future internet

MDPI

Article

Embedded Deep Learning for Ship Detection and Recognition

Hongwei Zhao, Weishan Zhang *, Haoyun Sun and Bing Xue

College of Computer and Communication Engineering, China University of Petroleum (UPC), Qingdao 266580, China; upcvagen@163.com (H.Z.); hys_upc@163.com (H.S.); xueb2016@163.com (B.X.)
* Correspondence: zhangws@upc.edu.cn

Received: 31 December 2018; Accepted: 31 January 2019; Published: 21 February 2019

Abstract: Ship detection and recognition are important for smart monitoring of ships in order to manage port resources effectively. However, this is challenging due to complex ship profiles, ship background, object occlusion, variations of weather and light conditions, and other issues. It is also expensive to transmit monitoring video in a whole, especially if the port is not in a rural area. In this paper, we propose an on-site processing approach, which is called Embedded Ship Detection and Recognition using Deep Learning (ESDR-DL). In ESDR-DL, the video stream is processed using embedded devices, and we design a two-stage neural network named DCNet, which is composed of a DNet for ship detection and a CNet for ship recognition, running on embedded devices. We have extensively evaluated ESDR-DL, including performance of accuracy and efficiency. The ESDR-DL is deployed at the Dongying port of China, which has been running for over a year and demonstrates that it can work reliably for practical usage.

Keywords: ship identification; fully convolutional network; embedded deep learning; scalability

1. Introduction

With the development of the marine economy, marine transportation and management have been attracting more and more attention in modern ports [1]. Ship detection and recognition play an important role for marine transportation management. To accomplish the task of ship detection and recognition, video surveillance with static cameras is a good choice. Surveillance cameras are increasingly deployed for port management and security in order to realize a smart port [2]. However, this is challenging due to complex ship profiles, ship background and object occlusions, variations of weather and light conditions, and other issues.

Deep learning [3] provides a promising technology to tackle these issues. Vehicle plate text recognition is a popular image process method for vehicle identification, which shows promising results for accurate object recognition. The work in [4] handled Chinese car license plate recognition from traffic videos with image features extracted by DCNNs (Deep Convolutional Neural Networks). License plate recognition [5] based on deep learning was also used for feature extraction and classification. This regular character recognition is much simpler than these Chinese characters from ship license plates, due to the usage of various character types and complex backgrounds, and also the variations of ship plate locations.

At the same time, the number of monitor devices can be big, deployed at both the seashore and above the water, which are used to monitor ships sailing in the water and also ships going back and forth from a port. Therefore, the recognition approach requires good scalability and should have the capability to handle a considerate number of video streams. On the other hand, the transmission of all video streams back may not be possible as there may not be Internet connections in some places,

and also the cost of using 4G for transmission of video streams is an important factor to design possible recognition solutions.

To address these challenges, we propose an embedded deep learning approach called ESDR-DL (Embedded Ship Detection and Recognition using Deep Learning), in order to conduct ship recognition on the fly by connecting the embedded device to a camera directly. In ESDR-DL, we propose a neural network named DCNet (composed with DNet and Cnet as detailed later) which conducts ship recognition as a classification problem by detecting and identifying key parts of a ship (the bow, the cabin, and the stern), and classifies the ship's identity based on these key parts. These classification results are then voted for the decision of the ship's identity. In order to boost performance, ESDR-DL is designed to handle multi-channel video at the same time. We conduct comprehensive evaluations for the embedded system of ESDR-DL, including the performance of accuracy and efficiency.

The contributions for this paper include:

- To decrease model parameters, we design a tiny network DNet for ship detection, and share the base convolutional layers with CNet.
- To address challenges of variations of ship license plate locations and text types, we propose a classification network CNet to recognize ships.
- We run the DCNet on embedded devices, which has good scalability and can handle a large number of video streams at the same time.

The remainder of the paper is organized as follows: Section 2 discusses the related work. Section 4 presents the architecture design of ESDR-DL. Section 3 discusses the implementation and training steps of DCNet. Section 5 presents comprehensive evaluations of the deployed solution. Section 6 concludes the paper.

2. Related Work

We will discuss deep learning and embedded device supported object recognition as ESDR-DL is in principle an embedded deep learning approach, we will also discuss vehicle recognition as ship is also a kind of vehicle.

2.1. Deep Learning

The concept of deep learning originates from the study of artificial neural networks, proposed by Hinton et al. [6]. Deep learning have made remarkable achievements in the field of image processing, especially for object detection. SSD is a typical one stage detector proposed in [7], which processes images in a single network, and and has good efficiency and accuracy. Faster R-CNN [8] is a two-stage detector, which uses RPN (Region Proposal Network) to produce high-quality region proposals and then detect them with Fast R-CNN [9].

Redmon presents a single neural network named YOLO, which abandons anchor boxes, and predicts bounding boxes and class probabilities directly from a full image in one evaluation [10]. YOLO considers object detection as a regression problem to predict bounding boxes and class probabilities. It can be optimized as end-to-end directly with good detection performance. Fast YOLO can process 155 frames per second. Compared with other state-of-the-art detection algorithms, YOLO makes more localization errors.

YOLOV2 [11] is based on YOLO [10]. YOLOV2 removes the fully connected layers from YOLO and uses anchor boxes to predict bounding boxes. The YOLOV2 model can run with various image sizes, and it is easy to make a trade-off between speed and accuracy. YOLOV2 is faster than YOLO, which can process 200 frames per second with the Tiny model. Table 1 shows the performance of these algorithms.

Table 1. The performance of the algorithms.

Algorithm	Datasets	mAP	FPS	Proposed Year
SSD300	VOC 2007 + 2012	74.3	46	2016
SSD500	VOC 2007 + 2012	76.8	19	2016
Faster-RCNN	VOC 2007 + 2012	73.2	31	2015
YOLO	VOC 2007 + 2012	63.4	45	2015
YOLOv2	VOC 2007 + 2012	76.8	67	2016
Tiny YOLO	VOC 2007 + 2012	57.1	207	2016

2.2. Vehicle Identification

There are many state-of-the-art algorithms that can be used for vehicle detection, such as RCNN [9] YOLO [10], which have high real-time performance, but the accuracy is not high for ship recognition. In [12], Wang proposed a vehicle real-time detection algorithm based on YOLOv2. It optimized important parameters of the model, and improved the number and size of anchors in the model, which can achieve both real-time and high accuracy for vehicle detection. It tested by a home-made dataset, which showed higher accuracy and ran faster than YOLOv2 [11] and RCNN. However, the real-time performance is based on high-performance equipment, which is not suitable for us.

Plate recognition is the most typical application for vehicle identification. Liu et al. [13] proposed CogniMem, which used a neural-network chip to recognize license plates. The CogniMem combined a video image processing module with a neural network module by using an equalized image processing algorithm and network classification algorithm. It contained processes of license location, character segmentation and character recognition. CogniMem can recognize car plates with low error; however, it required that the plates have a fixed character position and limited character type and numbers. Lin [14] proposed a method named ALPR to detect and recognize the characters in the plate region of an image. The approach is not applicable to the situation in which new targets emerge that are not annotated in its database.

2.3. Embedded Object Recognition

Embedded image processing has been attracting a lot of efforts. In [15], Arth et al. designed a full-featured license plate detection and recognition method using DSP. The processing core is a single Texas Instruments fixed point DSP with 1 MB RAM. Additionally, a slower SDRAM memory block of 16 MB exists. It can achieve real-time performance. In addition, Kamat and Ganesan [16] implemented a license plate detection system on a DSP using the Hough transform. Kang et al. [17] implemented a vehicle tracking and license plate recognition system on a PDA. An FPGA was used by Bellas et al. [18] to speed up parts of their license plate recognition system.

There was research that ran the Fast R-CNN on Jetson TK1 platform [19]. Although additional modifications on the Fast R-CNN were made to fit TK1, the detection speed was very low (1.85 frames per second-fps). The work in [20] ran a seven-layer CNN on TDA3x SoC for object classification, and the overall system performance was 15 fps. Therefore, a powerful software/hardware platform is needed to support efficient embedded deep learning based real-time video processing.

3. Designing a Recognition Neural Network-DCNet

DCNet is a two-stage network that consists of a DNet and a CNet as shown in Figure 1. DNet is a fully convolutional network [21] for ship parts detection including ship bow, cabin and stern. CNet is a classifier that can takes an image of any size and output a set of classification scores. We locate the ship parts from the DNet, and feed them into the CNet to get three classification scores (bow score, cabin score, stern score) of ship identify. Finally, a voter is used to recognize the ship as shown in Figure 1.

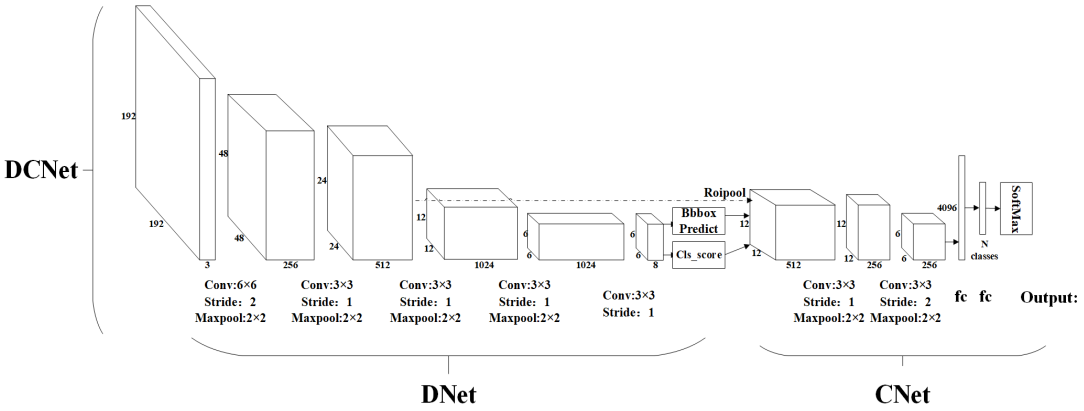

Figure 1. DCNet.

3.1. DNet

Region Proposal is one of the key points for a target detection network, such as Faster R-CNN [8] using RPN to generate better regions, and YOLOv1 [10] splitting an image into grid cells as region proposals to improve the detection efficiency. As shown in Figure 2, DNet divides the input image into 6×6 grid cells as region proposals like YOLOv1 [10]. Each region proposal consists of eight predictions: $x, y, w, h, c, C \times 3$. The (x, y) coordinates represent the center of the predicted box. w and h represent the width and height of the predicted box. c represents the IOU (intersection-over-union) between the predicted box and ground truth box. $C \times 3$ represents the probability classes of bow, cabin and stern.

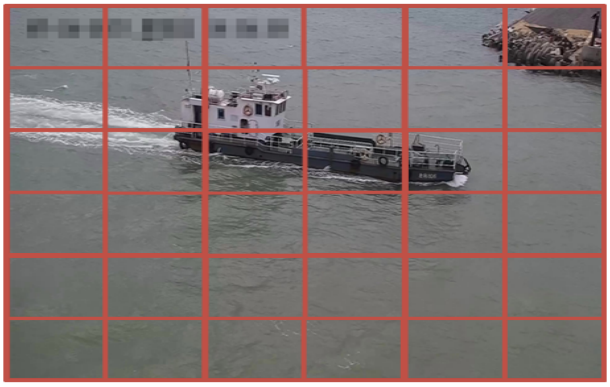

Figure 2. Region proposal of DNet.

Inspired by YOLOv1, DNet models the detection as a regression problem. Since the object and background are relatively simple, object features are relatively obvious, and the network is not as deep as VGGNet [10] and ResNet [22]. We pay more attention to the decreasing network model parameters. DNet resizes the image to 192×192 as the input and we design five layers to extract features from an image; the last two layers predict the object probabilities and its coordinates.

DNet predicts bounding boxes based on grid cells. A grid cell produces one bounding box predictor. We need one bounding box predictor to be "responsible" for each object, and choose the one based on which prediction has the highest current IOU (intersection-over-union) with the ground truth. To choose a proper predictor for each object, at training time, we design the loss L_g function as follows:

$$L_{grid} = \sum_{i=0}^{S^2} (C_i - C_i^*)^2, \tag{1}$$

where S^2 is the number of the grid cells, C_i is the confidence value that the predicting box contain an object, and C_i^* is the IOU between predicted bounding box with ground truth. If there is no object in predictions, then $C_i^* = 0$.

The final layer predicts both class probabilities and bounding box coordinates; we calculate the coordinates loss and classification loss only when the predictor is a proper one, the loss function is:

$$L_{box} = \sum_{c \in propers} \sqrt{(x_c - x_c^*)^2 + (y_c - y_c^*)^2 + (w_c - w_c^*)^2 + (h_c - h_c^*)^2} + (p_c - p_c^*)^2, \quad (2)$$

where p_c is the predicting class and p_c^* is the truth class. The loss of L_{box} is under the assumption that the predictor is a proper one. Therefore, it may not be ideal to weight the L_{grid} equally to L_{box}. We use λ to weight the loss, and the final loss function is designed as follows:

$$L = \lambda L_{grid} + (1 - \lambda) L_{box}. \quad (3)$$

3.2. CNet

Ship recognition is challenging, and we can make use of the fact that there is only a limited number of ships in a port. CNet model considers the recognition problem as a classification problem, which is connected to the end of the DNet. We set the output boxes and classes as the input and share the first three layers' feature maps of the DNet. The boxes is resized to 14×14 by a ROIPool layer as shown in Figure 3, which was proposed in [8]. Two extra convolutional layers followed by the ROIPool layer are added, and, finally, two fully connected layers and a softmax layer are used to predict the output probabilities.

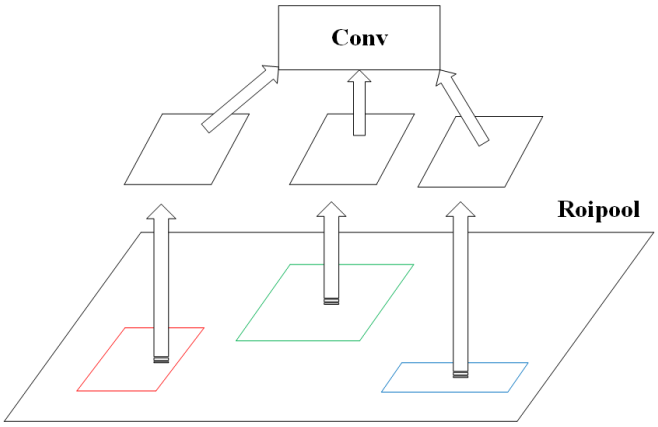

Figure 3. ROIPool Layer.

Finally, CNet outputs three ship classification scores of bow, cabin and stern. We design the voting strategy as

$$S_{i:i \in (probabilities)} = \lambda_b Score^i_{bow} + \lambda_c Score^i_{cabin} + \lambda_s Score^i_{stern}.$$

$Score^i$ denotes the output score of probabilities i. It weights the score of cabins equally with scores of bow and stern which may not be ideal. To resolve this, we use λ to weight the scores.

3.3. Training and Running

Before training, we have to label the ship data set. We quadrangle the bow, cabin and stern with (*c*,*i*,*x*,*y*,*w*,*h*). *c* represents the key point of ships, and *i* represents the identification of the key point, (*x*,*y*) represents the upper-left coordinates of the box, *w* represents the with of the box and *h* represent the hight of the box. To learn the shared features, we train the DCNet with two steps as shown in Figure 4. In the first step, we train the DCNet use the ship data set, we set the initial learning rates as 0.01 and decrease by one tenth per 10,000 iterations; after 50,000 iterations, the losses tend to stabilize.

In the second step, we fix the shared convolutional layers and only fine-tune the unique layers of CNet. During CNet training, we feed the ship data sets to the shared convolutional layers and rectangle the box feature maps, unify the box feature maps size by the ROIPool layer, and, lastly, classify the feature maps with the unique layers of CNet. We set the initial learning rates as 0.1 and decrease by one-tenth per 5000 iterations; after 40,000 iterations, the loss tends to be stabilized.

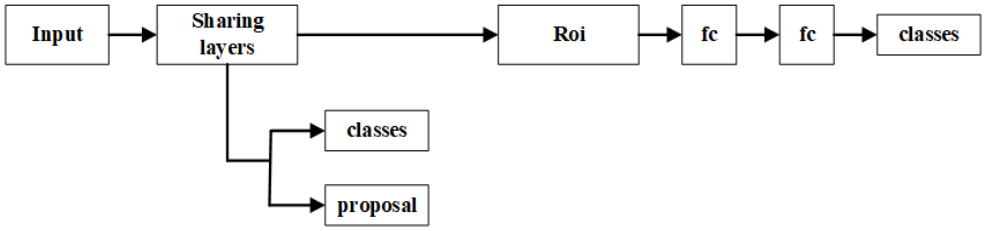

Figure 4. DCNet Training.

When running the model, firstly, the DNet predicts the coordinates and classes of bow, cabin and stern, and then it rectangles key ship parts from the sharing feature maps and feeds them to the CNet to get the probability scores, as shown in Figure 5.

Figure 5. Labels of bow, cabin and stern.

4. Architecture Design of ESDR-DL

In order to reduce network traffic caused by video streaming from surveillance cameras, and resolve the limitation of low transmission bandwidth, we design an embedded architecture for deep learning, which connects surveillance cameras and performs image processing at the front end, as shown in Figure 6. In this ESDR-DL, the video stream is connected to a nearby TX2 through a LAN. To ensure real-time performance of video surveillance, each TX2 receives only one or two video streams. When the system is running, a Video Stream Receiver in TX2 is responsible for receiving the video stream accessed by the current device, decoding the video stream through a Video Stream Decoder, and inputting the decoded images to an Image Processor for detection. In the Image Processor, the DCNet model is used to detect and identify key parts of a ship (the bow, the cabin, and the stern), and classify the ship's identity based on these key parts, and output three prediction results. These prediction results are then used in a Voter for the decision of the ship's identity.

We use NVIDIA Jetson TX2 as it is an industry-leading embedded computing device. Table 2 lists the main properties of TX2 related to our work in this paper.

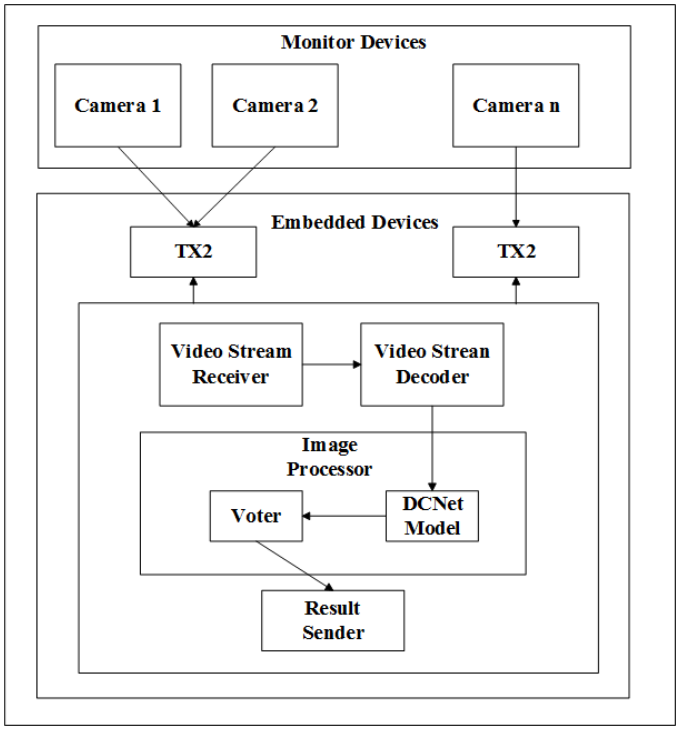

Figure 6. System architecture of ESDR-DL.

Table 2. Jetson TX2.

Versions	Jeston TX2
GPU	NVIDIA Pascal 256 CUDA core
CPU	64-bit Denver 2 and A57 CPUs
Memory	8 GB 128-bit LPDDR4
Storage	32 GB eMMC
Video Encode	4 K × 2 K 60 Hz
Video Decode	4 K × 2 K 60 Hz
Camera	1.4 Gpix/s 2.5 Gbps per lane
Connectivity	1 Gigabit Ethernet, 802.11ac WLAN

5. Experiment Results

We use the recall-R and precision-P as the evaluation standard, defined as

$$R = TP/(TP + FN)\ ,\ P = TP/(TP + FP).$$

TP refers to true positive, FN indicates false negative, and FP means false positive.

5.1. Algorithm Performance

To evaluate the performance of DNet, we use a ship data set that has 6000 images collected from Donging port, Shandong, China. We have tested both Yolo Tiny and DNet, running on TX2 and GTX TITAN X. In addition, 4700 images are used for training and 1300 are used for testing. Table 3 shows the test results. We can see that DNet achieves much higher energy efficiency with a little lower accuracy.

Table 3. Test results of Tiny YOLO and DNet.

Method	Device	Accuracy	Efficiency (FPS)	Power (w)	Energy Efficiency (fps/w)
Tiny YOLO	TX2	0.9316	18.91	7.84	2.41
DNet	TX2	0.9233	34.87	7.73	4.51
Tiny YOLO	GTX TITAN X	0.9492	155.24	180	0.86
DNet	GTX TITAN X	0.9297	298.43	177	1.68

YOLOv1 splits an image into 7×7 grid cells; considering the big target of ship and the limitation of computing capacity of a TX2, we decrease the grid cells to reduce model parameters. As is shown in Table 4 where efficiency is measured by FPS (Frames Per Seconds), a test is made to check this, and DNet splits an image into 6×6, considering the performance–accuracy trade off.

Table 4. Grid cells number test for DNet.

Grid Cells Number	Detection Accuracy	Efficiency
4×4	0.7642	59
5×5	0.8310	51
6×6	0.9233	43
7×7	0.9251	38
8×8	0.9282	32
9×9	0.9263	21

The λ for loss L in Equation (3) can be changed based on different scenarios and targets. We adjust λ experimentally and the results is shown in Table 5. Concluding from the tests, we set $\lambda = 0.7$.

Table 5. λ test for Loss L.

λ	Detection Accuracy
0.1	0.5216
0.2	0.7442
0.3	0.7513
0.4	0.8121
0.5	0.9021
0.6	0.9113
0.7	0.9233
0.8	0.9035
0.9	0.8945

We adjust λ experimentally for voting strategy and test its impact on accuracy as in Table 6. We can conclude the weights from it that $\lambda_c > \lambda_b > \lambda_s$. We set $\lambda_b = 0.3$, $\lambda_c = 0.5$ and $\lambda_s = 0.2$.

Table 6. λ impacts on recognition accuracy.

λ_b	λ_c	λ_s	Accuracy
1	0	0	0.83
0	1	0	0.85
0	0	1	0.80
0.33	0.34	0.33	0.84
0.25	0.5	0.25	0.85
0.3	0.5	0.2	0.86
0.2	0.5	0.3	0.85

5.2. System Performance

ESDR-DL is deployed to Dongying port, China. The video cameras used are Hikvision DS-2CD3T25D-I5. The pixel used is 1920 × 1080 and the frame rate is 30 fps. We use seven TX2s for 10 cameras as shown in Table 7. Four cameras are installed on both sides of the entrance with a height of 8 m. Others are installed inside the port.

Table 7. Deployment of cameras and TX2s.

TX2-1	entrance camera-1
TX2-2	entrance camera-2
TX2-3	entrance camera-3
TX2-3	entrance camera-4
TX2-4	inside port camera-1
TX2-5	inside port camera-2
TX2-6	inside port camera-3
TX2-6	inside port camera-4
TX2-7	inside port camera-5
TX2-7	inside port camera-6

During one month's running, we collect 13,000 recognized records and checks the accuracy manually. There are a total of 14,536 ships in videos.

Table 8 illustrates the recall and precision rates of ship detection and recognition. S denotes the ship number occurring in each camera, D-P stands for the ship detection precision, D-R is the detection recall, R-P is the recognition precision, R-R refers to the recognition recall, and T denotes the processing efficiency of each camera.

Table 8. Performance of ESDR-DL.

Camera	S	D-P	D-R	R-P	R-R	T
entrance camera-1	903	0.86	0.80	0.82	0.74	27 fps
entrance camera-2	903	0.86	0.79	0.82	0.74	27 fps
entrance camera-3	887	0.85	0.79	0.81	0.74	27 fps
entrance camera-4	891	0.86	0.80	0.80	0.73	27 fps
inside port camera-1	1532	0.89	0.84	0.84	0.79	13 fps
inside port camera-2	1129	0.87	0.82	0.80	0.75	13 fps
inside port camera-3	1413	0.90	0.85	0.85	0.78	13 fps
inside port camera-4	1410	0.89	0.84	0.84	0.79	13 fps
inside port camera-5	1611	0.89	0.85	0.82	0.79	13 fps
inside port camera-6	1611	0.88	0.85	0.82	0.80	13 fps

Comparing Tables 6 and 8, we can find that the accuracy of actual running is lower than the accuracy testing in the home-made data set because there are new ships arriving the port and the ESDR-DL can not recognize these new ships. In addition, ESDR-DL performs better for the inside-port monitoring cameras because there are some far away scenes of ships from the entrance cameras and only close scenes exist from the inside-port cameras, while DCNet focus on big target detection and recognition. In addition, as shown in Figure 7, the system can run in bad weather conditions (such as rain and smog) in practice. In order to test the performance of the system in bad weather, we run the system in rain and smog weather, and run it at dusk (5:00 p.m.–6:00 p.m.). The detection recognition results are shown in Table 9.

The recognition results are shown in Table 10. We can see that the accuracy of the system is dropping sharply in rain and smoggy weather, while performing well at dusk. This is not a problem in practice as there are very few ships in such weather conditions.

Table 9. Performance of ship detection in bad weather.

Camera	Rain-S	Rain-P	Ran-R	Smog-S	Smog-P	Smog-R	Dusk-S	Dusk-P	Dusk-R
entrance camera-1	46	0.72	0.65	114	0.61	0.52	203	0.83	078
entrance camera-2	46	0.74	0.65	114	0.59	0.51	203	0.87	0.80
entrance camera-3	39	0.69	0.56	99	0.49	0.39	211	0.85	0.79
entrance camera-4	40	0.70	0.55	103	0.48	0.40	225	0.83	0.80
inside port camera-1	70	0.75	0.70	91	0.64	0.51	293	0.82	0.76
inside port camera-2	58	0.71	0.64	69	0.60	0.55	233	0.80	0.71
inside port camera-3	132	0.79	0.71	155	0.70	0.53	254	0.83	0.79
inside port camera-4	140	0.74	0.70	169	0.67	0.59	254	0.85	0.80
inside port camera-5	129	0.78	0.73	143	0.72	0.56	223	0.85	0.76
inside port camera-6	129	0.75	0.71	143	0.70	0.51	223	0.80	0.77

Table 10. Performance of ship recognition in bad weather.

Camera	Rain-S	Rain-P	Ran-R	Smog-S	Smog-P	Smog-R	Dusk-S	Dusk-P	Dusk-R
entrance camera-1	46	0.53	0.45	114	0.36	0.20	203	0.79	0.72
entrance camera-2	46	0.45	0.38	114	0.31	0.18	203	0.81	0.75
entrance camera-3	39	0.46	0.36	99	0.29	0.21	211	0.82	0.74
entrance camera-4	40	0.45	0.32	103	0.26	0.15	225	0.83	0.73
inside port camera-1	70	0.55	0.41	91	0.34	0.21	293	0.80	0.72
inside port camera-2	58	0.52	0.45	69	0.29	0.17	233	0.75	0.70
inside port camera-3	132	0.49	0.39	155	0.39	0.23	254	0.80	0.74
inside port camera-4	140	0.54	0.45	169	0.35	0.19	254	0.82	0.76
inside port camera-5	129	0.48	0.43	143	0.32	0.24	223	0.80	0.75
inside port camera-6	129	0.51	0.41	143	0.25	0.12	223	0.84	0.75

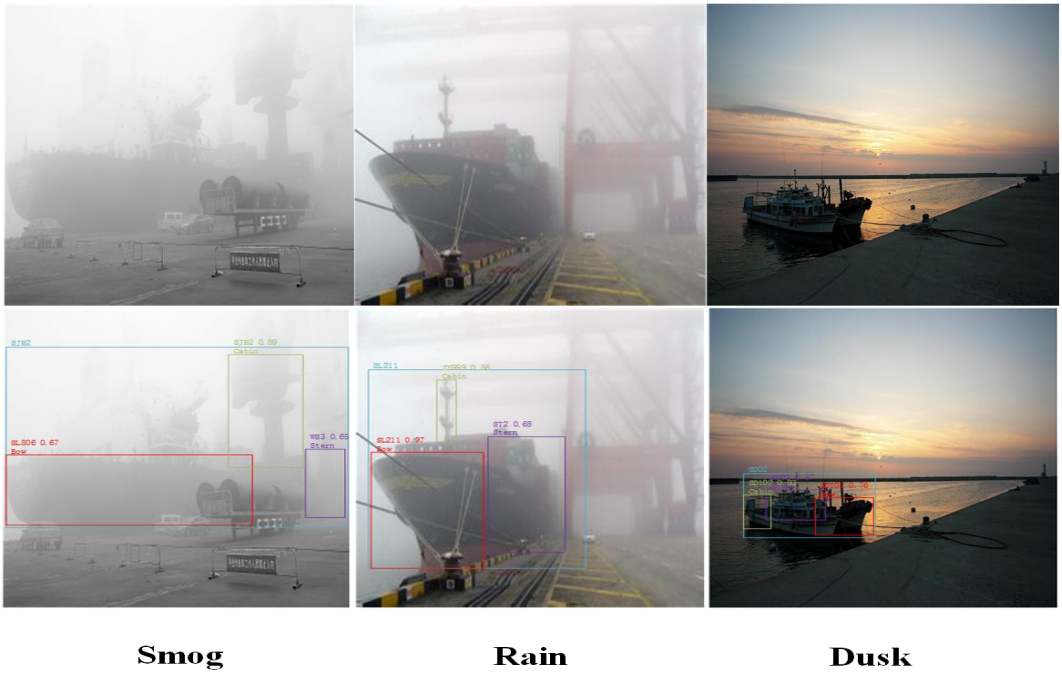

Figure 7. Ships in bad weather: The top is the ships in bad weather, and the bottom is the process results.

6. Conclusions

Considering the challenges of ship detection and recognition, this paper proposes an embedded deep learning system for ship detection and recognition named ESDR-DL. It first locates the bow, cabin and stern of the ship using DNet, and then recognizes them by a classification network named CNet. Finally, voting is used to recognize the ship identification. We implement the ESDR-DL with an embedded architecture which supports real-time video processing. We have deployed ESDR-DL at

Dongying port, China. It has been running stably in the past year, which shows the effectiveness of our solution. In the future, we will adopt a multi-model data fusion approach [23,24] to improve the recognition accuracy.

Author Contributions: Conceptualization, H.Z. and W.Z.; methodology, H.Z.; validation, H.Z., H.S. and B.X.; formal analysis, W.S.; investigation, B.X.; resources, W.Z.; writing–original draft preparation, H.Z.; writing–review and editing, H.Z. and H.S.; visualization, W.Z. and B.X.; project administration, W.Z.

Funding: This research was funded by the Key Research Program of Shandong Province under Grant No. 2017GGX10140 and the National Natural Science Foundation of China under Grant No. 61309024.

Conflicts of Interest: The authors declare no conflicts of interest.

Nomenclature

The following abbreviations are used in this manuscript:

ESDR-DL	Embedded Ship Detection and Recognition using Deep Learning
DCNet	Detection and Classification Network
DNet	Detection Network
CNet	Classification Network
DCNN	Deep Convolutional Neural Network
RPN	Region Proposal Network
RCNN	Region-based Convolutional Neural Network
ROI	Region Of Interest
DSP	Digital Signal Processing
PDA	Personal Digital Assistant
FPGA	Field-Programmable Gate Array
TDA3x SoC	Threat Discovery Appliance
SSD	Single Shot MultiBox Detector
YOLO	You Only Look Once

References

1. Wang, Z.; Tang, W.; Zhao, L. Research on the modern port logistics development in the city-group, China. In Proceedings of the 2010 International Conference on IEEE Logistics Systems and Intelligent Management (ICLSIM), Harbin, China, 9–10 January 2010; pp. 1280–1283.
2. Alderton, P.M. *Port Management and Operations*; Harbors: Suffolk, NY, USA, 2008.
3. Xu, L.; Ren, J.S.J.; Liu, C.; Jia, J. Deep convolutional neural network for image deconvolution. In Proceedings of the International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 8–13 December 2014; pp. 1790–1798.
4. Zang, D.; Chai, Z.; Zhang, J.; Zhang, D.; Cheng, J. Vehicle license plate recognition using visual attention model and deep learning. *J. Electron. Imaging* **2015**, *24*, 033001. [CrossRef]
5. Masood, S.Z.; Shu, G.; Dehghan, A.; Ortiz, E.G. License Plate Detection and Recognition Using Deeply Learned Convolutional Neural Networks. *arXiv* **2017**, arXiv:1703.07330.
6. Krizhevsky, A.; Sutskever, I.; Hinton, G.E. ImageNet classification with deep convolutional neural networks. In Proceedings of the International Conference on Neural Information Processing Systems, Lake Tahoe, NV, USA, 3–6 December 2012.
7. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In Proceedings of the European Conference on Computer Vision, Amsterdam, The Netherlands, 8–16 October 2016; pp. 21–37.
8. Ren, S.; He, K.; Girshick, R.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2015**, *39*. [CrossRef] [PubMed]
9. Girshick, R. Fast r-cnn. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015. [CrossRef]
10. Redmon, J.; Divvala, S.K.; Girshick, R.B.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. *arXiv* **2015**, arXiv:1506.02640v5,

11. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. *arXiv* **2016**, arXiv:1612.08242.
12. Wang, H.; Zhang, Z. A vehicle real-time detection algorithm based on YOLOv2 framework. In Proceedings of the Real-Time Image and Video Processing, Orlando, FL, USA, 15–19 April 2018; p. 22.
13. Liu, Y.; Wei, D.; Zhang, N.; Zhao, M. Vehicle-license-plate recognition based on neural networks. In Proceedings of the 2011 IEEE International Conference on Information and Automation, Shenzhen, China, 6–8 June 2011; pp. 363–366.
14. Lin, D.; Lin, F.; Lv, Y.; Cai, F.; Cao, D. Chinese Character CAPTCHA Recognition and Performance Estimation via Deep Neural Network. *Neurocomputing* **2018**, *28*, 11–19. [CrossRef]
15. Arth, C.; Limberger, F.; Bischof, H. Real-Time License Plate Recognition on an Embedded DSP-Platform. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition 2007, Minneapolis, MN, USA, 17–22 June 2007. [CrossRef]
16. Kamat, V.; Ganesan, S. An efficient implementation of the Hough transform for detecting vehicle license plates using DSP'S. In Proceedings of the Real-Time Technology and Applications Symposium, Chicago, IL, USA, 15–17 May 1995; pp. 58–59.
17. Kang, J.S.; Kang, M.H.; Park, C.H.; Kim, J.H.; Choi, Y.S. Implementation of embedded system for vehicle tracking and license plates recognition using spatial relative distance. In Proceedings of the International Conference on Information Technology Interfaces, Cavtat, Croatia, 7–10 June 2003; Volume 1, pp. 167–172.
18. Bellas, N.; Chai, S.M.; Dwyer, M.; Linzmeier, D. FPGA implementation of a license plate recognition SoC using automatically generated streaming accelerators. In Proceedings of the 20th IEEE International Parallel & Distributed Processing Symposium, Rhodes Island, Greece, 25–29 April 2006.
19. Mao, H.; Yao, S.; Tang, T.; Li, B.; Yao, J.; Wang, Y. Towards real-time object detection on embedded systems. *IEEE Trans. Emerg. Top. Comput.* **2016**, *6*, 417–431. [CrossRef]
20. Jagannathan, S.; Desappan, K.; Swami, P.; Mathew, M.; Nagori, S.; Chitnis, K.; Marathe, Y.; Poddar, D.; Narayanan, S. Efficient object detection and classification on low power embedded systems. In Proceedings of the 2017 IEEE International Conference on Consumer Electronics (ICCE), Las Vegas, NV, USA, 8–10 January 2017; pp. 233–234.
21. Long, J.; Shelhamer, E.; Darrell, T. Fully convolutional networks for semantic segmentation. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Boston, MA, USA, 7–12 June 2015; pp. 3431–3440, doi:10.1109/CVPR.2015.7298965. [CrossRef]
22. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
23. Zhang, W.; Zhang, Y.; Zhai, J.; Zhao, D.; Xu, L.; Zhou, J.; Li, Z.; Yang, S. Multi-source data fusion using deep learning for smart refrigerators. *Comput. Ind.* **2018**, *95*, 15–21. [CrossRef]
24. Zhang, W.; Wang, Z.; Liu, X.; Gong, W.; Sun, H.; Zhou, J.; Liu, Y. Deep Learning based Real-Time Fine-grained Pedestrian Recognition using Stream Processing. *IET Intell. Transp. Syst.* **2018**, *12*. [CrossRef]