test pretrained model's and not pre-trained model's performance and compare them. We use MMDetection [3] to implement the model training and testing. MMDetection is a open-source library developed by OpenMMLab, and is one of the popular packages in object detection field. We implement

Faster R-CNN based on this package, and utilize the training and testing tools to perform experiment. For HOG, we implement the model training and testing from scratch.

5.1. Faster R-CNN

Faster R-CNN [14] has 42 millions parameters. We are interested in seeing the different performances between pretrained Faster R-CNN and non-pretrained Faster R-CNN. Therefore, we conduct two experiments based on this model architecture. The first experiment is to train the model from scratch, which means the weights are randomly initialized. The second experiment focuses on the behavior of the pretrained model. We load the model with weights pretrained on the COCO dataset [11]. In addition, to decrease the computing time and improve the training process, we resize the image to 320×320, and then use random flip augmentation.

For both experiments, we utilize step learning rate scheduler. In this scheduler, initial learning rate is set to 0.02 and after 8 epochs the learning rate decays to 0.0002. We set 50 epochs training and run the process on NYU HPC Greene Cluster, with one RTX8000 GPU, setting the batch size at 16. After all training epoch, both models are evaluated based on the test tools from mmdetection [3] and testing set. We mainly focus on the evaluations on bounding boxes and classification.

5.2. HOG & SVM

For this experiment, we scale the image to 320×320, and normalize them with 0.5 for mean's value and 0.5 for standard deviation value. Then we iterate through the dataloader and compute HOG features for each image by using scikit-image [16]. After getting the HOG featurs for each images, we train the SVM classifier with scikit-learn [13].

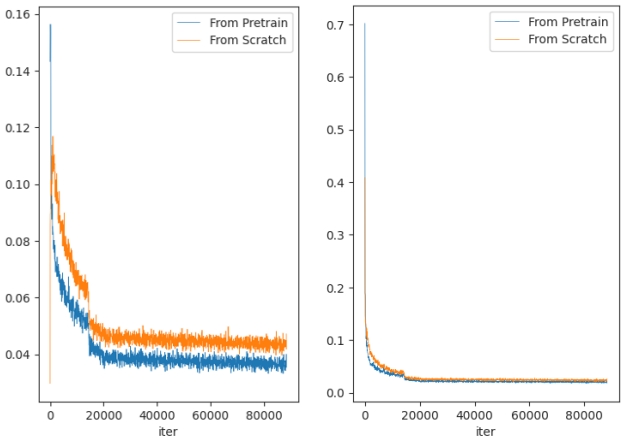
Since HOG [4] and SVM [1] were designed and developed during the time when CPU is the primary choice for computation source, we decide to train such model on CPU rather GPU.

6. Results

We use standard COCO Average Precision (AP) and mean Average Precision (mAP) as our performance evaluation metric. COCO AP is computed by selecting 10 IoU thresholds between 0.50 and 0.95 and computing the Average Precision for each class, and mAP can be gotten by computing the mean of Average Precision. Due to the time limitation, we decide to use *easy* test set to evaluate our model performance.

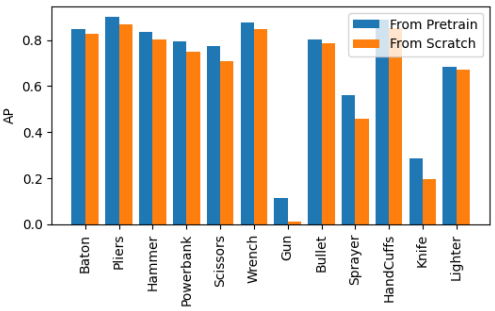
6.1. Faster R-CNN

Based on the training process, the losses of regression and classification have converged. Fig. 5 illustrates the loss curves of regression and classification from Faster R-CNN

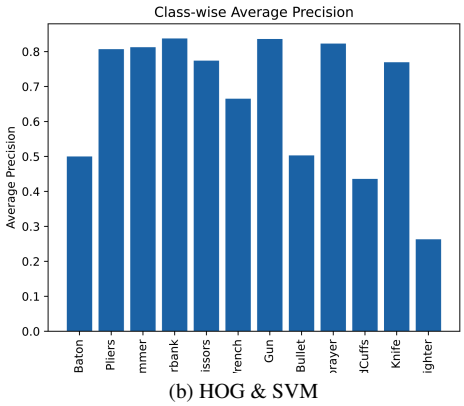


(a) Regression loss VS Iteration (b) Classification loss VS Iteration

Figure 5. Faster R-CNN Training Loss



(a) Faster RCNN



(b) HOG & SVM

Figure 6. Class-wise AP on *easy* test set

training process. From Fig. 5a, the pretrained model shows better performance, while according to Fig. 5b, the loss curves are similar. This might be caused by randomly initializing both model heads.

Another interesting point that we notice from Fig. 5 is the loss curves of the pretrained model and non-pretrained model show a falling off a cliff at around 8 epochs. This is due to the decay of the learning rate.

Method	mAP
Faster-RCNN / Scratch	0.656
Faster-RCNN / Pretrain	0.697
HOG & SVM	0.669

Table 1. Final test results on *easy* test set.

From Fig. 6a, we notice that for class *Gun*, *Sprayer*, and *Knife*, both models show significantly lower average precision compare to other classes. We assume this phenomenon can be caused by lower number of training/testing images, or the images contain these classes are more complicate and more noisy. Considering the dangerous level of these three types of prohibited items, this phenomenon can be an interesting research point for future.

6.2. HOG

From 6b, we can see that the HOG & SVM model also has a good performance. Most target detection and classification for the 12 classes can reach higher than 0.7 of average precision. However, comparing to the class-wise average precision of Faster R-CNN, we found that the HOG & SVM model has better performance on detecting and classifying *Gun*, *Sprayer*, and *Knife*, but it does not reach a satisfying average precision for *Baton*, *Bullet*, *Lighter*, and *HandCuffs*.

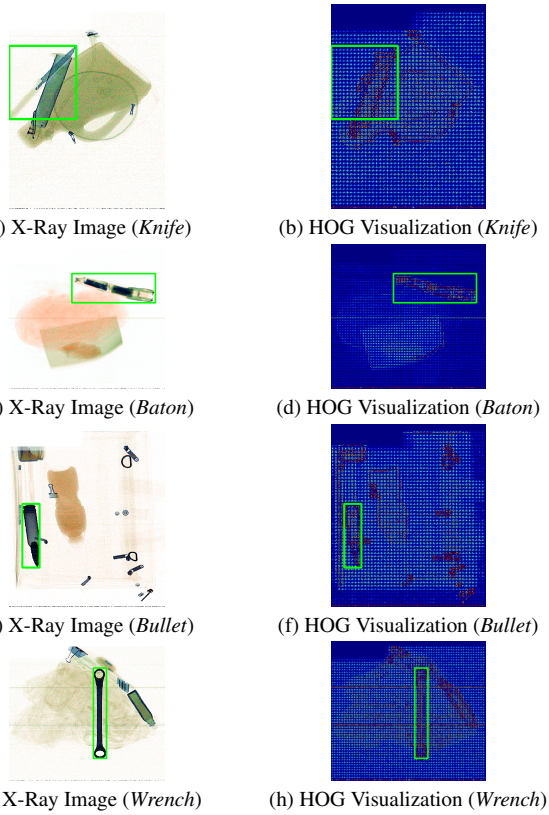
6.3. Model Comparison

The mAPs for each three models are shown at table 1. Based on the mAP, we can see that the three models have similar performance results on PIDray dataset, but in general, the Pre-trained Faster R-CNN has the best performance of detecting and classifying the TSA forbidden items, and HOG & SVM reaches the second best performance.

However, if we compare these three models based on the classification of the 12 classes, we can see that for *Gun*, *Sprayer*, and *Knife*, HOG & SVM has the better performance comparing to Faster R-CNN, while Faster R-CNN reaches higher average precision on the rest of classes, especially for for *Baton*, *Bullet*, *Lighter*, and *HandCuffs*. We believe this feature can be an interesting point and worth to do more research in the future, considering that these two types of models have completely different architectures.

6.4. Visualize HOG Features

We calculate the HOG features for detected prohibited items and visualize the HOG features for them. The two set examples are provided at 7. We use the trained model to make prediction of bounding boxes, and visualize the bounding box locations on the HOG visualization map.



(a) X-Ray Image (*Knife*) (b) HOG Visualization (*Knife*)
(c) X-Ray Image (*Baton*) (d) HOG Visualization (*Baton*)
(e) X-Ray Image (*Bullet*) (f) HOG Visualization (*Bullet*)
(g) X-Ray Image (*Wrench*) (h) HOG Visualization (*Wrench*)

Figure 7. Example of HOG Visualization

6.5. Future Directions

Based on the results we gain from the experiment, we think the following three questions can be interesting and worthy research points in the future:

1. Why Faster R-CNN does not have good performance on detecting *Gun*, *Knife*, and *Sprayer*?
2. Why HOG & SVM does not have good performance on detecting *Baton*, *Bullet*, *Handcuffs* and *Lighter*?
3. Why HOG & SVM model and Faster R-CNN can have distant performance on classification?

7. Conclusion

In this paper, we design, build, train, and evaluate two models that have different types of architectures. We compare these two models' performances in general and in classification for the 12 classes. We find that the model bases on machine learning algorithms and deep learning model can have distance performance on classification tasks, but they also show the similar mean average precision. For some deep learning model, like Faster R-CNN, pre-trained version can have higher precision (4% higher mAP) after the model converges. For furture research, we believe it would

be interesting and worthy to learn why machine learning models and deep learning models can have distant performance on the classification tasks, especially for some specific classes from the dataset.

8. Acknowledgement

We appreciate Dr. Jean Ponce from New York University and New York University HPC team to provide necessary guides, help, and resources to support this project.

References

[1] 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. ACM, 1992. 1, 3, 4

[2] Tomasz Malisiewicz Antonio Torralba Carl Vondrick, Aditya Khosla. Hoggles: Visualizing object detection features. *2013 IEEE International Conference on Computer Vision*, 2013. 3

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 3, 4

[4] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005. 1, 2, 4

[5] Ross B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. 1

[6] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. 1

[7] Rafael C Gonzalez, Richard E Woods, and Steven L Eddins. *Digital Image Processing*. Pearson Prentice Hall, 2006. 3

[8] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. 1

[9] Bernt Schiele Jan Hosang, Rodrigo Benenson. Learning non-maximum suppression. *Computer Vision and Pattern Recognition*, abs/1911.08287, 2017. 2

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[11] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Doll'a r, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 4

[12] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 2

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. 4

[14] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 1, 2, 3, 4

[15] Simon Tong and Edward Chang. Support vector machine active learning for image retrieval. In *Proceedings of the Ninth ACM International Conference on Multimedia (MULTIMEDIA '01)*, pages 107–118. ACM, 2001. 3

[16] Stéfan van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014. 4

[17] Boying Wang, Libo Zhang, Longyin Wen, Xianglong Liu, and Yanjun Wu. Towards real-world prohibited item detection: A large-scale x-ray benchmark. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5412–5421, 2021. 1, 2

[18] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685, 2019. 2