WWJMRD 2022; 8(12): 72-77
www.wwjmrd.com
International Journal
Peer Reviewed Journal
Refereed Journal
Indexed Journal
Impact Factor SJIF 2017: 5.182 2018: 5.51, (ISI) 2020-2021: 1.361
E-ISSN: 2454-6615

Harsh Shekhar
B.Tech Graduate, Computer Science and Engineering, VIT, Vellore, Tamil Nadu, India.

Object detection and classification on X-ray baggage scanning dataset using neural network

Harsh Shekhar

Abstract
The aim for this project was to create an automatic system for security baggage scanning at the airports by training and testing three different neural network algorithms namely FRCNN, RFFCN, YOLOv2 and I also wanted to make the system faster for training purpose for which I made some changes on FRCNN code and also on YOLOv2 and then tested these algorithms on x-ray baggage image dataset which would include various firearm components such as guns, knives, pliers and wrenches, which would be detected by the algorithms and then these algorithms would be compared on various parameters. I have divided the comparing parameters as primary parameters and secondary parameters where primary parameters would be the one which would be manually analyzed from the resultant output testing images and the secondary parameters would be the ones which would be calculated from the primary parameters. These parameters were then compared for each of the firearm component separately to conclude the best algorithms out of the three for x-ray image dataset scanning.

Keywords: security, collaboration, special education, disability

1. Introduction

As everyone would have experienced on airports that there is a huge delay while security check due to manual checking of X-Ray baggage scans, which is time taking, as human mind takes more time to capture the image and analyze it thoroughly to draw conclusions. For some baggage they have to cross check it again and again as they are not sure of their decision, all this causes a lot of delays in the process. The objective is to minimize the delay as well as to increase the precision of the decisions by creating an optimized automated system for baggage scanning using neural networks which would give more efficient results, would be able to identify if there is any harmful or illegal object present in the baggage or not and would be much faster than the manual process.

For implementation purpose I have used an x-ray baggage image dataset and implemented it on three neural network algorithms namely- FRCNN, YOLO v2 and R-FCN. Out of these three I made some changes in FRCNN code and YOLO v2 code to make them train and process faster without disturbing their accuracy and precision. These algorithms would detect various components like – guns, knife, pliers and wrenches. After testing I compared the algorithms using various parameters, I have divided these parameters as primary parameters and secondary parameters. The primary parameters (True Positive, False Positive, True Negative and False Negative) are the ones which were observed manually through the testing images and secondary parameters (Accuracy, True Positive Rate, False Positive Rate, Precision, F-score) are the one which are calculated using the primary parameters. Using these parameters, I finally concluded the best algorithm out of the three for such application.

2. Related Work

The aviation security systems have been researched on for decades in order to improve and increase the level of security using imaging technologies. Different kinds of scanning techniques are used in order to detect harmful substances, weapons, etc. in order to maintain safety of the passengers. However, it still remains a big area of concern as new and

Correspondence:
Harsh Shekhar
B.Tech Graduate, Computer Science and Engineering, VIT, Vellore, Tamil Nadu, India.

developed imaging techniques are developed. Previous works have focused on image enhancement, classification, segmentation or detection. My main purpose is to address the object classification and detection tasks as explained below.

Classification- Previous works [1], [2], [3], [4], [5] include the use of traditional approaches of machine learning based on Bag-of-Visual-Words(BoVW) scheme for representation of features, also using hand-crafted features together with Support Vector Machine(SVM) classifier. The work done by [1] shows the use of SVM classification along with feature representations (DoG, DoG+SIFT, DoG+Harris) in order to apply the BoVW concept. In [3], various feature point descriptors have been used as visual word variants within a BoVW model. This is done in order to achieve threat detection which is based on image classification. Descriptor combination along with FAST-SURF feature detector is used in order to achieve maximal performance with an SVM classification.

Detection-In order to identify and separate objects as a threat or non-threat, classification is a major step. To identify and detect an object, localization is done on the X-ray image which is then denoted by a shape or bounding box. In [6], regions of interest (ROI) are being detected using a geometric model of the object. In [7], sliding window detection is used along with the SVM classifier as well as histogram of oriented gradients (HOG)[8]. As seen above, most research is done using BoVW techniques; therefore, this paper is focused on CNN classification architectures as the advancements in CNN literature have only been evaluated to a limited extent.

BoVW techniques basically created a visual dictionary and were trained on the images using this visual dictionary; this dictionary included various features of the object to be detected. The major problem with this approach was that different objects may have some features in common or may have features whose properties are similar thus just by describing such features and forming a dictionary did not result in accurate results and it gave a lot of false alarms.

Hence for this paper I have focused more on new emerging techniques of convolution neural networks and tried various algorithms on CNN for comparing and concluding I also made some changes and experimented with some of the algorithms to make them faster and to find the best approach for this particular application.

3. System Architecture

Figure 1 shows my system architecture and my whole process of implementation.

I started by obtaining a suitable image dataset of x-ray baggage scans and then I trained and tested the three neural network models, after the testing was completed, I manually observed the primary parameters and using these primary parameters I calculated the secondary parameters and finally by comparing these parameters, moved towards the result and conclusion.

The implementation details of all the three algorithms are given below -

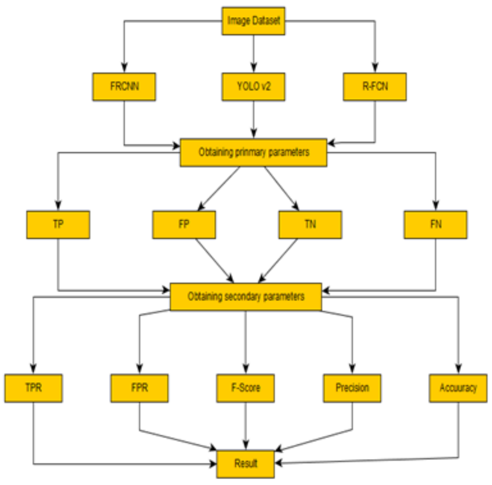

Fig. 1: System Architecture

3.1 FRCNN

Faster RCNN takes image as input and applies various convolution layers on it to obtain feature maps, from which the region proposals are obtained using RPN. These region proposals are then provided to the classifier after going through ROI pooling which is basically applying max pooling but only on the region of interests rather than the whole image.

I implemented the FRCNN code on Google CoLab platform as it provided with online GPU on its servers, but the session time provided was of only 10 hours within these 10 hours I was able to train on an image dataset of 500 images for 500 epochs consisting of 50 iterations each thus undergoing a total of 25000 rounds of training.

For making it faster I made some changes in the stride of the FRCNN algorithm. Stride is one of the parameters of FRCNN code, part of the convolution layer, which determines the movement of the convolution filters over the image, i.e. amount of pixels the convolution filter will move after every iteration. I tried the FRCNN code with stride value of 1 and stride value of 2. The convolution layer used is of 3x3 matrix hence possible values of strides are 1 and 2 as values above 3 would have hampered the accuracy and precision. Stride value one means the filter would move by one pixel after each iteration whereas in case of stride 2 the filter would move by two pixels at a time.

I trained and tested on only 3 out of the 4 components and the plier’s component was left for this algorithm as the number of images for training were too less and training only 500 images for 4 components would have resulted in much more drop in accuracy for all the components.

Once the training was completed testing was done on 200 images and the parameters were noted for each component separately.

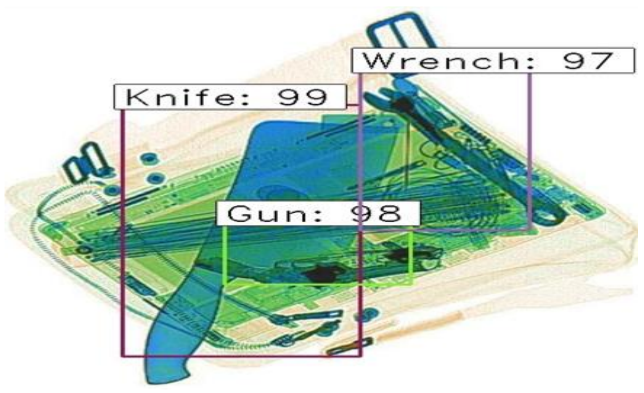

Fig. 2: Sample output of FRCNN

Figure 2 shows a sample output of our FRCNN code implementation, here we can clearly see a gun, a knife and a wrench detected successfully with their labels and percentage of match with the class.

3.2 R-FCN

The R-FCN algorithm is similar to FRCNN code where an image is taken as input then CNN layers are applied to get feature maps. The difference is that the RPN layer is absent in R-FCN and in place of RPN, ROI pooling is done alongside with generation of score maps which would map the class vote to each of the features detected and thus as per the vote scores the objects would be classified.

I trained the R-FCN code on Kaggle Platform similar to Google CoLab, it is a platform which provides with free online GPU for the training and testing purposes.

Kaggle also provided 10 hours of session time for training; within this time period I was able to train the system for 800 images and for all the 4 components.

The training was completed in 4 stages –

Stage 1 – 240 epochs

Stage 2 – 480 epochs

Stage 3 – 960 epochs

Stage 4 – 1920 epochs

And each epoch consisted of 4 iterations.

Once the training was completed, the model was implemented on 200 images.

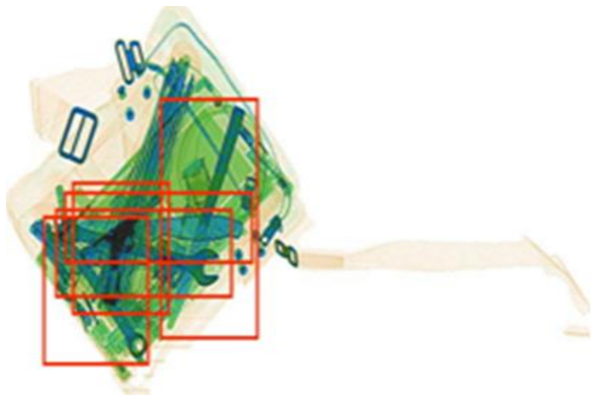

Figure 3 – Sample output of R-FCN

Figure 3 represents a sample output image of R-FCN algorithm where we can observe various components detected in red boxes.

3.3 YOLO v2

Yolo v2 algorithm takes image as the input and applies convolution layers followed by max pooling layers for feature extraction by decreasing the resolution of the image and increasing the depth of the image.

The input image is divided into squares and on each of those square grids the algorithm predicts five bounding boxes each of them with different aspect ratios. After obtaining the bounding boxes, the algorithm predicts the center of the box and then calculates the confidence score of having any object in that square along with the probabilities of which class the object belongs to; in our case I have set the threshold of the confidence score and probability both at 0.3 i.e. if the algorithm detects any object as firearm of 30% or above or if any object gives 30& or more match to any of the classes, the algorithm would form a bounding box around it and label the object as the class name with which it has the highest match.

I have implemented the YOLO v2 algorithm on Google CoLab platform which provided 10 hours of session time just like the other algorithms and in these 10 hours I was able to train 1000 images which is the maximum among the three which clearly indicates YOLO v2 is the fastest among the three. I made some changes in the YOLO v2 code to make it faster in the training period so that it would be able to train on more number of images in the certain amount of time period.

For making it faster I added a variable to store the loss value of the last iteration and a counter variable. After the first epoch the loss value would be stored in a variable and the counter would be set to 1. After every epoch the obtained loss value would be compared to the loss variable if it is equal or greater, then the counter would be incremented else the counter would be reset to one and the loss variable would be updated to the new loss value. Once the counter reaches the value of three the system would go under early execution and would not process further epochs.

This condition shows that the loss value is not decreasing for the last three epochs which means that the training set is reaching its saturation and the system is well trained for this particular training set and weights file is ready and the system can go ahead for the testing step and training on further epochs is not necessary as the loss value is not likely to decrease much further. If we want to decrease the loss value further, we would need to increase the number of images in the training set.

I trained for 50 epochs but the execution went under early execution at 36 epochs due to no decrease in the loss function since the last three epochs.

The training was conducted for all the four objects and after the training was completed, I tested the model for 200 images.

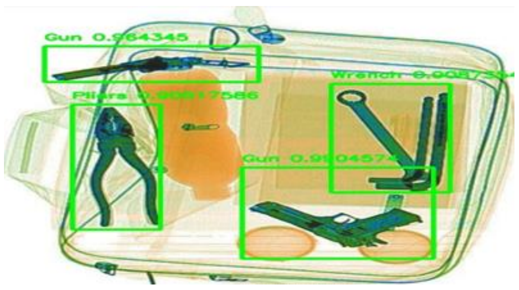

Fig. 4: Sample output of YOLO v2

Figure 4 represents a sample output of YOLO v2 where we can see two guns, one wrench and one plier are detected in green boxes with labels as their class name and probabilities of matching.

3.4 Parameters

I have divided the comparison parameters as primary parameters and secondary parameters –

3.4.1 Primary Parameters

Primary parameters are the one which I observed manually from the tested results.
TP: True Positive, which was determined by how many correct predictions the algorithm made, i.e., how many components were correctly detected.
TN: True Negative, which is determined by how many negative predictions were done correctly, i.e., if any particular component is absent in the image, then the system, should not detect the component in that image.
FN: False Negative, which is determined if any component is left undetected in any image.
FP: False Positive, which is determined if any component is wrongly detected or a blank space is detected as one of the components.

3.4.2 Secondary Parameters

These are the parameters which are calculated from the primary parameters.
TPR: True Positive Rate is given by the following formula –
TPR = TP/ (TP+FN)
It is also known as Recall it determines - out of the present components, how many components were predicted correctly.
FPR: False Positive Rate is given by the following formula –
FPR = FP/ (FP+TN)
It determines - out of all the blank spaces how many were wrongly detected as a component.
Accuracy: Accuracy is calculated by the following formula –
Accuracy = TP+TN/ (TP+FN+TN+FP)
It determines the ratio of total number of correct predictions of presence and absence of components to the total number of cases.
Precision: Precision is calculated as –
Precision = TP/ (TP+FP)
It determines how many detections made are correct.
F-score: It is the harmonic mean of Recall and Precision. Hence the formula is,
F-score = 2* Precision * Recall / (Precision + Recall)
Higher the value of F-score more perfect is the Precision and Recall.

4 Results and Discussion

First, I made some changes in the FRCNN code to make it faster by changing number of strides, for this I tested for stride value of one and stride value of 2. Thus, I had initial two cases for FRCNN. I trained the system for both the cases for the time interval of 12 hours as provided by the online platforms as a result I was able to observe that in these 10 hours the code with stride value of 1 was able to train only 400 images which would give really bad and non-accurate results for 4 different components as training on only 400 images is not sufficient for identifying four different objects. In the other case where the stride value was two I trained for the same amount of time of 10 hours and was able to train on 500 images which is still not sufficient for 4 different objects but going for stride value of 3 on a convolution matrix of 3X3 would have resulted in overall depletion of accuracy and precision as many features could have gone undetected, thus I proceeded with stride value of two and trained it only for 3 objects as training on 500 images could be used for detecting 3 objects and stride value of 2 made the code faster as it was able to train on 100 more images in the same time span.
Next, I modified the YOLOv2 code where I constructed an additional function of early exiting as I observed in training from the base code, that after a certain value of epochs the loss was not decreasing but training on large number of images caused each epoch to execute slower and thus, I added early exiting function, through which the code will undergo early execution if the loss value is not decreased for three consecutive epochs. This helped to improve the code by training a greater number of images in less time once the loss value is not changing the execution would stop. Though it has a smaller number of epochs as compared to other algorithms but each epoch of YOLOv2 takes much more time than the epochs for other algorithms thus waiting for complete execution of all the epochs without improving the loss value was just a wastage of time hence through my method one can execute the code faster and there is no need to execute all the epochs, once the loss value hits minimum the code will stop training.
I planned to go for 50 epochs and my code had early exiting on epoch value of 36 which occurred under 10 hours and the system was trained for 1000 images the highest out of all the three and thus I can test it for all the objects, but if it would have to complete all the 50 epochs it couldn’t have completed it in 10 hours, the number of images had to be greatly decreased.
I have tested each of the algorithms for 200 images. For analyzing the results, I have calculated the various parameters for each component to be used for comparative analysis between the algorithms.
The comparative analysis would be done in tabular form for each component.

4.1 Gun Component

Table 1: Gun Component parameters for all 3 algorithms

	YOLO v2	FRCNN	R-FCN
TP	277	131	59
FP	28	21	12
TN	48	103	113
FN	7	7	9
TPR	0.975	0.95	0.867
FPR	0.368	0.17	0.096
Accuracy	0.902	0.89	0.86
Precision	0.908	0.86	0.83
F-score	0.940	0.90	0.848

As one can observe in the above table TPR is highest in YOLOv2 i.e., 0.975 or 97.5% algorithm that means the algorithm was able to detect 97.5% of the samples correctly.
Accuracy, Precision and F-score are also observed highest in YOLO v2 algorithm, hence one can successfully conclude from these results that the YOLO v2 algorithm gave the best results for the Gun component.

4.2 Knife Component

Table 2: Knife Component parameters for all 3 algorithms

	YOLO v2	**FRCNN**	**R-FCN**
TP	190	275	117
FP	18	7	15
TN	69	0	76
FN	27	97	11
TPR	0.875	0.74	0.914
FPR	0.206	1	0.164
Accuracy	0.852	0.725	0.965
Precision	0.91	0.975	0.886
F-score	0.892	0.841	0.899

As observed in the above table TPR and Accuracy is highest in R-FCN for the knife component that indicates R-FCN was able to detect most samples correctly.
The precision is highest in FRCNN i.e., 0.975 or 97.5% which implies that majority of the predictions made by the algorithm were correct, but there is a vast difference in its accuracy and precision as it had left lots of samples undetected which had hampered its accuracy. So, even though the algorithm made minimum error when it detected the component but it had left lots of samples undetected. Which has also affected its F-score which implies the TPR and the Precision is not that perfect.
The FPR observed in FRCNN is 1 or 100% that is because its TN value is 0 which means there was no image in which the Knife component was absent.
As observed the F-score is highest in R-FCN algorithm which implies its TPR and Precision are the most perfect.

4.3 Wrench Component

Table 3: Wrench Component parameters for all 3 algorithms

	YOLO v2	FRCNN	R-FCN
TP	53	33	26
FP	2	3	8
TN	145	142	110
FN	4	76	3
TPR	0.92	0.3	0.896
FPR	0.013	0.02	0.067
Accuracy	0.97	0.688	0.68
Precision	0.96	0.916	0.764
F-score	0.939	0.45	0.824

Results are clearly conclusive as YOLO v2 has best TPR, FPR, Accuracy, Precision and F-score as compared to the other two algorithms.
There is a vast difference in the accuracy and precision of FRCNN algorithm because though it was able to detect the components accurately with very less number of wrong detections but it left out a lot of samples undetected due to which it's accuracy was hampered greatly. And so is its F-score which has fallen even below 50%.

4.4 Pliers Component

Table 4: Pliers Component parameters for all 2 algorithms

	YOLO v2	R-FCN
TP	60	8
FP	0	4
TN	142	120
FN	0	1
TPR	1	0.88
FPR	0	0.032
Accuracy	1	0.64
Precision	1	0.66
F-score	1	0.758

FRCNN is not present since it was trained for only 500 images and thus it couldn't be trained for all the 4 components thus, Plier's component was absent in training as well as testing of FRCNN.
For the YOLO v2 algorithm there were no wrong detection and neither was any Pliers left undetected thus its FN and FP values are 0 and since FP was 0 thus FPR also became 0%.
The Accuracy and Precision are 100% since there were no errors found in detecting the Pliers component using YOLO v2 algorithm.
So clearly, YOLO v2 algorithm gave best results for detecting Plier's component with an F-score of 100% which implies that it's TPR and Precision has achieved perfection.

5. Conclusion

I was successfully able to implement 3 algorithms namely – YOLO v2, FRCNN and R-FCN, on the online platforms. I had successfully trained all the algorithms and tested these algorithms on 200 images each. I was successful in improving the performance of YOLOv2 code and FRCNN code by decreasing their time taken for training and hence completed training on a greater number of images ultimately giving better results.
I have accurately calculated all the parameters for each component namely, TP, TN, FP, FN, TPR, FPR, Accuracy, Precision, F-score.
And observing these results one can reach to the conclusion that out of the three, YOLO v2 is the best algorithm for detecting various firearms components in X-ray Baggage scanning, as in the training period YOLO v2 was the fastest to execute and was able to train on 1000 images which is the highest compared to the other two algorithms, and it gave the best results for most of the components except the Knife component where R-FCN had the best F-score but even in that case the difference between F-score of R-FCN and YOLO v2 was minimal.
Hence, one can arrive on the conclusion that YOLO v2 is the best algorithm to implement on X-ray Baggage Dataset.

Reference

1. M. Bas ̧tan, M. R. Yousefi, and T. M. Breuel, "Visual words on bag- gage X-ray images," in Computer Analysis of Images and Patterns. Berlin, Germany: Springer, 2011, pp. 360–368.
2. D. Turcsany, A. Mouton, and T. P. Breckon, "Improving feature- based object recognition for Xray baggage security screening using primed visualwords," in Proc. IEEE Int. Conf. Ind. Technol., Feb. 2013, pp. 1140 1145.
3. M. E. Kundegorski, S. Akçay, M. Devereux, A. Mouton, and T. P. Breckon, "On using feature descriptors as visual words for object detection within X-ray baggage security screening," in Proc. Int. Conf. Imag. Crime Detection Prevention, Nov. 2016, p. 12.
4. D. Mery, E. Svec, and M. Arias, "Object recognition in baggage inspec- tion using adaptive sparse representations of X-ray images," in Image and Video Technology. Cham, Switzerland: Springer, 2016, pp. 709–720.
5. D. Mery, V. Riffo, I. Zuccar, and C. Pieringer, "Object recogni- tion in X ray testing using an efficient search

algorithm in multiple views," Insight Non-Destructive Test. Condition Monitor., vol. 59, no. 2, pp. 85–92, 2017.

6. D. Mery, "Automated detection in complex objects using a tracking algorithm in multiple X-ray views," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, Jun. 2011, pp. 41–48.
7. T. Franzel, U. Schmidt, and S. Roth, "Object detection in multi-view X- ray images," in Pattern Recognition. Berlin, Germany: Springer, 2012, pp. 144–154.
8. N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. (CVPR), vol. 1. Jun. 2005, pp. 886–893.