Pedestrian Detection in Infrared Images Using FastRCNN

Asad Ullah (*Author*)
Information Engineering School
Chang'An University
Xi'an, 710064
P.R of China,
Asad_uop92@yahoo.com

Farooq Muhammad Omer (Co-Author)
Department of Electronics
University of Peshawar,
25000, Pakistan,
omer908@yahoo.com

Xie H.M
School of Electronics and Information
NWPU, Xi'an, 710072,
P.R of China,
xiehm@nwpu.edu.cn

Sun Zhaoyun,
Information Engineering School
Chang'An University
Xi'an, 710064
P.R of China,
mahw@xust.edu.cn

***Abstract*—Compared to visible spectrum image the infrared image is much clearer in poor lighting conditions. Infrared imaging devices are capable to operate even without the availability of visible light, acquires clear images of objects which are helpful in efficient classification and detection. For image object classification and detection, CNN which belongs to the class of feed-forward ANN, has been successfully used. Fast RCNN combines advantages of modern CNN detectors i.e. RCNN and SPPnet to classify object proposals more efficiently, resulting in better and faster detection. To further improve the detection rate and speed of Fast RCNN, two modifications are proposed in this paper. One for accuracy in which an extra convolutional layer is added to the network and named it as Fast RCNN type 2, other for speed in which the input channel is reduced from three channel input to one and named as Fast RCNN type 3.**

Fast RCNN type 1 (original Fast RCNN) has better detection rate than RCNN (60.3636 : 54.54318), with 3.5 x RCNN training and 2.5 x RCNN test speed. Compare to Fast RCNN, Fast RCNN type 2 has better detection rate (63.42% : 60.36%) while Fast RCNN type 3 is faster (having 1.018 x FastRCNN training and 1.04 x FastRCNN test speed).

Keywords*—*ANN, CNN, Overfeat, SPPnet, RCNN, Fast RCNN.

I. INTRODUCTION

Pedestrian detection is an extensively studied research area in image processing due to its importance for practical application. In image processing, as compared to visible spectrum image the infrared image is much clearer in poor lighting conditions. Infrared imaging devices are capable of operating even without the availability of visible light. The fast-improving functioning and decreasing prices have spurred speedy expansion in infrared cameras usage in a wide range of areas with varied capabilities. Infrared Image processing has numerous applications and great research value in industrial, scientific, and medical fields. E.g. automatic detection of targets through infrared devices has been widely used in civil and military domains and forms the basis for infrared search and track (IRST) [1] [2]. In the civil domain, video surveillance and smart vehicle driver assistance mostly use infrared technology. Pedestrian detection systems can be implemented in vehicles to automatically brake a vehicle to avoid striking a pedestrian [3] [4]. Pedestrian detection and recognition is an important part of intelligent monitoring system, hence becoming an actively growing research area in computer vision.

Conventional methods for pedestrian detection [5] [6] [7] consist of feature extraction, which is manually designed, and trained classification, which is based on machine learning techniques. In particularly, feature design requires human knowledge to ensure robustness. To overcome this problem, deep learning-based approaches have attracted attention recently. Deep learning, especially in CNN, has excellent feature representation and classification abilities. Krizhevsky has shown the potential of CNN on 1000 class object recognition benchmark (ILSVRC) [8] . After this breakthrough, Deep Learning approaches have been applied to many problems, such as house number recognition, scene labeling, object classification and detection.

II. RELATED WORK

In 2014 an integrated framework was proposed by Pierre Sermanet et al by using Convolutional Network for classification, localization and detection simultaneously [9]. This new methodology for localization and detection by collecting projected enclosing boxes depicted that integrating various localization calculations, detection is possible without the requirement of training on background samples. It also showed that lengthy and complex bootstrapping training passes are avoidable which allows the network to be focused on positive classes only. Later in 2014, Region Based Convolution Neural Network (R-CNN) was proposed by Ross Girshick et al [10] in which they used 2000 category independent region proposals generated for the input image, a fixed-length feature vector was extracted from each proposal using CNN which then classified each region with category-specific linear SVMs. To compute a fixed-size CNN input from each region proposal "affine image wrapping" is used, regardless of the region's shape. This warpping of the content sometimes results in unwanted distortion. Recognition accuracy is decreased due to the content loss or distortion. Besides, a pre-defined scale may not be suitable when object scales vary. To solve this problem, Kaiming He et al in 2015 introduced Spatial Pyramid Pooling Layer to CNNs to remove the fixed size constraint and named it SPPnet [11] in which the convolutional layers accept arbitrary input sizes but they produce outputs of variable sizes. The classifiers (SVM/Softmax) or fully-connected layers requires fixed-length vectors. Such vectors can be generated by the Bag-of-Words (BoW) approach [12], that pools the features together. Spatial pyramid pooling [13] improves BoW in which it maintains spatial information by pooling in local spatial bins. These spatial bins have sizes proportional to the image size, keeping the number of bins fixed regardless of the image size. Considering the advantages RCNN and SPPnet, Ross Grishick in 2015 proposed Fast RCNN [14], a combined version of

RCNN [10] and SPPnet [11] which efficiently classifies object proposals using deep convolutional networks.

III. FAST RCNN

A Fast R-CNN network takes as input an entire image and a set of object proposals generated by region proposal network as input [15]. The network first processes the whole image with several convolutional and max pooling layers to produce a conv feature map. Then, for each object proposal a region of interest (RoI), pooling layer extracts a fixed-length feature vector from the feature map. Each feature vector is fed into a sequence of fully connected (fc) layers that finally branch into two sibling output layers One that produces softmax probability estimates over K object classes plus a catch-all "background" class and Another layer that outputs four real-valued numbers for each of the K object classes. Each set of 4 values encodes refined bounding-box positions for one of the K classes. (See figure below).

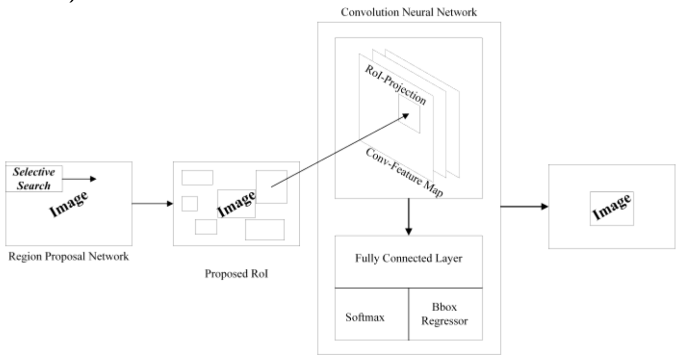

Fig.1. Overview of Fast RCNN

A. Region Proposal

A variety of methods are available for generating category-independent region proposals such as Selective search [16], Objectness [17] and category-independent object proposals [18] etc. Fast RCNN uses selective search 'Edge Boxes' locating object proposal [15] method to generate RoI from edges within an image.

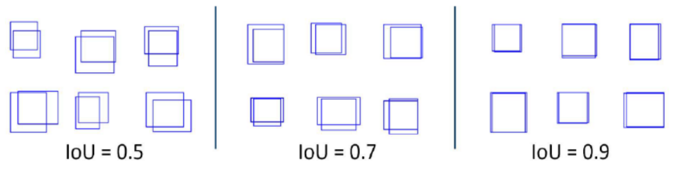

Fig. 2. An illustration of random bboxes with IoU of 0.5, 0.7, and 0.9

a bounding box is typically measured using the Intersection over Union (IoU) metric. IoU computes the intersection of a candidate box and the ground truth box divided by the area of their union. IoU scores of greater than 0.5 are generally desired. An IoU of 0.7 provides a reasonable compromise between very loose (IoU of 0.5) and very strict (IoU of 0.9) overlap values.

B. RoI Pooling

The RoI pooling layer uses max pooling to convert the features inside any valid region of interest into a small feature map with a fixed spatial extent of H ×W (e.g., 7 × 7), where H and W are layer hyper-parameters that are independent of any particular RoI. RoI is a rectangular window into a conv feature map as shown in fig.1 by RoI-Projection. Each RoI is defined by a four values (r, c, h, w) that specifies its top-left corner (r, c) and its height and width (h, w).

C. Training

Fast R-CNN uses a streamlined training process with one fine-tuning stage that jointly optimizes a softmax classifier and bounding-box regressors, rather than training a softmax classifier, SVMs, and regressors in three separate stages. The components of this procedure are the loss, mini-batch sampling strategy, back-propagation through RoI pooling layers, and SGD hyper-parameters.

1) Multi-task loss

A Fast R-CNN network has two sibling output layers. The first outputs a discrete probability distribution (per RoI), $p = (p_0, \dots\dots, p_k)$, over $K + 1$ categories. As usual, p is computed by a softmax over the $K + 1$ outputs of a fully connected layer. The second sibling layer outputs bounding-box regression offsets, $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$, for each of the K object classes, indexed by k. The parameterization for t^k specifies a scale-invariant translation and log-space height/width shift relative to an object proposal. Each training RoI is labeled with a ground-truth class u and a ground-truth bounding-box regression target v. Multi-task loss L on each labeled RoI to jointly train for classification and bounding-box regression:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1] L_{loc}(t^u, v) \qquad (1)$$

In which $L_{cls}(p, u) = -\log p_u$ is log loss for true class u. The second task loss, L_{loc}, is defined over a tuple of true bounding-box regression targets for class $u, v = (v_x, v_y, v_w, v_h)$, and a predicted $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$, again for class u. The Iverson bracket indicator function$[u \geq 1]$ evaluates to 1 when $[u \geq 1]$ and 0 otherwise. By convention the catch-all background class is labeled $u = 0$. For background RoIs there is no notion of a ground-truth bounding box and hence L_{loc} is ignored. For bounding-box regression, we use the loss

$$L_{loc}(t^u, v) = \sum_{i \epsilon \{x,y,w,h\}} Smooth_{L_1}(t_i^u - v_i) \qquad (2)$$

In which

$$Smooth_{L_1}(x) = \begin{cases} 0.5x^2 & if\ |x| < 1 \\ |x| - 0.5 & otherwise, \end{cases} \qquad (3)$$

is a robust L_1 loss that is less sensitive to outliers than the L_2 loss used in R-CNN and SPPnet. When the regression targets are unbounded, training with L_2 loss can require careful tuning of learning rates in order to prevent exploding gradients. Eq. 3 eliminates this sensitivity. The hyper-parameter λ in Eq. 1 controls the balance between the two task losses. We normalize the ground-truth regression targets v_i to have zero mean and unit variance. All experiments use = 1 .

2) Mini-batch sampling

During fine-tuning, each SGD mini-batch is constructed from N = 2 images, chosen uniformly at random (as is common practice, which actually is iterated over permutations of the dataset). Mini-batches of size R = 128, sampling 64 RoIs is used from each image. Total of 25% of the RoIs from object proposals that have intersection over union (IoU) overlap with a groundtruth bounding box of at least 0.5 are chosen. These RoIs comprise the examples labeled with a foreground object class, i.e. $[u \geq 1]$. The remaining RoIs are sampled from object proposals that have a maximum IoU with ground truth in the interval $[0.1, 0.5)$. These are the background examples and are labeled with $u = 0$. During training, images are horizontally flipped with probability 0.5. No other data augmentation is used.

3) Back-propagation through RoI pooling layers

Backpropagation routes derivatives through the RoI pooling layer. Let $x_i \in \mathbb{R}$ be the i-th activation input into the RoI pooling layer and let y_{rj} be the layer's j-th output from the r- th RoI. The RoI pooling layer computes $y_{rj} = x_{i*(r,j)}$, in which $i * (r,j) = argmax_{i'*(r,j)} x_{i'}. R_{(r,j)}$ is the index set of inputs in the sub-window over which the output unit y_{rj} max pools. A single x_i may be assigned to several different outputs y_{rj}.

The RoI pooling layer's backwards function computes partial derivative of the loss function with respect to each input variable xi by following the argmax switches:

$$\frac{\partial L}{\partial x_i} = \sum_r \sum_j [i = i * (r,j)] \frac{\partial L}{\partial y_{rj}} \qquad (4)$$

In words, for each mini-batch RoI r and for each pooling output unit y_{rj}, the partial derivative $\frac{\partial L}{\partial y_{rj}}$ is accumulated if i is the argmax selected for y_{rj} by max pooling. In back-propagation, the partial derivatives $\frac{\partial L}{\partial y_{rj}}$ are already computed by the backwards function of the layer on top of the RoI pooling layer.

4) SGD hyper-parameters.

The fully connected layers used for softmax classification and bounding-box regression are initialized from zero-mean Gaussian distributions with standard deviations 0.01 and 0.001, respectively. Biases are initialized to 0. All layers use a per-layer learning rate of 1 for weights and 2 for biases and a global learning rate of 0.001.

D. Scale invariance

To achieving scale invariant object detection image pyramids is used which provides approximate scale-invariance to the network through an image pyramid. At test-time, the image pyramid is used to approximately scale-normalize each object proposal.

E. Detection

Once a Fast R-CNN network is fine-tuned, the network takes as input an image and a list of R object proposals (≈ 2000) as input. When using an image pyramid, each RoI is assigned to the scale such that the scaled RoI is closest to 45x30 pixels in area. For each test RoI r, the forward pass outputs a class posterior probability distribution p and a set of predicted bounding-box offsets relative to r (each of the K classes gets its own refined bounding-box prediction). A detection confidence pr is assigned to r for each object class k using the estimated probability pk.

IV. IMPROVED FAST RCNN

Two modifications are made to Fast RCNN in order to improve the accuracy and speed of Fast RCNN. Modification made for Accuracy is named as Fast RCNN type 2 while Modification made for Speed is named as Fast RCNN type 3.

A. Fast RCNN type 2 - Accuracy Improvement

Original Fast RCNN having only two convolutional layers is shown in fig. 3 (top) while the modification made for accuracy in the form of additional convolutional layer is shown in fig 3 (bottom), encircled and highlighted.

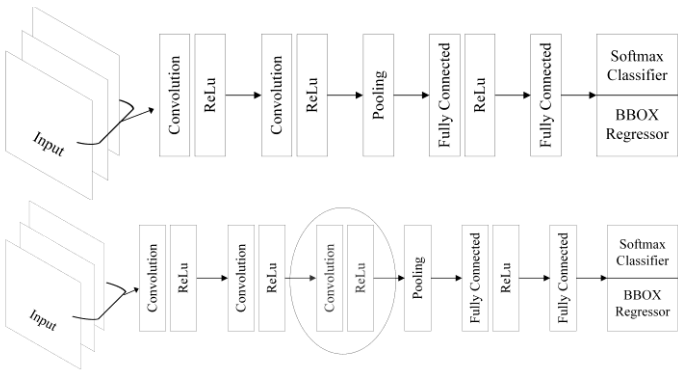

Fig. 3. Modification made to Fast RCNN for Accuracy

B. Fast RCNN type 3 - Speed Improvement

Original Fast RCNN having input channel size 3 (RGB) is shown in fig. 4 (top) while the modification made for speed in the form of reduction of input channel size to one (grayscale) shown in fig. 4 (bottom), encircled and highlighted.

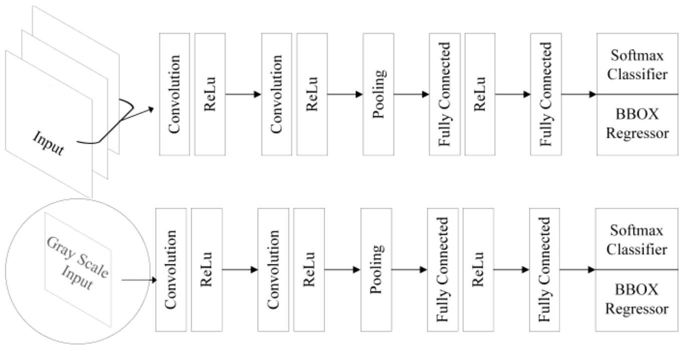

Fig. 4. Modification made to Fast RCNN for Speed

V. EXPERIMENTS AND RESULTS

A. DataSet

Dataset 03: OSU Thermal Database of OTCBVS Benchmark Dataset Collection [19] has been used for experiment. The dataset comprises of Thermal (Infrared) Images of two locations. Further each location has been divided into 3 subsets. Location 1 having subsets 1, 2 and 3 and location 2 having subsets 4, 5, 6. There are total 8544 grayscale bitmap images all of which have dimension equal to 320 x 240 pixels. The dataset has total 23210 RoIs i.e. pedestrians.

TABLE I. OTCBVS THERMAL DATASET 03: IMAGES AND ROIS

Location	Sequence No.	No. of Images (train/test)	No. of Pedestraians_RoI (train/test)
Loc-1	**1**	1054 (843/211)	6981 (5581/1400)
	2	601 (480/121)	2947 (2353/594)
	3	1700 (1360/340)	8791 (7028/1763)
Loc-2	**4**	1506 (1205/301)	745 (595/150)
	5	2031 (1624/407)	2638 (2110/528)
	6	1652 (1321/331)	1108 (886/222)
DataSet	**Total**	85433 (6833/1711)	23210 (18533/4657)

Out of 8544 images having 23210 pedestrians, 80% (6833 images having 18533 pedestrians) of the dataset are used for training and remaining 20% (1711 images having 4657 pedestrians) are used for testing the detector.

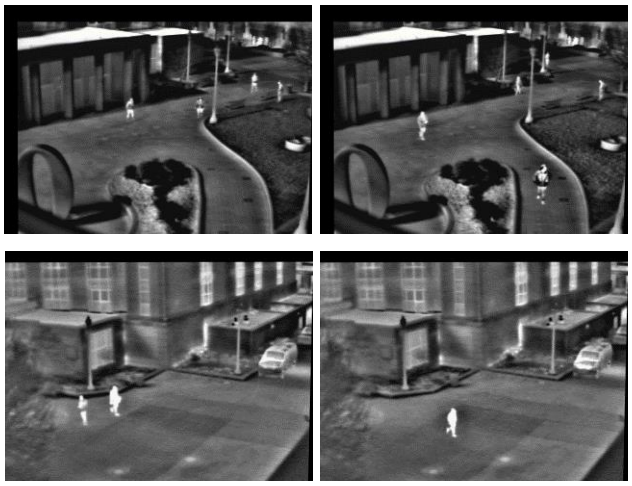

Fig. V. OTCBVS OSU Thermal Database 03

B. Experimental Setup

Fast RCNN code is implemented using Mathworks MATLAB 2017a. As Deep Learning needs a powerful processor, a GPU is required to run the algorithm or it would take days to train and test a simple algorithm even with a small dataset. For this experiment NVIDIA GeForce GT 740 GPU is used which has Compute capability of 3.0 and memory equal to 1 GB.

C. Network Layers

FastRCNN Type 1 has a total of 11 layers comprising of Input layer, Sevral convolution followed by ReLu, Fully connected layers Softmax classifier and Output layer. The detail about the layers is given in the Table II.

FastRCNN Type 2 all the layers of type 1 FastRCNN but with an additional 3rd Convolution Layer followed by ReLu layer which makes total of 13 layers as shown in Table III.

FastRCNN Type 3 has a total of 11 layers, same as type 1 FastRCNN but with a little modification of input channel size, instead of 3 channel input 1 channel has been used (input size = 32x32x1) See Table IV for Layers details.

D. Training Options

The following training options are used to train the network.

- Optimization Algorith: SGDM with momentum of 0.9
- Initial Learn Rate: 1.e-06
- Max Epoch: 32
- Mini Batch size: 128

TABLE II. TYPE 1 FAST RCNN LAYERS

Layer Name	Layer Description
1. Input Layer	32x32x3 images with 'zerocenter' normalization
2. Convolution Layer	32 filters, Kernel Size 3x3, with stride [1 1] and padding [1 1]
3. ReLu Layer	ReLU
4. Convolution Layer	32 filters, Kernel Size 3x3, with stride [1 1] and padding [1 1]
5. ReLu Layer	ReLU
6. Max Pooling Layer	3x3 max pooling with stride [2 2] and padding [0 0]
7. Fully Connected Layer	64 fully connected layer
8. ReLu Layer	ReLU
9. Fully Connected Layer	3 fully connected layer
10. Softmax Layer	softmax
11. Classification Output Layer	cross_entropyex with classes 'Ped' and 'Background'

TABLE III. TYPE 2 FASTRCNN LAYERS

Layer Name	Layer Description
1. Input Layer	32x32x3 images with 'zerocenter' normalization
2. Convolution Layer	32 filters, Kernel Size 3x3, with stride [1 1] and padding [1 1]
3. ReLu Layer	ReLU
4. Convolution Layer	32 filters, Kernel Size 3x3, with stride [1 1] and padding [1 1]
5. ReLu Layer	ReLU
6. Convolution Layer	**32 filters, Kernel Size 3x3, with stride [1 1] and padding [1 1]**
7. ReLu Layer	**ReLU**
8. Max Pooling Layer	3x3 max pooling with stride [2 2] and padding [0 0]
9. Fully Connected Layer	64 fully connected layer
10. ReLu Layer	ReLU
11. Fully Connected Layer	3 fully connected layer
12. Softmax Layer	softmax
13. Classification Output Layer	cross_entropyex with classes 'Ped' and 'Background'

TABLE IV. TYPE 3 FASTRCNN LAYERS

Layer Name	Layer Description
1. Input Layer	**32x32x1 images with 'zerocenter' normalization**
2. Convolution Layer	32 filters, Kernel Size 3x3, with stride [1 1] and padding [1 1]
3. ReLu Layer	ReLU
4. Convolution Layer	32 filters, Kernel Size 3x3 with stride [1 1] and padding [1 1]
5. ReLu Layer	ReLU
6. Max Pooling Layer	3x3 max pooling with stride [2 2] and padding [0 0]
7. Fully Connected Layer	64 fully connected layer
8. ReLu Layer	ReLU
9. Fully Connected Layer	3 fully connected layer
10. Softmax Layer	softmax
11. Classification Output Layer	cross_entropyex with classes 'Ped' and 'Background'

After generation of RoIs by using edge boxes algorithm (as shown in Fig. 6), the detector is trained using positive instances i.e. only those RoI are considered by the network for training whose boundary box has and IoU of 0.7 with the ground truth while the remaining RoIs which are labeled as background during the process are left out.

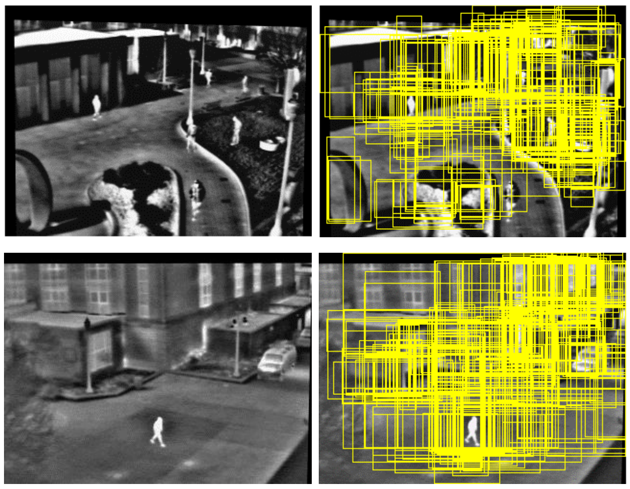

Fig. 6. RoI (Positive and Background Instances)

E. Detection

After separately training the detectors for all the subset of the dataset, the detectors are tested on test data. The results are shown in Tables V,VI,VII and VIII.

1) Fast RCNN Type 1 vs RCNN

The Detection percentage of FastRCNN and RCNN is shown in table V. While the time taken by training and testing calculated in seconds is shown in table VI.

TABLE V. TYPE 1 FASTRCNN VS RCNN DETECTION RATE

Subset Number	No. of Test Images	No. of Pedestrians (RoI)	Detection %	
			Fast RCNN_T1	RCNN
1	211	1400	51.3571	48.6428
2	121	594	41.9192	36.8686
3	340	1763	34.7136	25.5813
4	301	150	90.6670	81.3333
5	407	528	85.4167	78.9772
6	331	222	58.1080	55.8558
Total	**1711**	**4657**	**60.3636**	**54.54318**

TABLE VI. TYPE 1 FAST RCNN VS RCNN TRAINING AND TESTING SPEED

Subset Number	Training Speed (sec)		Testing Speed (sec)	
	Fast RCNN_T1	RCNN	Fast RCNN_T1	RCNN
1	765.751641	2122.318721	32.204502	84.0363340
2	138.760579	405.5073566	18.683158	47.9111250
3	638.202716	1803.784265	47.237358	126.350541
4	659.863006	2913.374721	36.559944	84.4541700
5	823.980966	3137.592315	52.764089	133.193406
6	264.390669	1404.776525	43.299481	102.494498
Avg_Time (sec)	**548.491596**	**1964.558984**	**38.458089**	**96.4066790**

The difference in detection rate in table V clearly indicates that the softmax classifier of Fast RCNN outperformed the SVM classifier of RCNN while from table VI it is evident that by processing the whole image first with several convolutional and max pooling layers to produce a conv feature map and later by using RoI pooling layer to extract fixed length feature vector from the feature map speeded up the detection process.

2) FastRCNN Type 2

The training and testing time along with detection percentage of FastRCNN type 2 is shown in table 5-7.

TABLE VII. TYPE 2 FASTRCNN RESULTS

Subset Number	Training Speed (sec)	Test Speed (sec)	Detection %
1	1161.83180	34.935577	55.6571
2	198.836582	20.021811	57.6082
3	966.818034	52.029601	35.3375
4	994.594042	39.964303	92.0000
5	1238.99810	57.309426	81.8182
6	377.540465	47.045292	58.1081
average	**823.103172**	**41.884335**	**63.4215**

Table VII detection column shows that modification made as an introduction of extra convolution layer to the architecture as shown in fig. 3, helped the network to learn more by extracting additional features, which improved the detection rate from 60.36% to 63.42%.

3) FastRCNN Type 3

The training and testing time along with detection percentage of FastRCNN type 3 is shown in table VIII.

TABLE VIII TYPE 3 FASTRCNN RESULTS

Subset Number	Training Speed (sec)	Test Speed (sec)	Detection %
1	752.862071	31.951306	43.7857
2	137.893466	18.584171	39.5623
3	630.406242	47.102926	45.7486
4	649.081506	36.508590	76.6667
5	800.540570	52.274447	65.5303
6	259.201657	43.200784	57.6577
average	**538.3309187**	**38.27037067**	**54.8252**

The advantage of using infrared image dataset for pedestrian detection is evident from the values of training and testing speed columns of table VIII. Lack of color information in infrared images lead to the modification as reduction of input channel from three (RGB) to one (grayscale) as shown in fig. 4 which helped in speeding up the whole process.

F. Results

1) Detection

Comparing the detection percentage of all algorithms for 4657 test pedestrians FastRCNN Type 2 has better detection rate with an average of 63.42% , followed by FastRCNN Type 1 which has detection rate of 60.36%. The least accurate is RCNN having detection rate of 54.54%. (see fig. 7)

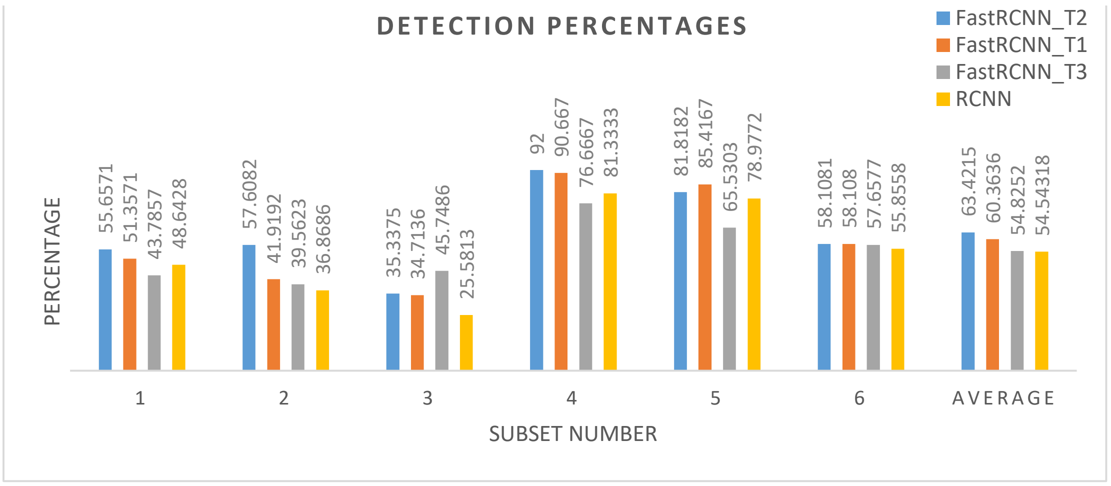

Fig. 7. Detection Percentages Comparison

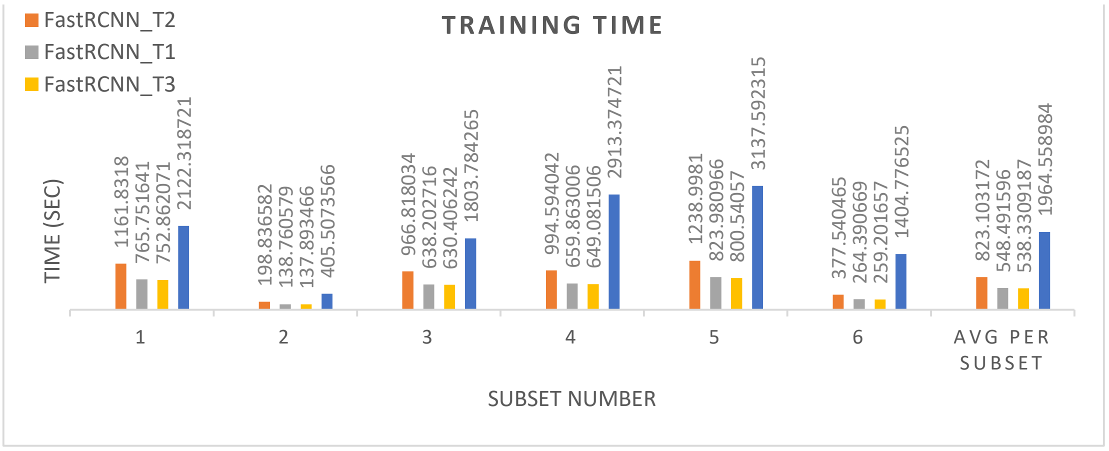

Fig. 8. Training Time Comparison

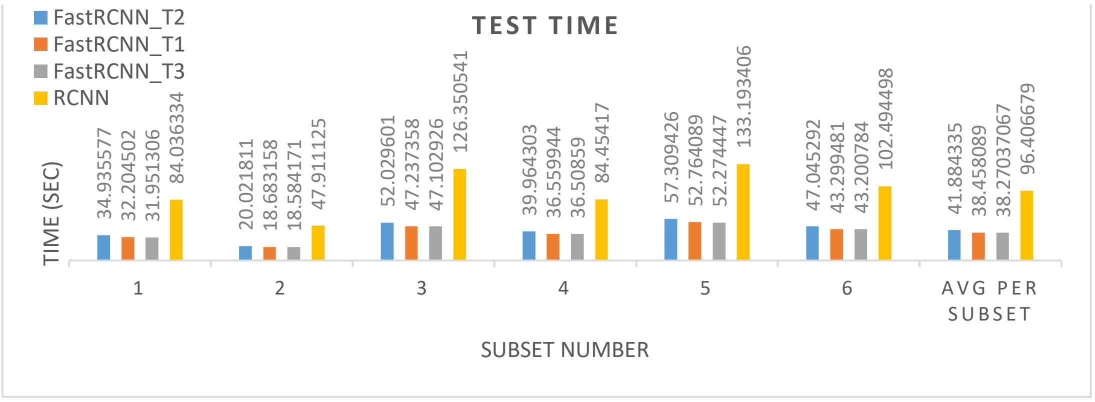

Fig.9. Test Time Comparison

2) Training Speed

Comparing training speed of all algorithms for 6833 training images having 18533 training pedestrians FastRCNN Type 3 is faster of all having average elapsed time of 538.33 sec, followed by Fast RCNN type 1 with 548.49 sec. The slowest (time consuming) is RCNN with 1964.55 sec. (see fig. 8)

3) Test Speed

Comparing Test speed of all algorithms for 1711 test images with 4657 test pedestrians FastRCNN Type 3 is faster of all having an average elapsed time of 38.27 sec, followed by FastRCNN type1 with 38.45 sec. RCNN with 96.40 is the most time consuming algorithm. (see fig. 9)

4) Detection Figures

After training the detector using the parameters given in section V.D figure 10 shows the detection results in which green boundary boxes indicates the detector has correctly detected a pedestrian while yellow boundary box shows that the detector has miss classified the pedestrian as non-pedestrian.

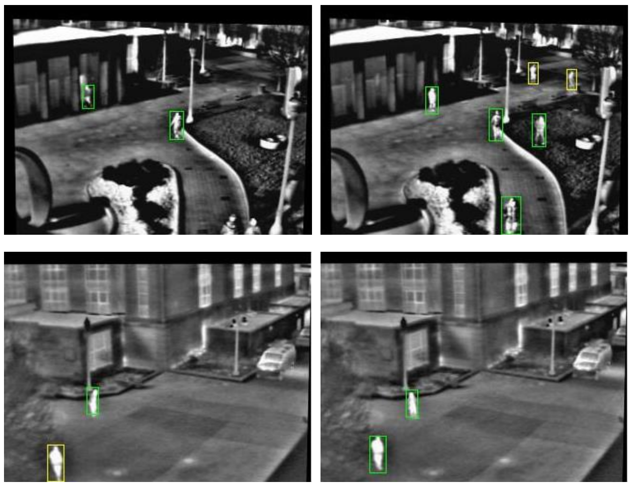
Fig. 10. Detection Results

VI. CONCLUSION

Taking advantage of characteristics that infrared images possess, Infrared images are used for pedestrian detection using Fast RCNN with a small network architecture having better accuracy and time as compared to RCNN. Furthermore the performance of Fast RCNN has been improved by two modifications made to the architecture, one for accuracy in which an extra convolutional layer was added to the the network and other for speed in which the input channel was reduced from three channel input to one.

REFERENCES

[1] Chaoxiang Chen, Wenshu Li, Hongting Li, Shiping Ye, "A Novel Pedestrian Detection in Infrared Images," *IJSSST*, vol. 17, no. 28, 2016.

[2] Jiabao Wang,Yafei Zhang, Jianjiang Lu and Yang Li, "Target Detection and Pedestrian Recognition in Infrared Images," *Journal of Computers*, vol. 8, no. 4, pp. 1050-1057, 2013.

[3] M Bertozzi · Alberto Broggi · Paolo Grisleri · T Graf · M M Meinecke, "Pedestrian detection in infrared images," *Information Visualization*, pp. 662-667, 2003.

[4] Lianna Chen, Wenshu Li, Zhengxi Xu, Limei Tang, "Pedestrian Detection Based on ISC in Infrared Images," in *international conference on networking*, 2012.

[5] Paul A Viola, Michael J Jones, "Robust real-time face detection," *international conference on computer vision*, vol. 57, no. 2, pp. 137-154, 2004.

[6] Navneet Dalal, Bill Triggs, "Histograms of oriented gradients for human detection," in *computer vision and pattern recognition*, 2005.

[7] Pedro F Felzenszwalb, David A Mcallester, Deva Ramanan, "A discriminatively trained, multiscale, deformable part model," in *computer vision and pattern recognition*, 2008.

[8] A. Krizhevsky, I. Sutskever, and G. Hinton, "ImageNet classification with deep convolutional neural networks," in *NIPS*, Lake Tohe, 2012.

[9] Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun, "OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks," in *Computer Vision and Pattern Recognition*, 2014.

[10] Ross B Girshick · Jeff Donahue · Trevor Darrell · Jitendra Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *computer vision and pattern recognition*, 2014.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," in *Computer Vision and Pattern Recognition*, 2014.

[12] J. Sivic and A. Zisserman, "Video google: a text retrieval approach to object matching in videos," in *International Conference on Computer Vision*, 2003.

[13] Svetlana Lazebnik · Cordelia Schmid · Jean Ponce, "Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories," in *computer vision and pattern recognition*, 2006.

[14] R. B. Girshick, "Fast R-CNN," in *international conference on computer vision*, 2015.

[15] C Lawrence Zitnick · Piotr Dollar, "Edge Boxes: Locating Object Proposals from Edges," in *European Conference on Computer Vision*, 2014.

[16] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *Internation Journal of Computer Vision,* 2013.

[17] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE, Transactions on Pattern Analysis and Machine Intelligence,* 2012.

[18] I. Endres and D. Hoiem., "Category independent object proposals.," *European Conference on Computer Vision,* 2010.

[19] J. Davis and V. Sharma, "Background-Subtraction using Contour-based Fusion of Thermal and Visible Imagery," *Computer Vision and Image Understanding*, vol. 106, no. 2-3, pp. 162-182, 2007.