UBFC
UNIVERSITÉ
BOURGOGNE FRANCHE-COMTÉ

UNIVERSITÉ DE BOURGOGNE

THÈSE DE DOCTORAT DE L'ÉTABLISSEMENT UNIVERSITÉ BOURGOGNE FRANCHE-COMTÉ

PRÉPARÉE À L'UNIVERSITÉ DE BOURGOGNE

École doctorale n°37

Sciences Pour l'Ingénieur et Microtechniques

Doctorat d'Instrumentation et informatique de l'image

par

GONG SONGCHENCHEN

Implementation of real time reconfigurable embedded architecture for people counting in a crowd area

Thèse présentée et soutenue à Dijon, le 13 November 2020

Composition du Jury :

BOURENNANE EL-BAY	Professeur à l'Université de Bourgogne Franche Comté Faculté des Sciences Mirande, France	Directeur de thèse
YANG FAN	Professeur à l'Université de Bourgogne Franche Comté Faculté des Sciences Mirande, France	Examinateur
KECHADI M-TAHAR	Professeur à l'Université College Dublin (UCD) École d'informatique, Ireland	Rapporteur
RABAH HASSAN	Professeur à l'Université de Lorraine Responsable du groupe MAE, Institut Jean Lamour-UMR, France	Rapporteur

N° 0 0 1

école doctorale **sciences pour l'ingénieur et microtechniques**

Title: Implementation of real time reconfigurable embedded architecture for people counting in a crowd area

Keywords: Texture features,Edge detection, M-MCNN, FPGA

Abstract:

The crowd counting task is an important research problem. Now more and more people are concerned about safety issues. Considering the scenario of a crowded scene: a population density system analyzes the crowds and triggers a warning to divert the crowds when their population density exceeds a normal range. With such a system, the incident of the Shanghai New Year's stampede will not happen again. The most difficult problem of population counting at present: On the one hand, in the densely populated area, how to make the model distinguish human head features more finely, such as head overlap. The second aspect is to find a small-scale local head feature in an image with a wide range of population density. The most critical aspect, in some public places, is that we can not install an intelligent video surveillance system. So how do we estimate the high-density crowd area to avoid crowd trampling accidents? Facing these challenges, we propose implementation of real time reconfigurable embedded architecture for people counting in a crowd area. First, our work integrates the features of HOG and LBP, which not only combines the effective identification information of multiple features, but also eliminates most of the redundant information, thereby realizing effective compression of information, saving information storage space. Then, in terms of crowd counting, we use multiple sources of information, namely HOG, LBP and CANNY based filtering. These sources provide separate estimates of the number of counts and other statistical measures, through the support vector Machine SVM, classification. At the same time, in order to effectively solve the problem of extracting scale-related features in crowd counting. We propose a new framework M-MCNN based on MCNN for crowd counting on any single image. M-MCNN not only contains the original three columns of convolutional neural networks with different filter sizes, but replaces the fully connected layers with a convolutional layer of 1*1 filters, so the input image of the model can be of any size. Moreover, in a single individual sample, we greatly improve the learning of sample features by extracting the texture features of a single human head , and better use it for datasets. Finally, we implement our new framework M-MCNN through FPGA, and transplant it on the drone to estimate and predict the high-density crowd area in real time. Our model achieved good results in crowd counting.

Titre : Implementation of real time reconfigurable embedded architecture for people counting in a crowd area

Mots-clés : Caractéristiques de texture, Détection de contours, M-MCNN, FPGA

Résumé :

Le comptage des foules est un sujet de recherche important. De nos jours, la population est de plus en plus préoccupée par les problèmes de sécurité. Lorsque la densité de population atteint des pics élevés, les systèmes de comptage se mettent en route et analyse les foules, afin de réorienter le surplus de personnes lorsque le seuil normal est dépassé. Avec ce genre de système, le piétinement du nouvel an de Shanghai ne se reproduirait plus. Actuellement, le comptage de population rencontre deux problèmes majeurs : l'analyse des foules dans les zones à forte densité de population, ou comment faire pour que le modèle distingue le plus finement possible les caractéristiques d'une tête humaine d'une part; et comment trouver une caractéristique de tête dans une image avec une large gamme de densité de population d'autre part. L'aspect le plus critique pour cette analyse est l'impossibilité d'installer un système de vidéosurveillance intelligent dans certains lieux publics. Dans ces conditions, comment pourrions-nous estimer la densité de population dans ces zones afin d'éviter de futurs accidents ? Face à ces défis, nous proposons la mise en œuvre d'une architecture embarquée reconfigurable en temps réel pour le comptage des personnes dans les zones de regroupement. Premièrement, notre travail intègre les fonctionnalités de HOG et LBP, qui non seulement combinent les informations d'identifications de multiples caractéristiques, mais également la plupart des informations redondantes, réalisant ainsi une compression efficace des informations, économisant ainsi de l'espace mémoire pour le stockage des données. Pour le comptage de personnes dans une foule, nous utilisons plusieurs sources d'informations, à savoir HOG, LBP et le filtrage de CANNY. Ces sources fournissent des estimations distinctes du nombre de personnes comptées et d'autres mesures statistiques de classification, par le biais du vecteur de support machine SVM. Dans le même temps, afin de résoudre efficacement le problème d'extraction des fonctionnalités liées à l'échelle dans le comptage de foules, nous proposons un nouveau environnement M-MCNN basé sur MCNN utilisé pour le comptage de foules sur une seule image. M-MCNN contient non seulement les trois colonnes originales des réseaux de neurones convolutionnels avec différentes tailles de filtres, mais aussi remplace les couches entièrement connectées par une couche convolutionnelle de filtre 1*1, de sorte que l'image d'entrée du modèle peut être de n'importe quelle taille. De plus, pour un échantillon individuel, nous améliorons considérablement l'apprentissage des caractéristiques de l'échantillon en extrayant les caractéristiques de texture d'une seule tête humaine et mieux l'utiliser pour les jeux de données. Enfin, nous implémentons notre nouveau framework M-MCNN sur un FPGA et l'installons sur un drone pour estimer et prévoir la zone de foule à haute densité en temps réel. Notre modèle a obtenu de bons résultats en comptage de personnes dans une foule.

Université Bourgogne Franche-Comté
32, avenue de l'Observatoire
25000 Besançon, France

ACKNOWLEDGEMENTS

I devote my success in Computer field (Research results) to my professor El-Bay Bourennane, who enlightened me, inspired me and encouraged me in the past 4 years.

I am delighted to have the opportunity to study at the prestigious University of Burgundy. Particularly, I want to express my sincere gratitude to The University of Burgundy ImVia lab for funding my research. Thanks to ImVia Lab for providing me a good working environment and platform. At the same time, I also thank China Lian Chuang Company for its financial support. Without the ImVia lab, I wouldn't have the opportunity to accomplish my research dream and to make acquaintance with so many talented people around the world. In the future, I am looking forward to have the honor to bring more glory to the ImVia lab.

Furthermore, I would love to give the credit to my team, my colleagues, and my family for supporting me all the time.

GONG SONGCHENCHEN

September 2020

CONTENTS

I

INTRODUCTION

1

BACKGROUND

China is a populous country. According to the population announced by the Bureau of Statistics in 2019, the population of China is 1.397 billion. This is a huge number. Hence, the problem that people care most about now is safety. Then, how can a security precaution be made in a populous country? In the 21st century, people's lives have become more and more colorful: large-scale concerts, music and dance festivals, World Cup, New Year's fireworks and more. These entertainment activities bring us joy and make life more exciting. On the contrary, do we know exactly the potential safety hazards that these entertaining activities bring to people? How do we solve it? For example: December 31, 2014, the New Year's Eve in Shanghai, was a day that impresses everyone. As it was the New Year's Eve, many tourists and citizens gathered on the Bund to celebrate the upcoming new year. During the celebration, one person fell to the bottom of the walkway. Following this first incident, many others tripped and fell onto each other one after another, eventually leading to large-scale crowding and stamping. If an alert could be issued quickly, the crowd could be dispersed earlier which would prevent the Shanghai New Year stampede from happening. Therefore, the research on crowd counting is becoming increasingly hot. If you can accurately estimate the crowd density of the current scene, issue an alert, and arrange corresponding security measures, you can effectively reduce or avoid such incidents.

Video surveillance is an important part of security protection, and the number of people and crowd density are an important factor of concern for video surveillance. In order to clearly introduce the development history of people counting and crowd density estimation technology, we elaborate from the development of monitoring equipment.

Electronic surveillance systems began to appear in the 1970s, and the development of video surveillance technology can be divided into three stages. The first generation is the analog video surveillance system. In the 1970s, closed-circuit television monitoring systems that depended on coaxial cable transmission began to appear. The image quality transmitted by this generation of technology is poor, making it difficult to adapt to large-scale surveillance. The second generation is the digital video surveillance system. In the mid-1990s, thanks to advances in digital encoding technology and chip technology, a digital video surveillance system was born. The image quality of this generation of technology is good, and is suitable for city-level. The third-generation is the intelligent video surveillance system. The second-generation is the digital video surveillance system gave birth to large-scale video surveillance. The demand for video surveillance systems is also increasing. There are questions that people are interested in, such as: Where are you? what are you doing? The third-generation intelligent video surveillance system analyzes the original video image based on a series of computer vision algorithms. Thus,

it becomes possible to automatically answer these questions raised by people.

With the rising of people's safety requirements and the improvement of economic conditions, there are more and more surveillance cameras with wider coverage. Traditional video surveillance systems require surveillance personnel to be on duty all the time. However, when the monitoring staff looks at the screen for a long time, it could cause visual fatigue and it is difficult to respond to some emergencies in a timely manner. In order to prevent crowd trampling accidents, researchers turned to computer vision-based population statistics and crowd density estimation. Automatic and reliable estimation of the number of people or crowd density not only provides means of alert in cases of abnormality, but also facilitates research in crowd simulation and behavioral analysis.

The birth of artificial intelligence video surveillance system, not only provides a large amount of monitoring data, but also provides conditions for the development of deep learning people counting and crowd density estimation technology. The development of deep learning algorithms has also prompted video surveillance systems to become more intelligent. Intelligent crowd density estimation and motion estimation can be used for crowd monitoring and management, and can also be applied to commercial fields, such as market research, traffic safety, and architectural design. They can directly or indirectly improve the work efficiency and utilization of building facilities in the above-mentioned occasions. Therefore, the study of population density estimation method has far-reaching significance and broad prospects.

2

PROBLEM DEFINITION

Early population research focused on detection-based methods by using a sliding window detector to detect the crowd in the scene and count the corresponding number of people. Detection-based methods fall into two broad categories. One is based on the whole detection, and the other is based on the detection of part of the body. Based on the overall detection method: a classifier is trained to detect pedestrians using features such as wavelets, HOG, and edges extracted from the pedestrian's entire body. Algorithms include SVM, boosting, and random forest methods. The method based on holistic detection is mainly suitable for sparse crowd counting. With the increase of crowd density, the occlusion between people becomes more and more serious. Therefore, a method based on partial body detection is used to deal with the problem of counting people. It mainly counts the number of people by detecting parts of the body, such as the head and shoulders. This method has a slight improvement over the overall detection.

The regression-based method mainly learns the mapping between extracted features and the number of people. These methods are mainly divided into two steps: the first step is to extract low-level features, such as foreground features, edge features, texture and gradient features; the second step is to learn a regression model. For example, linear regression, piecewise linear regression, ridge regression, and Gaussian process regression are used to learn the mapping relationship between a low-level feature and the number of people.

In recent years, Deep learning (DL) based approach has been widely used in various research fields: Computer vision, Natural language processing, etc. With its excellent feature learning capabilities, deep learning is also used by researchers in the study of population counting. Many corresponding data sets are established, such as ShanghaiTect A and B, UCSD, Expo2010, Mall, UCF-CC-50 and UCF-QNRF. Different from the traditional detection and regression-based methods, the prediction density map (DM) method is used to obtain better prediction results for dense crowd areas in the image. MCNN (CVPR 2016), due to the extremely uneven population density distribution in the image, researchers have used Multi-column arrays of the Convolutional Neural Network (CNN) to extract head features at different scales. The features of crowd images were extracted by using three networks with different convolution kernel sizes, and finally the features at 3 scales were fused by 1×1 convolution. This type of model using multiple networks has many parameters and a large amount of calculation, so it cannot perform real-time crowd counting prediction. Switch-CNN (CVPR 2017) uses the idea of three sub-networks and classification to let patches of different density levels pass through the corresponding sub-networks. CSRNet (CVPR2018) uses a pre-trained VGG16 network followed by Dilated Convolution to obtain state-of-the-art results, making it easier to get

head edge information.

The most difficult problem of population counting at present: On the one hand, in the densely populated area, how to make the model distinguish human head features more finely, such as head overlap.

For example: in the current tourist attractions, the staff will install surveillance cameras on the necessary road sections to monitor the flow of crowds in real time to better diffuse crowds and avoid trampling.

Here, we show a image of a crowd at a tourist attraction. We can observe from the image that there is heavy occlusion and head overlaps among the crowds.

Figure 2.1 – Severe occlusion of crowds in tourist attractions.

In this case, the places marked by the red arrow in the 2.1 crowd image pose challenges for crowd density estimation due to severe occlusion between people. The main reason is that occlusion is prone to cause wrong population density estimates. For example: When two people in the image overlap each other, it is possible that we only count that there is only one person. This is just an occlusion between 2 people. If there are many such occlusion problems in the crowd image, it is a great challenge for us to estimate the crowd density using an intelligent video surveillance system.

The second aspect is to find a small-scale local head feature in an image with a wide range of population density.

Figure 2.2 – Crowd image of GOLDEN LANE in Prague, Czech Republic. The position marked with blue dashed lines and squares in the image shows the part of the head that is severely occlusion between people.

The image above is an image of the crowd from GOLDEN LANE in Prague, Czech Republic. We can clearly see that there are many tourists and local residents in the image.

In this case, the places marked by the blue dashed lines and squares in the figure 2.2 crowd image pose challenges for crowd density estimation due to severe occlusion between people. So, in such a crowded image, how can we find local heads feature on a smaller scale? We believe that an intelligent video surveillance system has also been installed on this GOLDEN LANE to ensure people's safety. Suppose in the image, a mother walks with her child in the crowd, how about we quickly find this child through the intelligent video surveillance system? How can we quickly find this child by her feature? Again, this is a daunting challenge.

The third aspect is that in large sports halls, large concert venues, we can not install an intelligent video surveillance system. So how do we estimate the high-density crowd area to avoid crowd trampling accidents? The image below is an image of a concert crowd.

Figure 2.3 – An image of a concert crowd.

In the fourth aspect, how do we estimate and predict the high-density population area in real time? Faced with these challenges, how do we solve these problems. In the following chapters, we propose solutions.

3

PROPOSED APPROACHES

As discussed in the problem definitions, the most difficult problem with crowd counting is in densely populated areas. How to make the model distinguish human features more finely, thus accomplishing the task of counting people. The second problem is that in an image with a wide population density, how to find a small-scale local feature head? The third problem is how to achieve crowd density estimation and prediction where we cannot install an intelligent video surveillance system. The fourth problem is how to estimate and predict the high-density population area in real time. Therefore, the following methods are proposed to solve these problems.

We propose a multi-feature fusion population count based on a multi-column convolutional neural network. It is mainly divided into 3 parts.

The first part is: Multi-feature fusion technology. The main purpose is to find head features. We extract image features from multiple information sources, and find head features through texture feature analysis and crowd image edge detection. First of all, we concatenate the HOG feature vector and the LBP feature vector in series to form a joint feature vector followed by SVM. Here, in the classification process, the linearly inseparable low-dimensional space is converted into a linearly separable high-dimensional space mainly through SVM kernel functions , and the cross-validation method is used to select the SVM optimal parameters, so that the classifier has the highest classification accuracy of the input training samples. It is shown by the study of the human visual system, image boundary is particularly important; often a rough outline of the line alone can be used to identify an object. We were inspired by machine vision research, then, we optimize the original canny operator to suppress false edges caused by noise, make target edges thinner, and better obtain target edge features. Multi-feature fusion technology obtains clear head contours by extracting target edge detection features and analyzing texture features.

The second part is: A new framework M-MCNN based on multi-column convolutional neural network is used for crowd counting on any single image. M-MCNN not only contains the original three columns of convolutional neural networks with different filter sizes, but replaces the fully connected layers with a convolutional layer of 1*1 filters, so the input image of the model can be of any size. This avoids the need for image resizing which may induce distortion. In a single individual sample, we use the first part to extract the texture features of a single human head and detect the head edge features, greatly improving the learning ability of sample features. At the same time, the loss of density map details is also reduced, and it is better integrated with our convolutional neural network. We conducted a large number of experiments on the ShanghaiTech, USCD, WorldExpo'10 and

GCC datasets. Not only that, we have created a new dataset. This data set is called the CHDP (high-density population) data set. It contains image of high-density people that we collected from Google, Baidu, and other major websites. The size of each high-density crowd image varies. Its purpose is to better detect the M-MCNN crowd counting algorithm we have proposed. Our model outperforms the state-of-art crowd counting methods on all data sets used for evaluation.

The third part, we implement the neural network architecture through FPGA hardware. It is mainly designed and implements the key parts of the hardware implementation of neural networks. The process of implementing the hidden layer is described in detail, and the calculation process of each stage of the network is analyzed and simulated. From the analysis of the experimental results, we can build a system that can well implement the function of crowd counting and having higher accuracy and stability, so as to achieve the experimental purpose. We carried it on a drone to estimate and predict the high-density crowded area in real time.

4

PUBLICATIONS

INTERNATIONAL JOURNALS

R-1. **Gong Songchenchen**, El-Bay Bourennane, "Implementation of real time reconfigurable embedded architecture for people counting in a crowd area", *(submitted to Springer* (**Modelling and Implementation of Complex Systems**)).

INTERNATIONAL CONFERENCES AND WORKSHOPS

CI-1. **Gong Songchenchen**, El-Bay Bourennane, "Implementation of real time reconfigurable embedded architecture for people counting in a crowd area", *Symposium on Modelling and Implementation of Complex Systems (MISC 2018), Dec 2018, Laghouat, Algeria.* DOI: 10.1007/978-3-030-05481-617

CI-2. **Gong Songchenchen**, El-Bay Bourennane,"A method based on texture feature and edge detection for people counting in a crowded area", *Digital Image and Signal Processing (DISP'19), 29-30 April, 2019, St Hugh's College, University of Oxford, United Kingdom.* DOI: hal-02275646

CI-3. **Gong Songchenchen**, El-Bay Bourennane, "A method based on multi-source feature detection for counting people in crowded areas", *IEEE 4th International Conference on Signal and Image Processing (ICSIP 2019), 19-21 July, 2019, Southeast University in Wuxi, China.* DOI: 10.1109/SIPROCESS.2019.8868691.

CI-4. **Gong Songchenchen**, El-Bay Bourennane, Junyu Gao, "Multi-feature Counting of Dense Crowd Image Based on Multi-column Convolutional Neural Network", *IEEE In2020 5th International Conference on Computer and Communication Systems (ICCCS) 2020 May 15, Shanghai, China*.DOI: 10.1109/ICCCS49078.2020.9118564..

CI-5. **Gong Songchenchen**, El-Bay Bourennane, Xuecan Yang, "MFNet: Multi-feature convolutional neural network for high-density crowd counting", *IEEE In2020 11th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON) 2020 November 4-7, Vancouver, Canada.*

NATIONAL CONFERENCES AND WORKSHOPS

CN-1. **Gong Songchenchen**, El-Bay Bourennane,"Implémentation en temps réel d'une architecture embarquée pour le comptage de personnes dans une foule", *13ème Colloque du GDR Sol/Sip juin 2018, Paris, France.* DOI: hal-01866947

CN-2. **Gong Songchenchen**, El-Bay Bourennane, Hiliwi Leake Kidane, "Implémentation en temps réel d'une architecture embarquée pour le comptage de passagers dans le transport public", *Colloque national FUTURMOB-18, Nevers, France, June 18-19, 2018* DOI: hal-01866931

5

DISSERTATION OUTLINE

This manuscript is divided into nine parts:

The first part includes chapters 1 to 5:

In Chapter 1, we present current computational aspects of artificial intelligence. Research in this area includes robotics, speech recognition, image recognition, and natural language processing.

In Chapter 2, we define the problem to address.

In Chapter 3, we propose solutions to research topics.

In Chapter 4, we summarize our contributions.

In Chapter 5, dissertation outline.

The second part includes chapters 6 to 8:

In Chapter 6, we introduce the first contribution: Multi-feature fusion technology for target edge detection and analysis.

In Chapter 7, we introduce the second contribution: Multi-feature counting of dense crowd image based on convolutional neural network.

In Chapter 8, we introduce the third contribution: Implementation of real time reconfigurable embedded architecture for people counting in a crowd area.

The third part includes Chapter 9:

In Chapter 9, Conclusion and Perspectives.

II

CONTRIBUTION A

6

MULTI-FEATURE FUSION TECHNOLOGY FOR TARGET EDGE DETECTION AND ANALYSIS

6.1/ BACKGROUND

In the research of target detection technology, pedestrian detection has a huge market prospect and a wide range of applications, such as banks, shopping malls, airports, railway stations, and parking lots of security-sensitive occasions. Most of the researches are aimed at improving the accuracy and speed of pedestrian detection.

6.2/ PEDESTRIAN DETECTION AND CROWD COUNTING

6.2.1/ PEDESTRIAN DETECTION

Pedestrian detection is the use of computer vision technology to determine the presence of pedestrians in an image or video sequence and give precise positioning. This technology can be combined with pedestrian tracking, pedestrian re-identification, etc. It can be applied to artificial intelligence systems, intelligent robots, intelligent video surveillance, human behavior analysis, intelligent transportation and other fields. Becasue pedestrians are easily affected by clothing, scale, occlusion, posture, and perspective, pedistrian detection has become a subject of research value in the field of computer vision.

- ***Wavelet Concept Features***:

  Papageorgiou and Poggio first proposed the concept of Haar wavelet. The detection technique is based on a wavelet representation of an object class derived from a statistical analysis of the class instances [53]. In 2001, Viola and Jones introduced the concept of integral graphs. Based on the AdaBoost algorithm, face detection is performed using Haar wavelet features and integral graph methods. They designed more effective features for face detection and cascaded the strong classifiers trained by AdaBoost [72]. When it comes to face recognition and detection, it is a hot research area in computer technology. These include face tracking detection, night infrared detection, and automatic adjustment of exposure intensity. For example, the

DPM model proposed by Markus Mathias et al.[49], combined with a rigid template detector, is well used in commercial and research systems. A method based on K-Mean clustering, Bresenham algorithm, Graham Scan algorithm. With the help of image morphology, the algorithm will detect the skin texture of the face [47]. A multi-task deep learning method called HyperFace, which can simultaneously detect faces, locate landmarks, estimate head pose, and recognize gender [56].

- ***Texture feature*** : Then, in 2005, Dalal and Triggs proposed the concept of gradient direction histogram (HOG). HOG is an edge feature, which uses the orientation and intensity information of the edges by calculating the gradient orientation and intensity at each point. It then forms a gradient direction distribution histogram of all pixels in the grid. Finally, they are summarized to form the entire histogram feature [12]. HOG features are well used for pedestrian detection[46]. Almost 100% detection success rate was obtained on the pedestrian database provided by MIT, it contains multiple changes of perspective, lighting and background. The INRIA pedestrian database also achieved a detection success rate of approximately 90%.

- ***Global and local features and classifiers*** : Then, in 2007, Sabzmeydani and Mori proposed a feature that can be obtained automatically using machine learning methods [60], named Shapelet feature. The algorithm first extracts gradient information in different directions of the picture of the training samples. These functions focus on local areas of the image, and based on low-level gradient information that can distinguish between pedestrian and non-pedestrian categories. By using the AdaBoost algorithm for training, these wavelet features can be created as a combination of directional gradient responses. In the same year, a framework for detecting and segmenting people with a large number of poses in crowded videos was proposed [59]. The framework integrates local and global shape cues [65] by learning a set of average pose clusters and a codebook of human body shape distribution in various poses. It helps human segmentation and detection. In 2010, a combination of ISM model and joint occlusion analysis was used for individual segmentation [28]. The advantage is that there is enough information in the current scene area, and even individuals in dense areas can be processed. The other is that when the contour of the current scene is not very accurate, the rough foreground area can also be obtained as well. One year later, in 2011, Mikel Rodriguez et al. proposed the use of information about the global structure of the scene and jointly solved all detected problems [58]. In particular, they study the constraints imposed by crowd density and formulate the detection of people into an optimization of a joint energy function that combines crowd density estimation and personal positioning. This optimization of the energy function significantly improves the detection and tracking of people in the crowd.

- ***FPGA based solutions*** : Fradi et al. [15] proposed a FPGA-based design for crowd counting using low-level features. Use different circular patterns to process images in parallel and detect human heads of different sizes. When detecting around the head, the different features extracted are given to the tracking algorithm. They target image preprocessing and edge extraction on reconfigurable hardware. Tomasz Kryjak et al. proposed a system for head and shoulder positioning in a video stream. It is implemented on FPGA for head and shoulder detection and pedestrian counting. For feature extraction, a local binary (LBP) texture descriptor is used. The main

advantages of LBP descriptors are low computational complexity and invariant to local illumination changes, and support vector machine (SVM) classification [39]. To reduce the probability of false positions, foreground object detection is used as an additional verification criterion.

- ***Bayesian 3D Model*** : In 2010, Lu Wang et al. proposed a Bayesian 3D model-based method to solve human detection in crowded scenes. Human candidates are first specified by the head (modeled by ellipsoid), shoulder (modeled by the upper half of the ellipsoid), and torso (modeled by cylinder) detectors. After optimization, the best configuration of the candidate and the corresponding shape model are found [75]. In 2011, Matthias Butenuth et al. Proposed a general framework for crowd analysis to detect and track pedestrians. By using the appearance-based object detection method, fine objects are well detected. At the same time, iterative Bayesian tracking method is used to focus on specific locations where danger may exist [6]. In 2014, Hui Liu et al. Proposed a head detection method based on head features and applied it to video pedestrian tracking based on the human head's similar round shape, and the unique color of hair. Here, the head vertex is used as the feature point, and the head contour is sampled to complete the ellipse fitting. Finally, the method of circle-like detection is used to realize the human head detection. Experimental results show that this method has a good effect when applied to scenes with sparse crowds [32].

In recent years, through the method of motion features, the target's motion information is added to the pedestrian detection system and combined with other static features to detect pedestrians. For example, as proposed by Viola et al., Haar-like features can be calculated between images when the camera is stationary. Then the motion information is combined with the gray-scale intensity of the image to construct a pedestrian detection system. In detection method based on human body parts, the human body is divided into several component parts, and then each part of the image is detected separately. Finally, the detection results are integrated according to a certain constraint relationship to finally determine whether there are pedestrians. Different detection methods are proposed for better pedestrian detection.

6.2.2/ CROWD COUNTING

In the past ten years, with the continuous development of computer vision technology, a large number of crowd counting methods have been proposed.

- ***HOG texture features*** : This corresponds to the head-shoulder shape based on the HOG only inside the foreground region. Based on the joint HOG feature [8], the Haar [53] [72] classifier is used for coarse filtering and the SVM cascade classifier is used for accurate verification. Using the MID model, the HOG is based on the head and shoulder detection, and the foreground segmentation estimates the number of crowded scenes [41].

- ***LBP texture feature*** : A new method specific to LBP is used to calculate the co-occurrence matrix on the LBP feature map. The cell-based method was used to construct the LBPCM feature vector [79]. The Adaboost classifier with LBP features

is used to optimize the foreground area of and perform head detection. Keju et al. [35] track the target object by the mean shift algorithm, and use the crossing-line method to determine whether the detected head is counted.

- ***Texture feature*** : Through a variety of texture classification, crowd patches are divided into multiple different categories. Each class represents a predefined range of population densities [51]. Vora create a set of multi-resolution density cells based on a perspective projection model. The GLDM function provides vectors for each cell. The function vector is fed back to the SVM training to solve the nonlinear regression problem [80]. Depending on the method using texture features, the extraction uses a Gabor filter and a least squares support vector machine [40].

- ***Adaboost's algorithm and spectral texture features*** : Population density estimation based on multi-class Adaboost algorithm and spectral texture features [36].

- ***Foreground and texture features*** : Through the foreground feature extraction and detection of the crowd, the combined LK optical flow and GBM method is used to eliminate background noise. Yang et al. [83] added new function of texture analysis settings for crowd density estimation.

- ***Gray-level texture analysis*** : Humans perceive images through their attributes, such as color, shape, size, and texture. Crowd surveillance is performed automatically based on the texture information on the grayscale transition probability in the digitized image. The crowd density estimation is based on the amount of texture of the image and gives very low, low, medium, and high density. The number and range of each estimation itself depends on the specific application and specific characteristics of the surveillance area. The crowd density feature vector is extracted from the digitized image for crowd estimation [48].

- ***Multiple sources*** : The count is estimated by fusing information from multiple sources, namely: points of interest (SIFT), Fourier analysis, wavelet decomposition. In[4] GLCM function and low-confidence head detection are used to estimate counts. The target detection and tracking algorithm are used to estimate the density population [23]. Based on local feature extraction, tracking, and kernel density estimation, population density estimation are performed [15].

- ***Background subtraction and SVM*** : A method based on training classifier and background subtraction. It further describes estimating the number of people in a population relative to the results of head detection [68]. The background image is automatically generated from the crowd video sequence and used as a reference image for crowd density measurement [84]. A discrete cosine transform is used to convert the motion state of the measurement area of the frequency domain to identify the frequency distribution. Feature values are extracted based on different frequency bands and different direction information for feature vectors used for training and classification. Support vector machine is used to classify feature vectors into multiple categories of population density [29].

- ***POI Features*** : In 2010, D. Conte et al. Proposed the establishment of a mapping between scene features and number of people. By detecting the points of interest

related to people, as well as the clustering of points of interest, the use of SURF feature extraction and the use of SVR regressors estimate of crowd counts [11]. Two years later, Venkatesh bala subburaman et al. Proposed a new type of point of interest detector based on gradient direction features. The top of the head area is detected by this detector. They detect the head of the image, and find the points of interest using the gradient information of the gray picture located approximately at the top. Combining two different background subtraction methods further narrows the search area, thereby simplifying the setting of background parameters. It helps to better estimate the population from a single frame [69]. In 2013, Haroon Idrees et al. used Fourier analysis with head detection and interest-based crowd counting [33].

- ***Crowd density estimation framework, Random Projection Forest (RPF)*** : An effective crowd density estimation framework is called Random Projection Forest (RPF). This framework combines a random forest regression model with a random projection method [81]. By using a random forest as a regression model, its tree structure is essentially fast and scalable. It embeds random projections in the nodes of the tree to overcome the interference of increasing dimensions at the same time. It also introduces randomness into the tree structure to more accurately and effectively describe the features and provide more reliable and accurate predictions of people statistics.

With the continuous improvement of people's living standards, there are more and more various entertainment activities, so people also like to gather together to participate in activities. Prediction of crowd density through video surveillance equipment.

- In 2003, Danny B. Yang et al. Proposed as an alternative method of directly estimating the number of people [82]. In this system, groups of image sensor segment foreground objects from the background. The contours generated by the network aggregation and the plane projection of the scene's visual shell are calculated. A geometric algorithm is also introduced here to calculate the number of people in each area of the projection after eliminating the phantom area. After that, crowd counting was performed.

- In 2004, Ruihua MA et al. Proposed a population density estimation for video surveillance based on pixel counting [45].

- A year later, X Liu et al. Proposed a video surveillance system, including visual tracking, automatic calibration, crowd segmentation and counting, and event recognition module [44]. This system is able to segment the population, while also counting the number of people entering or leaving a particular site. This model-based method can also make effective use of the spatial background to enable the system to automatically detect certain events.

- In 2006, Xinyu Wu et al. Proposed a method for calculating population density using density maps. In the initial stage, a set of multi-resolution density cells is created by seeing through the model. After that, the GLDM function extracts vectors for each cell. Finally, the function vector is fed back to the SVM training to solve the

nonlinear regression problem. To a certain extent, this method makes use of cells and search algorithms with multi-resolution scale changes, while also relying more on the background [80].

- In 2009, for the problem of passenger flow detection in dense traffic scenarios, Keju Zhu et al. Proposed a method based on support vector machine (SVM) multi-target detection combined with Mean Shift tracking. First, an adaptive detection window is used to extract the gradient direction histogram. After SVM classification and clustering algorithm, the initial hypothesis of the head image are obtained. Then the Mean Shift algorithm is used to track the head hypothesis to obtain a continuous head image sequence. Finally, the overall image is judged by the SVM classifier to obtain the passenger traffic information [35].

- In 2010, Donatello Conte proposed a new method of counting people for video surveillance applications, which is divided into a direct and an indirect method. In the direct method, the person is first detected and then counted. An indirect method is used to establish a link between some easily detectable scene features and the estimated number of people [10].

- In 2012, Zhong Zhang and others proposed an extended standard IVS system framework. They use available video analysis data and camera calibration information to provide accurate estimates of people count in crowded scenes [91]. In the same year, Hajer Fradi et al. Proposed the advantage of merging a uniform motion model into a Gaussian mixture model (GMM) background subtraction, thereby obtaining high-precision foreground segmentation. Crowd counting is based on foreground measurement, where perspective normalization is performed and the angular density associated with crowd measurement is introduced into a single function along with the number of foreground pixels. Then, the correspondence between the characteristics of this frame and the number of people is learned through Gaussian process regression [14].

- In 2013, Zhang Ma et al. Proposed a population counting method for integer programming [46]. It is used to estimate the instantaneous number of pedestrians crossing the line of interest in a video sequence. Through the line sampling process, the video is first converted into a time slice picture. Next, they estimate the sliding window where the number of people in a group overlaps with the time-sliced image. Count by using a regression function mapped from local features. Integrate over a specific time interval to get the number of pedestrians crossing the line.

- In 2014, Ayse Elvan et al. Proposed a population density classification method and applied it to video population density estimation [23].

Population quantity and density are important attributes of population analysis. The methods mentioned above still have challenging problems in measuring high population density. At present, the most difficult problem is how to make the model distinguish human features more precisely in densely populated areas, for example: the problem of overlapping heads. The other is how to find small-scale local feature heads in an image with widely distributed population density. Next, we propose a new method to solve these two problems.

6.3/ FUSION OF TEXTURE FEATURE AND EDGE DETECTION FOR PEOPLE COUNTING

To address the aforementioned challenges, we propose a population counting method based on for feature fusion and edge detection. Each image undergoes multiple pipelines including image feature extraction, texture feature analysis, and crowd head edge detection to estimate the count. We target people counting at potentially populous locations, such as city bus and subway stations. Our method uses a still image taken by the camera to estimate the count in the crowd density image, using multiple sources of information, namely: HOG, LBP, and CANNY edge detector. These sources provides separate estimates of the number of counts and other statistical measures, through the support vector Machine SVM, classification, and regression analysis to obtain high-density populations. Our method is shown in the Figure 6.1:

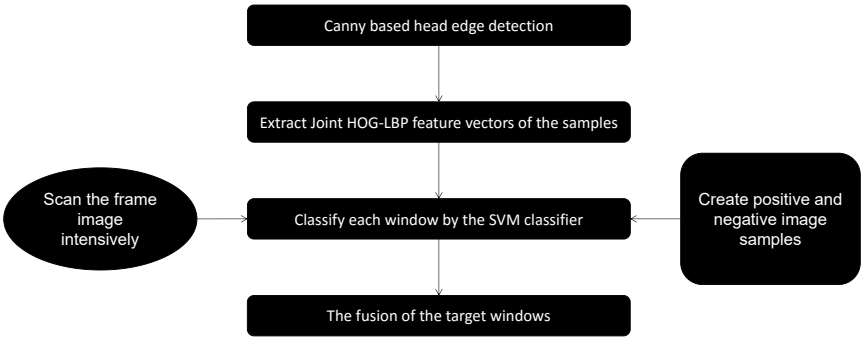

Figure 6.1 – Head feature extraction.

6.3.1/ EDGE DETECTION

In the actual image edge detection problem, as a basic feature of the image, the edge of the image is often used in higher-level image applications. It has a wide range of applications in the fields of image recognition, image segmentation, image enhancement, and image compression.

Because the image edge is one of the most basic features of the image, which often carries most of the information of the image, we need a very important feature condition, which requires us to detect and extract its edge. Edge detection has very important use value in many aspects, so people have been working to study and solve the problem of how to construct the edge detection operator which has good properties and good effect. We can think of the singular points and the image edge points in the image. The gray level changes can be reflected by the gradient of adjacent pixels. According to this feature, we propose a variety of edge detection operators such as Sobel operator, Roberts operator, Prewitt operator, Log operator, Laplace operator, Canny operator and so on. To achieve the extraction of image edge and good effect has been achieved.

The edges of objects appear in the form of local discontinuities in the image, for example, the abrupt changes in color and texture structure. Physically, edges often mean the end of one area and the beginning of another. Image edge information is very important

in image distribution and human vision. It is an important attribute for extracting image features in image recognition.

6.3.1.1/ EDGE DETECTION AND EXTRACTION PROCESS

Basic steps of image edge detection:

(1) Filter: Edge detection is mainly based on derivative calculation, but it is affected by the noise. Reducing noises often leads to the loss of edge strength.

(2) Enhancement: The enhancement algorithm will be in the neighborhood of the gray level has a significant change of the points highlighted. Generally through the calculation of the gradient amplitude.

(3) Check: But in some of the image gradient detection: in large amplitude and edge. Edge detection is not simple is the gradient threshold determination.

(4) Positioning: Accurately determine the location of the edge.

Feature extraction is an important part of image edge detection, and many effective methods have been made available for it. These methods have been tested in real-life applications and some of them even have been standardized. The classic edge detection operators include: Sobel operator, Roberts operator, Prewitt operator, Log operator, Laplace operator, canny operator, etc. These classic edge extraction operators use a pre-degined edge model to locate the occurence of edges in an image.

6.3.1.2/ SOME EDGE DETECTION AND EXTRACTION OPERATORS

Sobel operator:

The convolution of the Sobel operator [22] is shown in Figure 6.2. Each pixel in the image is used to do the convolution of the two kernel. The two kernel have the largest response to the vertical and horizontal edges respectively. The maximum of the two convolutions is used as the output bit of the point.

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Figure 6.2 – Sobel edge operator.

Roberts operator:

The principle of gradient amplitude can be calculated according to the difference of any pair of mutually perpendicular directions. Provides a simple approximation method for gradient amplitude calculation, that is, the difference between two adjacent pixels in the diagonal direction:

$$\Delta_x f = f(i, j) - f(i+1, j+1) \tag{6.1}$$

$$\Delta_y f = f(i, j+1) - f(i+1, j) \tag{6.2}$$

Their convolution operators are:

$$\Delta_x f = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \Delta_y f = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{6.3}$$

Difference scores will be interpolated point [i+1/2, j+1/2] office calculation, even if the gradient of the Roberts, f(i, j), but it is worth noting that the Roberts operator is the point of the gradient approximation, not the expected point (i, j) of the approximate value.

Roberts operator edge positioning is accurate, but it is sensitive to noise. It is suitable for image segmentation with obvious edge and less noise, and it often uses Roberts operator to extract the feature.

Prewitt operator:

Convolution of Prewitt operator as shown in Figure 6.3, each pixel in the image is used to perform the convolution of the two kernels, the maximum value as the output, but also produces an edge amplitude image.

-1	-1	-1
0	0	0
1	1	1

1	0	-1
1	0	-1
1	0	-1

Figure 6.3 – Prewitt edge operator.

The expression of the Prewitt operator is as follows:

$$S(x, y) = \begin{bmatrix} \left|[f(x+1, y-1) + f(x+1, y) + f(x+1, y+1)]\right| - \\ \left|[f(x-1, y-1) + f(x-1, y) + f(x-1, y+1)]\right| + \\ \left|[f(x-1, y+1) + f(x, y+1) + f(x+1, y+1)]\right| - \\ \left|[f(x-1, y-1) + f(x, y-1) + f(x+1, y-1)]\right| \end{bmatrix} \tag{6.4}$$

Prewitt operator in one direction handles differentiation, and in another direction handles smoothing, so the noise is relatively insensitive, as there is a noise suppression effect. But the average pixel equivalent to the image of low pass filtering.

Log operator:

Laplacian of Gaussian operator is a kind of zero crossing point using the two order derivative of the image intensity. The algorithm is very sensitive to noise, so the noise is filtered before the edge is enhanced.

LOG operator is used in order to reduce the noise influence and to deal with detection of image smoothing and smoothing function with normal distribution of Gaussian function g (x, y, σ), namely:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{x^2+y^2}{2\sigma^2}\right) \tag{6.5}$$

G (x, y, σ) of image f (x, y) of the filter can be obtained by the following convolution:

$$g(x,y) = G(x,y,\sigma) * f(x,y) \tag{6.6}$$

The Laplace operator for edge detection can be otained by the following formula:

$$\nabla^2 g(x,y) = \nabla^2 (G(x,y,\sigma) * f(x,y)) = \nabla^2 G(x,y,\sigma) * f(x,y) \tag{6.7}$$

$$\nabla^2 G(x,y,\sigma) = k\left(\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right) exp\left(-\frac{x^{2+y^2}}{2\sigma^2}\right) \tag{6.8}$$

Laplace operator:

It is a kind of two order derivatives edge detection operator of a continuous function f (x, y) its position in the image (x, y), the Laplace value is defined as:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{6.9}$$

0	1	0
1	-4	1
0	1	0

1	1	1
1	-8	1
1	1	1

Figure 6.4 – Laplace edge operator.

Commonly used Laplace is sensitive to noise, the Laplace operator has a drawback which is its dual response to some of the edges in the image. So the image after smoothing, usually the Laplace operator and smoothing operator combined to generate a new template.

Canny operator:

Canny edge detection algorithm [3][26][55][64]:

Step1: Gauss filtering to smooth images;

Step2: Calculate the gradient magnitude and direction by finite first-order partial derivatives of the filtred image;

Step3: non maximum suppression of amplitude of the gradient ;

Step4: Use of dual threshold algorithm for edge detection and link edges.

- ***Advantages of Canny operator detection method:*** :
  - ***Low Bit Error Rate (BER), rarely mistaken for non-edge edge points;***
  - ***The high positioning accuracy, which is precisely the edge location of the biggest change in gray pixels;***
  - ***To suppress the false edge.***

Select an ordinary person's head picture, add Gaussian noise, and then use Matlab programming language to compare and analyze the extraction results:

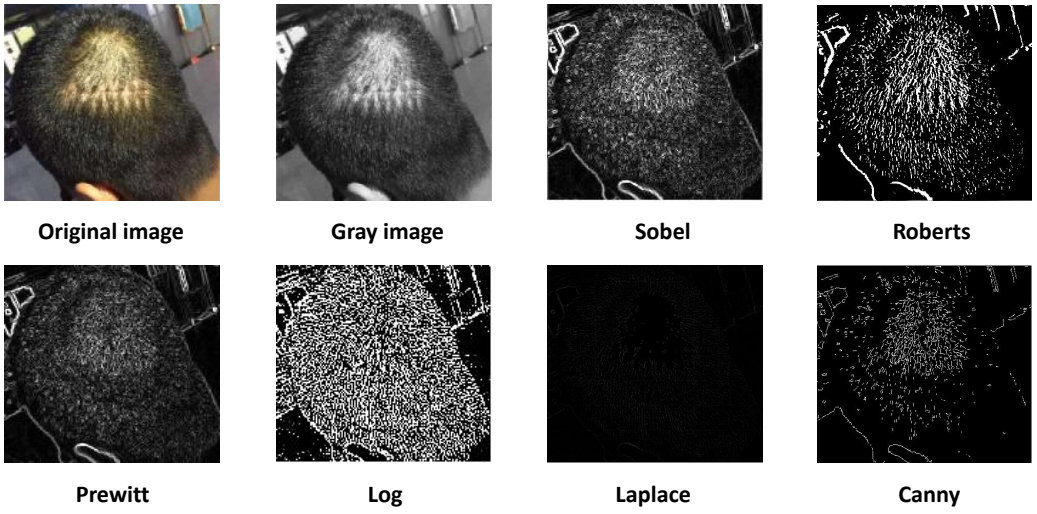

Figure 6.5 – Target image edge detection with different operators.

The results: The filter detected by using canny operator has the best quality. This is because canny has the ability to diagnose and optimize the edge detection. It first uses a two dimensional Gaussian function for noise filtering, then carries out the local maximum value of the image gradient to determine the image edge.

In summary:

- ***Sobel operator:*** The Sobel operator performs better on gray-scale gradients and images with more noise. Sobel operator is more accurate in positioning the edge. At the same time, Sobel operator is also a weighted average operator. According to its definition, the influence of neighboring pixels on the current pixel is not equivalent, thus, pixels with different distances have different weights. In general, the further a pixel is, the smaller its impact.
- ***Roberts operator:*** Roberts operator performs better on low-noise images. This also shows that Roberts operator is very sensitive to noise. Roberts operator is suitable for image segmentation with obvious edges and less noise. The resulting edges by the Robert operator are not very smooth. Through analysis, the Robert

operator usually produces a wider response in the area near the edge of the image. Roberts operator uses edge image thinning processing. As a result, its accuracy for edge positioning is not very high.

- ***Prewitt operator:*** The Prewitt operator is better for image processing with gray gradient and noise. This shows that the Prewitt operator has a good effect on suppressing noise. However, compared with Roberts operator, Prewitt operator is not very accurate in positioning the edge of the image.
- ***Log operator:*** The Log operator is more sensitive to noise, hence it is rarely used to detect image edges. Using the Log image edge detection algorithm with Gaussian function filtering can suppress the noise, however at the cost of smoothing out low-intensity image edges.
- ***Laplace operator:*** Laplace operator is very sensitive to noise. Laplace operator tends to partially lose the direction information of an edge during image detection, resulting in some discontinuous edge detection.
- ***Canny operator:*** The Canny operator is not easily affected by noise and can detect weak edges of the image well. The advantage of the Canny operator is that it uses two different thresholds to detect strong and weak edges, respectively. When the weak and strong edges are connected, the weak edges are included in the output image. Therefore, the Canny operator can detect true weak edges more easily, and is better used in image edge detection.

6.3.2/ IMPROVEMENT OF CANNY

By analyzing the principle of Canny's edge detection algorithm, we have conducted in-depth research on the Gaussian filtering process in Canny operator [26]. We extend the computation of gradients into 4 or more directions, effectively improving the accuracy of Canny edge detection.

6.3.2.1/ IMPROVED METHOD OF CANNY OPERATOR

(1) First of all, denoising is done by using, Gaussian filtering. Its purpose is to smooth the original image and remove or weaken the noise in the image.

Assuming two dimensional Gauss's function [26][55]:

$$G(x,y) = \frac{1}{2\pi\sigma^2} exp\left(-\frac{\left(x^2+y^2\right)}{2\sigma^2}\right) \tag{6.10}$$

Gradient vector:

$$\nabla_G = \begin{bmatrix} \partial_G/\partial_x \\ \partial_G/\partial_y \end{bmatrix} \tag{6.11}$$

To improve the speed of decomposition method, the 2 filters at convolution template G is decomposed into 2 one-dimensional filters:

$$\frac{\partial_G (x, y, \sigma)}{\partial_x} = kxe^{\frac{x^2}{2\sigma^2}} e^{\frac{y^2}{2\sigma^2}} = h_1 (x) h_2 (y) \tag{6.12}$$

$$\frac{\partial_G (x, y, \sigma)}{\partial_y} = kye^{\frac{y^2}{2\sigma^2}} e^{\frac{x^2}{2\sigma^2}} = h_1 (y) h_2 (x) \tag{6.13}$$

Where k is a constant, and σ is Gaussian filter parameter. It controls the degree of smoothing. Although the positioning accuracy is high for sigma, the signal to noise ratio is low. So it is need to select the Gaussian filter parameter according to the requirement.

(2) The traditional canny algorithm through the neighborhood for finite difference to calculate the amplitude of the gradient. We adopt a new 3x3 neighborhood in calculating the gradient amplitude. The procedure is as follows:

The partial derivatives of the 4 directions are calculated at first. And then expand to multiple directions.

Partial derivative along the X direction:

$$P_x (x, y) = G (x, y + 1) - G (x, y - 1) \tag{6.14}$$

Partial derivative along the Y direction:

$$P_y (x, y) = G (x + 1, y) - G (x - 1, y) \tag{6.15}$$

45 degree directional partial derivative:

$$P_{45} (x, y) = G (x - 1, y + 1) - G (x + 1, y - 1) \tag{6.16}$$

135 degree directional partial derivative:

$$P_{135} (x, y) = G (x + 1, y + 1) - G (x - 1, y - 1) \tag{6.17}$$

Difference in horizontal direction:

$$f_x (x, y) = P_x (x, y) + [P_{45} (x, y) + P_{135} (x, y)] /2 \tag{6.18}$$

Difference in vertical direction:

$$f_y (x, y) = P_y (x, y) + [P_{45} (x, y) - P_{135} (x, y)] /2 \tag{6.19}$$

The gradient magnitude is obtained:

$$M (x, y) = \sqrt{f_x (x, y)^2 + f_y (x, y)^2} \tag{6.20}$$

The gradient direction is:

$$\Theta (x, y) = arctan \left(\frac{f_x (x, y)}{f_y (x, y)} \right) \tag{6.21}$$

The method takes into account the intensity changes along the diagonal direction, which increases the pixels to compute the partial derivative direction, and improves the traditional canny operator in terms of accurate localisation of edges.

(3) Non maxima suppression of gradient amplitude.

For gradient "non-maximum suppression", it is not enough to determine the edge based on the gradient magnitude of the image. In order to determine the edge, the roof ridge band in the gradient magnitude image must be refined, so that a refined edge can be produced. “Non-maximum suppression” refines the gradient amplitude roof by suppressing the gradient amplitude of all non-roof peaks in the gradient direction.

Firstly, this algorithm reduces the range of gradient directions to the one of the four sectors, as shown in figure 6.6. The grades of the four sectors from 0 to 3. For the first pixel (x,y), the sector along with the gradient directions is given by ξ (x, y), namely:

$$\xi(x,y) = Sector(\theta(x,y)), \xi(x,y) = 0,1,2,3 \tag{6.22}$$

In the 3*3 neighborhood of the pixel located at (x, y), each neighborhood pixel must be in a sector. The center pixel (x, y) was compared between two neighborhood pixel gradient amplitude and its gradient direction of the sector, where the gradient line is at the center of the neighborhood by Sector ξ(x,y) value is given. If the gradient magnitude of the central pixel (x, y) is smaller than that of two neighborhood pixels of the sector, $\left|\nabla_f(x,y)\right|$ is set to be zero. Otherwise $\left|\nabla_f(x,y)\right|$ at constant. This process not only obtains the refinement of wide ridge, but also remains the maximum inhibition information in "non-gradient" amplitude.

$$N[i,j] = NMS\left(\left|\nabla_f(x,y)\right|, \xi(x,y)\right) \tag{6.23}$$

Represents the non-maximum suppression process. A non-zero value in N[i, i] corresponds to the contrast at which the image intensity changes stepwise. Although the image is smoothed in the first step of edge detection, the non-maximized suppressed amplitude image N[i, i] contains many false edge segments caused by noise and texture. In practice, the contrast of false edge segments is generally very small.

(4) Dual threshold detection and edge linking:

The gradient amplitude array processed by "non maximum suppression" is thresholded. A typical method to reduce the number of false edge segments is to use a threshold for G (x, y). Zero all values below the threshold.

The first is the marginal discriminant: if the edge intensity is greater than the high threshold value, it must be an edge point; if the edge strength is less than the low threshold value, it must not be an edge point; if the edge strength is greater than the low threshold value and less than the high threshold value, then whether there is an edge point in the adjacent pixels of the pixel, if there is, it is an edge point; if not, it is not an edge point.

The second is connected edge: the double threshold algorithm applies two thresholds T1 and T2 to the non maximum suppression image, so that two threshold edge images $G_1(x,y)$ and $G_2(x,y)$ can be obtained.

In the experimental part, we select the optimal threshold by adjusting the parameters.

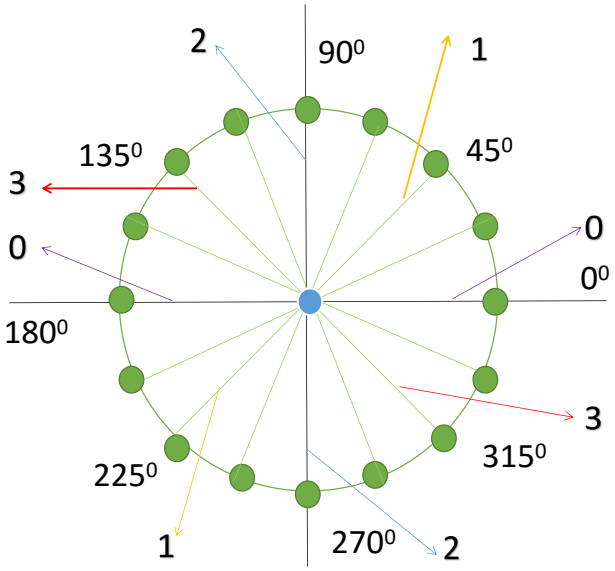

Figure 6.6 – Quantification of the gradient directions.

6.3.2.2/ ADAPTIVE CANNY AND MEDIAN FILTERING.

- ***Adaptive Canny***

An adaptive image edge detection algorithm based on canny operator is proposed in [85], its purpose is in the original image smoothing, filtering to remove or reduce the noise in the image. The image edge and noise are high frequency signal, the use of primitive Gaussian functions, will make the image edge fuzzy degree increase, which will make the subsequent detection process difficult. So, we proposed a modified Gaussian filtering method. The algorithm according to image pixel gray value mutation characteristics and the weights of the adaptive filter is changing, in smooth regions of the image edge sharpening and better to solve the noise smoothing, edge sharpening the filtering technique in the contradiction of, butt down edge detection to good pretreatment effect.

The calculation steps of adaptive filter are as follows:

(1) The K=0 iteration number is N, and set the value of the h parameter.

(2) Calculated gradient:

$$G_{x^k}(x,y) \ \ and \ G_{y^k}(x,y) \tag{6.24}$$

$$G_{x^k}(x,y) = \frac{1}{2}\left[f^k(x+1,y) - f^k(x-1,y)\right] \tag{6.25}$$

$$G_{y^k}(x,y) = \frac{1}{2}\left[f^k(x,y+1) - f^k(x,y-1)\right] \tag{6.26}$$

(3) Weight coefficients of the filter:

$$w^k(x,y) = exp\left[-\frac{\left[G_x^k(x,y)^2 + [G_y^k(x,y)^2\right]}{2h^2}\right] \tag{6.27}$$

(4) Weighted average of F(k)(x, y):

$$f^{(k+1)}(x,y) = \frac{\sum_{i=1}^{1}\sum_{j=1}^{1} f^{(k)}(x+i,y+1)\, w^{(k)}(x+i,y+i)}{\sum_{i=1}^{1}\sum_{j=1}^{1} w^{(k)}(x+i,y+1)} \tag{6.28}$$

(5) if k=K, then the end of iteration; Otherwise k=k+1.

It can be known from the above that the basic idea of the proposed adaptive smoothing filtering method is to iteratively convolve with the original image signal using a locally weighted template (the number of iterations is generally fixed). The weighting coefficient of each pixel point is changed at each iteration, and it is a gradient function of the pixel point. At the same time, the weighting coefficient of the filter also depends on the parameter h, which controls the amplitude of the mutation points to be retained during the iteration. This parameter guarantees the accuracy of image smoothing in different situations. In short, the weighting coefficient reflects the degree of continuity of the gray value of the image. After multiple iterations, the output image of the filter is composed of several uniform intensity regions. And there are good edges between these areas. Therefore, adaptive smoothing has two obvious effects: 1. Sharpen the edges of the area. 2. Smooth the inside of the area.

Simulation experimental results show that the smoothing effect of the adaptive filter is slow and gradual, and edge sharpening is required after a few iterations can, can achieve the desired smoothing, sharpening results. More iterative calculation will not have the most obvious improvement of the image, and will increase the amount of calculation, affect the efficiency. In this experiment, iteration number is set to 5, can achieve the desired results.

- ***Median filtering Canny***

From the analysis of the Canny algorithm can be seen using a canny edge extraction operator, Gaussian filter, its purpose is to the original image smoothing, to remove or reduce the noise in the image. Then we can try using median filter instead of Gaussian filter to the noise.

The median filter [55] is based on the order statistics theory, a can effectively restrain the noise of nonlinear signal processing technology. This filter has the advantages of simple operation and fast speed, noise filtering shows excellent performance. Median filter in filtering the noise at the same time can well protect the image edge and a good restoration of the image. In addition, median filter is easy to adapt, which can further improve the filtering performance. Therefore, it is very adapt to some linear filters cannot be competent to digital image processing applications.

The basic principle of median filtering is: First, determine a to a pixel as the center of the neighborhood, general square neighborhood, then in the neighborhood of each pixel of the sort the gray value, take the middle value as the new value of the center pixel. The neighborhood here we normally call from the window, when the window on the image for moving around, using the median filter algorithm can well for image smoothing. The

definition of median filtering algorithm in one dimension of the: When n is odd, the number of N $\{x_1,x_2,\ldots,x_n\}$,the median is according to the numerical order of size in the middle position number; When n is an even number, the average value of the definition of the two intermediate numbers of median. The median is represented by the symbols med $\{x_1,x_2,\ldots,x_n\}$.

For example: Med (1,3,4,0,6)= 3. In the two-dimensional median filtering algorithm is defined: Let X_{ij} be the gray value of each point in the digital image. Here (i,j) fetches a certain subset of Z^2. The filtering window is A, and its size is N = (2K + 1) · (2K + 1), and y_{ij} is the median value of window A in x_{ij}. then [55]:

$$y_{ij} = med\,(x_{i+r}, j+s, (r,s)\,\epsilon A) \tag{6.29}$$

The median filter is a neighborhood operation, which sorts the pixels in the neighborhood by gray level, and then selects the intermediate value in the group as the output pixel value.

Specific steps are as follows:

(1) Roam the template in the image, and overlap the center of the template with the position of a pixel in the image.

(2) Read the next gray value of the template for the pixel.

(3) Line up these gray values from small to large.

(4) Find the middle one of these values.

(5) Assign this intermediate value of the pixel at the center of the corresponding template.

The relationship between visual and median filtering: the image needs to be observed, to identify and understand. The image quality is closely related to the human eye's sensitivity. Because the human eye's perception of the image is mainly based on the human eye's rod cells, the rod cells in the display image is a number of cells, the median filtering of the image signal median filter, can be very good. To eliminate all kinds of random noise in the image, and doesn't affect the image to people's feeling. In the process of image transmission, external interference and system internal interference will bring a lot of random noise interference, the use of median filter can remove the interference, and the value of the image signal attenuation and the human eye does not affect the sense of image.

- ***Conclusion:***

  There are two key points in image edge detection: on the one hand, it is important to effectively reduce the influence of noise, and on the other hand, it is also crucial to select the correct threshold. Choosing the optimal threshold can not only reduce the influence of noise, but also has a good effect on the suppression of false edges. If the threshold is set too high, the edge may be broken. If the threshold is set too low, many false edges may appear in the extracted edges.

  Defects of the Canny algorithm: The traditional Canny operator calculates the gradient amplitude value by calculating the finite difference mean in a 2 * 2 neighborhood. It is more accurate for edge positioning, but too sensitive to noise. It is also easy to detect false edges and lose some details of the real edges. Overall. experimental results by Canny algorithm are poor.

To address this issue, we developed an improved canny algorithm from the conventional method. The proposed approach calculates the gradient amplitude by using an 8-pixel's neighborhood in the directions of X, Y, 135 degree and 45 degree as well. It also extends to multiple directions to calculate the gradient amplitude of the image. Not only that, we also choose the optimal threshold by adjusting the parameters and use it to judge whether the point is an edge point. In the following experimental part, we use the improved gradient amplitude algorithm Canny operator to detect more edge details. It not only retains effective edge information, but also obtains edge images very well. It has better performance than traditional gradient calculation methods. This method can effectively resist the interference of noise with high accurate detection results.

We will use the optimized Canny operator for the edge detection of the target head image. This is also based on the first information source we proposed to use multiple information sources. Next, we use image feature extraction and texture feature analysis to obtain more features of the target head image.

6.3.3/ APPROACHES BASED ON TEXTURE FEATURES

Texture is a visual feature that reflects homogeneity in an image. It reflects the surface structure, organization and arrangement properties of the object surface with slow or periodic changes. Texture has three major characteristics: a kind of continuous local repetition, non-random arrangement, and a uniform body in the texture area [48][80].

Unlike image features such as grayscale and color, texture is represented by the grayscale distribution of pixels and their surrounding spatial neighborhoods, that is, local texture information. In addition, the repetitiveness of local texture information to varying degrees is the global texture information. While the texture feature reflects the nature of global features, it also describes the surface properties of the scene corresponding to the image or image area. Because the texture is only a characteristic of the surface of an object, it does not fully reflect the essential attributes of the object. Therefore, it is impossible to obtain high-level image content using only texture features. Unlike color features, texture features are not pixel-based features, as they require statistical calculations in an area containing multiple pixels. In pattern matching, such regional features have great advantages, and will not fail to match successfully due to local deviations. When retrieving texture images with large differences in thickness, density, etc., using texture features is an effective method. But when there is little difference between easily-resolved information such as the thickness and density of textures, it is often difficult to accurately reflect the differences between textures with different human visual perception. For instance, reflections in the water and the effects of smooth metal surface reflections on each other can cause changes in texture. Because these are not the characteristics of the object itself, when applying texture information for retrieval, sometimes these false textures can cause "Misleading" retrieval.

Pros and the cons related to texture features:

- ***Advantage*** :
  - ***It performs statistical calculations in an area containing multiple pixels;***
  - ***It has rotation invariance;***

  - ***It has strong resistance to noise.***

- ***Disadvantages*** :
  - ***When the resolution of the image changes, the calculated texture may have a large deviation;***
  - ***The texture reflected from the 2-D image is not necessarily the true texture on the surface of the 3-D object;***
  - ***It may be affected by lighting and reflection conditions.***

The extraction of texture features generally involves setting a certain size window and then obtaining texture features from them. However, the choice of window has contradictory requirements: (1) The need for setting large windows: texture is a regional concept, which must be reflected by spatial consistency. The larger the observation window, the stronger the ability to detect identity; otherwise, the weaker the ability; (2) The window setting is small: because the boundaries of different textures correspond to the jump in the texture identity of the region. Therefore, in order to accurately locate the boundary, a smaller observation window is required; In this case, the difficulty will be: if the window is too small, it will cause mis-segmentation within the same texture; while the analysis window is too large, it will cause many mis-segmentations in the texture boundary area.

Here, we propose a texture feature fusion technique, namely HOG and LBP texture feature fusion. We apply texture, multi-feature fusion technology to solve these difficulties.

6.3.3.1/ HOG FEATURE BASED HEAD

Usually, we can use key points in the image for matching to detect objects in the image. These types of algorithms are useful when you want to detect objects that have many consistent internal characteristics and are not affected by the background. For example, these algorithms can achieve good results in face detection because faces have many consistent internal features that are not affected by the background of the image [57], such as eyes, nose, and mouth. However, these types of algorithms do not work well when trying to perform the more general object recognition, such as pedestrian detection in images. The reason is that people's internal characteristics are not as consistent as faces [24], because everyone's body shape and style are different. This means that everyone will have a different set of internal characteristics, so we need something that can describe a person more fully.

Detecting objects by the contours of an image is very challenging because we have to deal with the difficulties caused by the contrast between the background and the foreground. That's why we need HOG, histograms of Oriented Gradients, which were first proposed by Navneet Dalal and Bill Triggs in 2005 [12].

HOG represents the structural features of edges (gradients). It can not only describe the local shape information, the quantification of position and direction space, but also can suppress the effects of translation and rotation to a certain extent. Taking a normalized histogram in a local area can partially offset the effects of lighting changes. Because the effect of the color of the light on the image is ignored to a certain extent, the dimension of

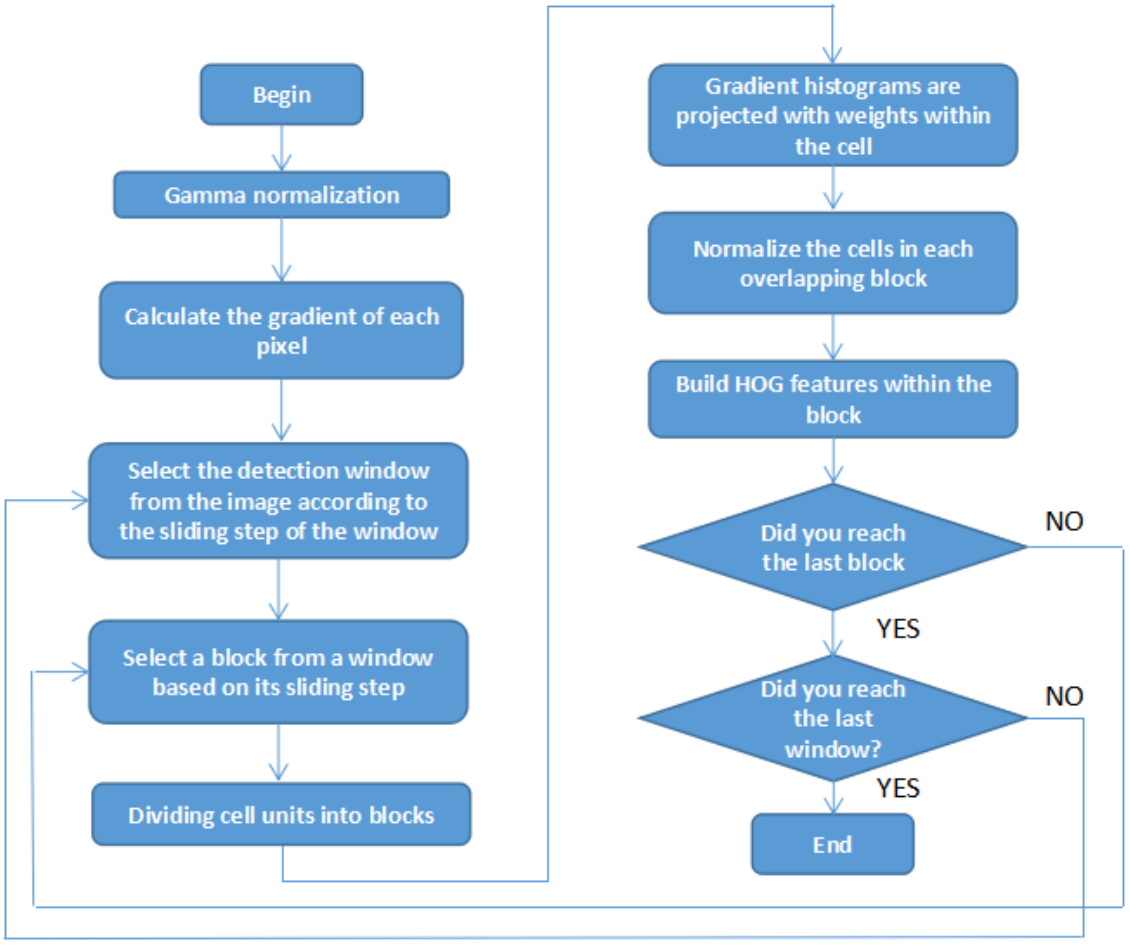

Figure 6.7 – HOG feature extraction flowchart.

the representation data required by the image is reduced. And because of its block-by-block processing method, the relationship between the local pixels of the image can be well represented.

Here, we introduce HOG texture features for human head image detection. First, we take one target image and convert it into gray-scale image, and treat the target image as a three-dimensional image (x, y, z).

In order to reduce the influence of illumination factors, the main purpose is to improve the robustness of the detector to illumination. First, the entire image needs to be normalized. In the texture intensity of the image, the proportion of local surface exposure contribution is relatively large. Therefore, our compression process can effectively reduce the local shadow and lighting changes of the image. Gamma compression formula [12]:

$$I(x,y) = I(x,y)^{Gamma} \tag{6.30}$$

When Gamma = 1, oblique 45° straight line, without correction, output = input; Gamma is greater than 1, the curve is pushed down, the output value is less than the input value; Gamma is less than 1, the curve is arched, the output value is greater than the input value; here the Gamma effective value , we take 0.5.

Then, we start to calculate the gradient value of the target image. We calculate the gradient of the image's abscissa and ordinate directions, and calculate the gradient direction value of each pixel position. The derivative operation can not only capture contours, silhouettes and some texture information, but also further weaken the influence of lighting. Here we have done a Gaussian smoothing. Image smoothing generally refers to the process of highlighting the low-frequeny components of an image while supressing noise which often correspond to high-frequency components. In this way, the image is smoothed, where abrupt changes inside are reduced. Generally in order to remove noise, we first use a discrete Gaussian smoothing template for smoothing, and the Gaussian function performs smoothing operations on grayscale images at different smoothing scales. Through experiments, the best human detection effect can be achieved without Gaussian smoothing, which reduces the error rate by about double. The reason for not doing the smoothing operation is that the image is edge-based. Smoothing will reduce the contrast of the edge information, thereby reducing the signal information in the image. The gradient of pixels (x, y) in the target image is:

$$G_x(x,y) = H(x+1,y) - H(x-1,y) \tag{6.31}$$

$$G_y(x,y) = H(x,y+1) - H(x,y-1) \tag{6.32}$$

where G_x, G_y represents the horizontal gradient, vertical gradient, and $H(x,y)$ pixel values at the pixel (x, y) in the input image.The gradient and gradient directions at the pixel (x, y) are:

$$G(x,y) = \sqrt{G_x(x,y)^2 + G_y(x,y)^2} \tag{6.33}$$

$$\Theta(x,y) = tan^{-1}(\frac{G_x(x,y)}{G_y(x,y)}) \tag{6.34}$$

Next, we are gradually changing the direction histogram for each cell. The purpose is to provide an encoding for the local image area. At the same time, a weak sensitivity to the pose and appearance of the human subject in the image can be maintained. We divide the image into several "cells", each cell consisting of 8*8 pixels and each block consisting of 2*2 cells. Here, we define the use of nine bin histograms to calculate 8*8 pixel gradient information, as shown in the figure:

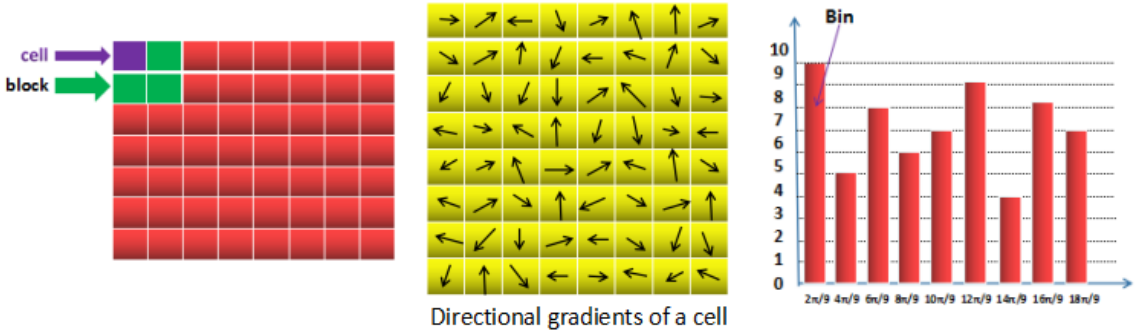

Figure 6.8 – The definition of a cell, a block and a bin.

We divide the gradient of the cell by 360° into nine directions. Due to changes in local illumination and changes in the foreground and background contrast, the range of gradient

intensity is very large. This requires normalizing the gradient intensity. Normalization can further illuminate light, shadows, and edge compression. The approach we take is to combine the individual cell units into large blocks and spatially connected intervals. We normalize the HOG feature vector in the block and introduce v to represent a vector that has not been normalized. It contains all histogram information for a given block, according to the $||V_k||$ standard [8], where k is 1 or 2, and e is a small constant. At this point, the normalization factor can be expressed as follows:

$$L2 - norm, f = \frac{v}{\sqrt{||V||_2^2 + e^2}} \tag{6.35}$$

Finally, we will detect all overlapping blocks in the window to collect HOG features, and combine them into the final feature vector for classification.

6.3.3.2/ LBP FEATURE

LBP (Local Binary Pattern) is an operator used to describe the local texture features of an image. The LBP feature has outstanding advantages such as gray invariance and rotation invariance. It was proposed by T. Ojala, M. Pietikäinen, and D. Harwood in 1994 [52]. LBP features are widely used in many fields of computer vision due to their simple calculation and good results. Not only that, LBP features are also applied in face recognition and target detection. Here, we apply it to head image detection.

LBP is a simple but very effective texture operator. It compares each pixel with its nearby pixels and saves the result as a binary number. The most important advantage of LBP is its robustness to changes in grayscale, such as illumination changes [7]. Its other important feature is its simple calculation, which makes it possible to analyze the image in real time. The basic LBP operator is defined as the 3*3 window [2]. Using the value of the center pixel of the window as the threshold, the gray value of the adjacent 8 pixels is compared with it. If the surrounding pixel value is greater than the central pixel value, the pixel value of the location is marked as '1'. Otherwise, it is '0'. In this way, the 8 points in the 3*3 neighborhood can be compared to produce 8-bit binary numbers (usually converted to decimal numbers [31]; there are 256 types of LBP codes), which is to get the LBP value of the pixel in the center of the window, and use this value to reflect the texture information of the area, for example: 00010011. Each pixel has 8 adjacent pixels, and 28 possibilities.

The basic LBP feature for a given pixel is formed by thresholding the 3x3 neighbourhood with the center pixel value as the threshold, where(Xc,Yc) is the center pixel, ic is the intensity of the center pixel and in (n=0,1,2....7) pixel intensities from the neighborhood. The LBP is given by:

$$LBP = \sum_{n=0}^{P-1} s\,(i_n - i_c) * 2^n \tag{6.36}$$

where: P is the number of sample points and:

$$s\,(x) = \begin{cases} 1 & if \quad x \geq 0 \\ 0 & else \end{cases} \tag{6.37}$$

The LBP could be interpreted as an 8-bit integer. The basic LBP concept is shown in the figure:

Figure 6.9 – Illustration of the standard LBP operator.

When the LBP operator is used for texture classification or face recognition, the statistical histogram of the LBP mode is often used to express the image information. However, more varieties in the pattern would require larger yet sparser histograms. Therefore, it is necessary to reduce the dimension of the original LBP mode so as to best represent the image information when considering efficient data usage. In order to solve the above problem, we proposed a "Uniform Pattern" to reduce the dimension of LBP operator's pattern. In an actual image, most LBP patterns only contain at most two transitions from '1' to '0' or from '0' to '1'. When a loop binary number corresponding to an LBP change from '0' to '1' or from '1' to '0' at most twice, the binary corresponding to the LBP is called an equivalent pattern class. For example, 00000000 (0 jumps), 00000111 (only one transition from '0' to '1'), and 10001111 (first jump from '1' to '0', then '0' to '1', thus two jumps in total) are all equivalent modes class. The modes other than the equivalent mode class are classified as another class, called a mixed model class, such as 10010111 (a total of four transitions). With this improvement, the variety of binary patterns is greatly reduced without losing any information. The number of modes is reduced from the original 2P to P (P-1)+2, where P represents the number of sampling points in the neighborhood set. For the 8 sampling points in the 3*3 neighborhood, the binary pattern is reduced from the original 256 to 58. This allows the feature vector to have fewer dimensions and can reduce the effects of high-frequency noise.

For the original LBP feature, the gray value of the fixed neighborhood is used. Therefore, when the scale of the image changes, the encoding of the LBP feature will be wrong. The LBP feature will not correctly reflect the texture information around the pixels, and we optimized it again. The biggest drawback of the basic LBP operator is that it only covers a small area within a fixed radius, which obviously cannot meet the needs of different sizes and frequency textures. In order to adapt to the texture features of different scales, and to meet the requirements of intensity and rotation invariance, we extended the 3*3 neighborhood to any neighborhood and replaced the square neighborhood with a circular neighborhood. The improved LBP operator allows any number of pixels in a circular neighborhood of radius R. Thus, an LBP operator containing P sampling points in a circular region with a radius R is obtained as shown below:

We define the central point calculation formula:

$$x_p = x_c + Rcos\left(\frac{2\pi p}{p}\right) \tag{6.38}$$

$$y_p = y_c - Rsin\left(\frac{2\pi p}{p}\right) \tag{6.39}$$

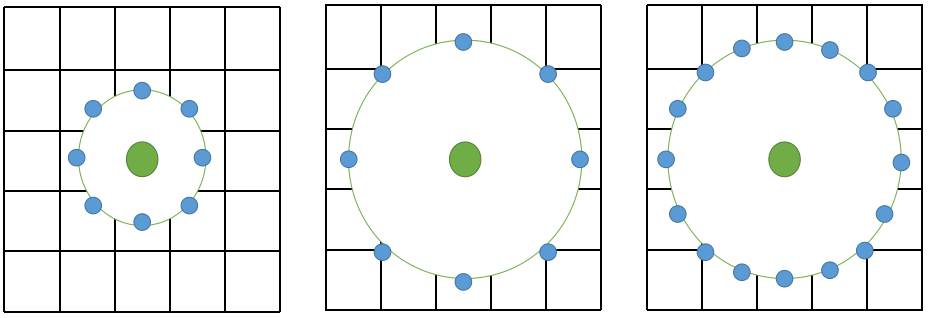

Figure 6.10 – LBP operator for P sampling points.

Since x_p and y_p may not be integers, they may not correspond to any pixel in an image. Here, we use bilinear interpolation. Bilinear interpolation extends from an interpolation function with two variables. The core idea is to perform a linear interpolation in each direction. Here we give an example, as shown below.

Given the red data points and the green points to be interpolated, here, the target is to get the value of the unknown function f at point P = (x, y). Suppose we know the value of the function f at four points Q11 = (x1, y1), Q12 = (x1, y2), Q21 = (x2, y1), and Q22 = (x2, y2). Then, we first perform linear interpolation in the x direction to obtain R1 and R2, and then perform linear interpolation in the y direction to obtain the value of point P. In this way, the desired result f (x, y) is obtained, in which the red points Q11, Q12, Q21, Q22 are 4 known pixel points. The coordinates of the four known points are (0, 0), (0, 1), (1, 0), and (1, 1), then the interpolation formula is:

$$f(x,y) = f(0,0)(1-x)(1-y) + f(1,0)x(1-y) + f(0,1)(1-x)y + f(1,1)xy \tag{6.40}$$

The formula for bilinear interpolation is simplified as follows:

$$f(x,y) \approx [1-x \;\; x] \begin{bmatrix} f(0,0) & f(0,1) \\ f(1,0) & f(1,1) \end{bmatrix} \begin{bmatrix} 1-y \\ y \end{bmatrix} \tag{6.41}$$

In this case, any value of point lying within Q1∼Q4 can be obtained via extrapolation. Bilinear interpolation is very suitable for us to calculate the effective value of the sampling point P. We get 8 sampling points. If the original LBP features are used, and the LBP feature value modes are 256, the LBP feature vector dimension of an image is: 64 * 256 = 16384 dimensions. If we use the LBP feature of our optimized equivalent model, the LBP value has 59 modes. The feature vector dimension is: 64 * 59 = 3776 dimensions. It can be concluded that the dimensionality of the feature vector is greatly reduced using the LBP feature of the optimized equivalent model. This means that the learning time using machine learning methods will be greatly reduced, and the performance will not be greatly affected. At the same time, the impact of high-frequency noise is also reduced.

Here, we are inspired by the statistical histogram of LBP features proposed by Ahonen et al. The LBP feature image is divided into H local blocks, and a histogram of each local block is extracted, and then these histograms are sequentially connected together to form

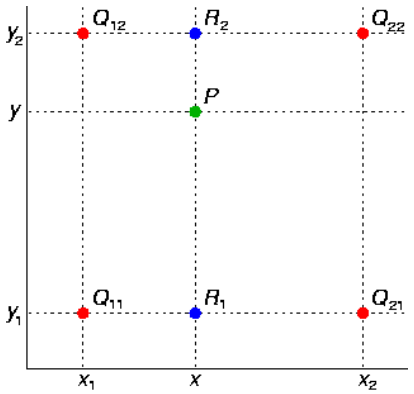

Figure 6.11 – The bilinear interpolation method calculates the value of the unknown point P.

a statistical histogram of LBP features, that is LBPH. Through the statistical histogram of LBP features, we can make good use of finding head features.

6.3.3.3/ MY FIRST CONTRIBUTION: COMBINING EDGE DETECTION AND FEATURES EXTRACTION FOR IMAGE ANALYSIS

We propose a crowd counting method for multisource feature fusion. Image features are extracted from multiple sources, and the population is estimated by image feature extraction and texture feature analysis, along with for crowd image edge detection. We count people in high-density still images. For instance, in the city's squares, sports fields, subway stations, etc. Our approach uses a still image taken by a camera on a drone to appraise the count in the population density image, using a kind of sources of information: HOG, LBP, CANNY. We furnish separate estimates of counts and other statistical measurements through several types of sources. Support vector machine (SVM), classification and regression analysis, along with obtaining a high density population, reasonable early warning, to ensure the safety of the population. The procedure is as follows:

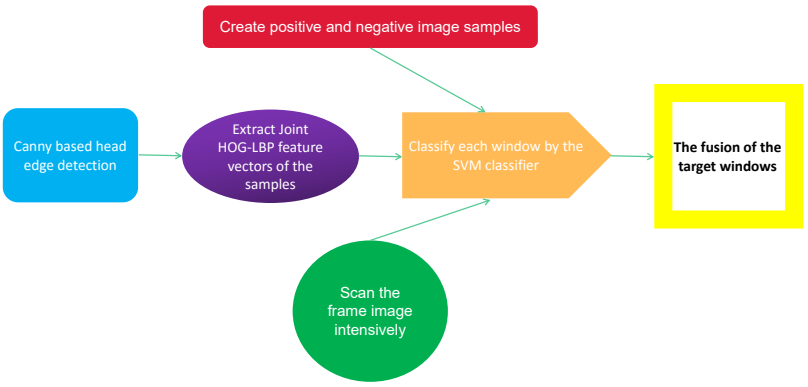

Figure 6.12 – Crowd counting.

Feature extraction is one of the most critical aspects of human head detection [73]. Extracting features with distinguishing the significance plays an important role in the accurate detection of the human head. Our work integrates the features of HOG and LBP,

which not only combines the effective identification information of multiple features, but also eliminates most of the redundant information, thereby realizing effective compression of information, saving information storage space, and facilitating the acceleration of operations and real-time processing of information. Here we use a serial fusion approach, as shown in Figure 6.13:

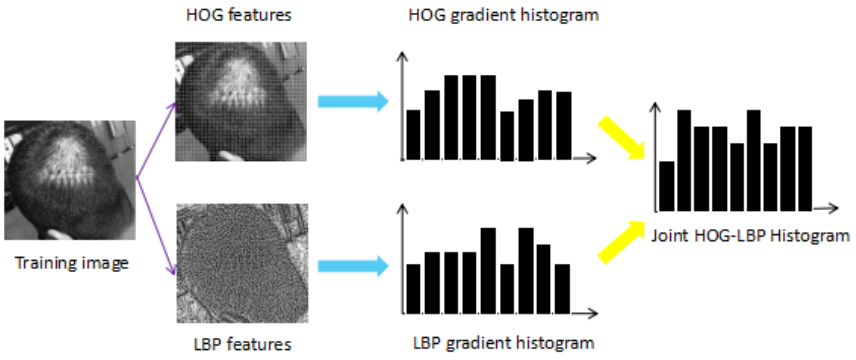

Figure 6.13 – Joint HOG-LBP Histogram.

Here we apply machine learning algorithms. It is an algorithm that automatically obtains ruling from data and uses rules to predict unknown data. It involves a lot of statistical theory and so on. A specific machine learning algorithms for pedestrian detection and head and shoulder detection are: AdaBoost algorithm and support vector machine (SVM)[1]. Here, we compare AdaBoost algorithm and SVM, which one can be better applied to head detection.

The first is the AdaBoost algorithm. Its basic principle is to combine multiple weak classifiers (weak single class decision trees are generally used) to make it a strong classifier [92]. Adaboost uses an iterative idea. Each iteration trains only one weak classifier. The trained weak classifier will participate in the use of the next iteration. That is to say, in the Nth iteration, there are N weak classifiers in total, of which N-1 are previously trained, and its various parameters are not changed, and the Nth classifier is trained this time. The relationship between weak classifiers is that the Nth weak classifier is more likely to pair the unpaired data of the first N-1 weak classifiers. The final classification output mainly depends on the comprehensive effect of these N classifiers.

Adaboost generally uses a single-level decision tree as its weak classifier [13]. A single-level decision tree is the most simplified version of a decision tree, with only one decision point. In other words, if the training data has multi-dimensional features, a single-layer decision tree can only select one-dimensional features to make a decision, and there is another key point, the decision threshold needs to be considered.

The Adaboost algorithm includes two weights, one is the weight of the data, and the other is the weight of the weak classifier. Among them, the weight of the data is mainly used by the weak classifier to find the decision point with the smallest classification error. After finding it, Adaboost uses this minimum error to calculate the weight of the weak classifier. The larger the weight of the classifier, the weaker the classifier has the greater decision power in the final decision. For example, for a total of 10 points, if the weight of each point is 0.1, and a 1 error is added, then the error rate is increased by 0.1; Likewise, if 3 errors are added, then the error rate is 0.3. Let's assume there are ten samples and each has a different weight as follows: [0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01,

0.01, 0.91]. If the first point is wrong, the error rate is 0.01. If the third point is wrong, the error rate is 0.01. If the last point is wrong, the error rate is 0.91. In this way, when selecting decision points, it is natural to try to pair the points with large weights as much as possible to reduce the error rate.

In the Adaboost algorithm, we adjust the weight after each weak classifier is trained, and the weight of the misclassified points in the previous round of training will increase. In this round of training, due to the influence of weights, the weak classifiers in this round will be more likely to pair the misclassified points from the previous round. If there is still no pairing, the weight of the wrong points will continue to increase. Weak classifiers will pay more attention to this point and try to pair them as much as possible. In this way, the next classifier mainly focuses on the unpaired points of the previous classifier, and each classifier has its own focus. Therefore, in Adaboost, we use each weak classifier to find samples it classifies poorly. Each weak classifier only pays attention to a part of the entire data set, so they must be combined together. When voting for the final decision, weighted voting needs to be performed according to the weight of the weak classifier, and the weight is calculated based on the classification error rate of the weak classifier. The general rule is that the lower the error rate of a weak classifier, the higher its weight.

Like Adaboost, SVM also has a same effect. SVM was first proposed by Vapnik et al [70]. It shows many unique advantages in solving small samples, nonlinear and high-dimensional pattern recognition, and can be applied to other machine learning problems such as function fitting.

The SVM method is based on statistical theory. It finds the best balance between model complexity and learning ability based on limited sample information. Complexity includes the learning accuracy of a specific training sample, and the learning ability is the ability to accurately identify any sample. The SVM method uses a non-linear mapping p. It works by mapping the sample space into a high-dimensional or even infinite-dimensional feature space. The non-linearly separable problem in the original sample space is transformed into a linearly separable problem in the feature space. To put it simply, it is dimensionalization and linearization. Dimension-raising is to map samples to high-dimensional space. Generally, the computational complexity will increase, so it is rarely used this way. SVM solves this problem through the expansion of kernel functions and calculation theory. At the same time, different kernel functions can be selected to generate different SVMs. The commonly used kernel functions are the following 4 equations as proposed by Vapnik et al [70].

The first is a linear kernel function:

$$K(x, y) = x \cdot y \tag{6.42}$$

The second is a polynomial kernel function:

$$K(x, y) = [(x \cdot y) + 1] \cdot d \tag{6.43}$$

The third is the radial basis function:

$$K(x, y) = exp\left(-|x - y|^2 / d^2\right) \tag{6.44}$$

The fourth is the kernel function of the two-layer neural network:

$$K(x, y) = tan(a(x \cdot y) + b) \tag{6.45}$$

The main idea of SVM can be summarized as two points:

The first point: analysis of linearly separable cases. In the case of linear inseparability, a linearly inseparable sample of the low-dimensional input space is transformed into a high-dimensional feature space to make it linearly separable by using a nonlinear mapping algorithm. This makes it possible to perform a linear analysis of the nonlinear features of the samples using a linear algorithm in the high-dimensional feature space.

Second point: Based on the theory of structural risk minimization, an optimal segmentation hyperplane is constructed in the feature space, so that the learner is globally optimized.

The goal of SVM is mainly based on the principle of structural risk minimization. SVM constructs an objective function to distinguish the two types of patterns as much as possible. There are usually two cases: linearly separable and linearly inseparable.

Comparison between two different classifiers: AdaBoost and SVM: In the former, a new weak classifier is added each round until a predetermined sufficiently small error rate is reached. At the same time, the wrong samples from the previous classifier are used to train the next classifier. In terms of training speed, AdaBoost algorithm is slower. In the latter case, the SVM method has good generalization. SVM can solve the problem of sample learning well. And it is an ideal method for the class two image classification.

Here, we use support vector machines to achieve the optimal classification of linearly separable data. For a linear SVM with the training samples $((x_i, y_i)), 1 \leqslant i \leqslant N\}$, where x_i is the ith instance sample, y_i is the corresponding category labels (i.e.,the expected response), its decision surface equation can be expressed as [70]:

$$\omega \cdot x + b = 0 \tag{6.46}$$

Where x is the input vector, ω is the dynamically variable weight vector,and b is the offset. In essence to find an optimal classifier is to find an optimal hyper plane, according to formula (5.51), which can not only separate two classes correctly but also maximize the intra-class distance. Here, the optimal hyperplane is the plane with the largest distance from the vector of each type of data. We obtain it through the second optimization:

$$min\Phi(W) = \frac{1}{2}\|W\|^2 \tag{6.47}$$

Here, the constraints satisfied are:

$$y_i(W \cdot x_i + b) \geq 1, i = 1, 2, 3, ..., n. \tag{6.48}$$

When the number of features is particularly large, this quadratic programming problem can be transformed into a dual problem:

$$maxW(a) = \sum_{i=1}^{n} a_i - \frac{1}{2}\sum_{i,j=1}^{n} a_i a_j y_i y_j \left(x_i \cdot x_j\right) \tag{6.49}$$

$$W^* = \sum_{i=1}^{n} a_i y_i x_i \tag{6.50}$$

$$b^{'} = y_i - w \cdot x_i \tag{6.51}$$

Here, it meets the constraints:

$$\sum_{i=1}^{n} a_i y_i = 0, a_{i \geq 0}, i = 1, 2, 3, ...n. \tag{6.52}$$

Here a = (a1, a2, ... an) is a Lagrange multiplier, W^* is the normal vector of the optimal hyperplane, and $b^{'}$ is the offset of the optimal hyperplane. Then, in the solution and analysis of this type of optimization problem, its solution must satisfy:

$$a_i \{y_i (w \cdot x + b) - 1\} = 0, i = 1, 2, ..., n. \tag{6.53}$$

It can be known from formula (6.53) that those samples with ai = 0 have no effect on classification, and only those samples with ai> 0 have effect on the classification. These samples are called support vectors.

Accordingly, support vectors refer to the training sample points located in the classification boundaries, which are the key elements of the training sample set. Based on these theories and concepts, the following formula is used to classify the input samples [70]:

$$f(x) = sgn \left\{ \sum_{i=1}^{n} a_i y_i (x_i \cdot x) + b \right\} \tag{6.54}$$

where ai is the weight coefficient corresponding to the support vector xi.

The next step, we connect the sample HOG feature vector and the LBP feature vector in series to form a joint feature vector as input to SVM. Here, in the classification process, the linearly inseparable low-dimensional space is converted into a linearly separable high-dimensional space mainly through SVM kernel functions. Cross-validation method is used to select the SVM optimal parameters, so that the classifier has the highest classification accuracy of the input training samples. In accordance with the above method, the training process for joint HOG-LBP SVM classifiers are shown in Figure 6.14:

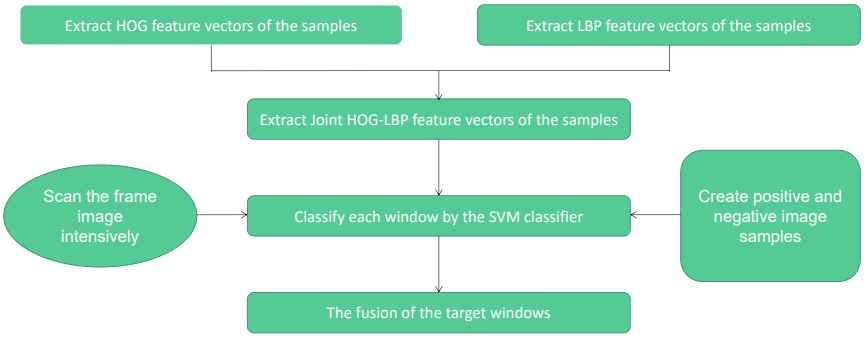

Figure 6.14 – Training process for joint HOG-LBP SVM classifiers.

- ***Conclusion:*** HOG features have gradient characteristics, resulting in high feature dimensions, too much redundant information, and poor description of gradient space characteristics. The binary coding strategy of LBP feature improves its

robustness to scale, rotation, illumination and noise. The algorithm complexity of the nonlinear kernel SVM classifier is large, and the real-time detection needs to be improved. Therefore, based on the Joint HOG + LBP algorithm framework, we have separately improved the HOG and LBP features analyzed above, and jointly improved the features. At the same time, based on the classifier with relatively low complexity and better real-time performance, relevant research on the method of extracting head features has been carried out. First, we introduce the technique of extracting head features from three aspects: research background, research significance and research status. Based on the framework of the head detection algorithm, the feature extraction algorithm and classifier are introduced and summarized respectively. Then, the rotation invariance and scale invariance of LBP features need to be improved and the HOG feature dimension contains too much redundant information. We have proposed feature serial fusion technique, which is Joint HOG + LBP SVM technique. Through feature serial fusion technique, we construct the final feature vector of the image that is more robust to rotation and scale, has higher utilization of texture information, and has lower dimensions. At the same time, combined with SVM classifier, a multi-feature fusion head feature detection algorithm based on dimension reduction is designed. In order to verify the effectiveness of the extracted features and feature dimensionality reduction, we explain in detail in the experimental part. The results of the experimental part prove that the head detection algorithm based on the Joint HOG + LBP feature classifier has overall superiority. Here, HOG and LBP are also the second and third information sources we proposed to use multiple information sources. Our work integrates the features of HOG and LBP, which not only combines the effective identification information of multiple features, but also eliminates most of the redundant information, thereby realizing effective compression of information, saving information storage space. In terms of crowd counting, we use multiple sources of information, namely HOG, LBP and CANNY. These sources provides separate estimates of the number of counts and other statistical measures, through the support vector Machine SVM, classification, and regression analysis to obtain high-density populations. In the experimental part, we have a detailed introduction and comparison of experimental data to prove the superiority of our method.

6.4/ EXPERIMENT

Our experiment is mainly divided into three parts: the first part is pedestrian detection, the second part is the optimized Canny operator; and the third part is counting the crowd in images.

The first part of the experiment:

The training set we used contains 5000 head face images clipped manually and enough non-head face images from several sample sets including INRIA, PETS2000, and MIT. During training, negative samples can be selected automatically from the background images. The test set contains 500 images with or without pedestrians, including about 1,500 apparent head faces and covering various scenes, angles, postures, and clothing, etc. The algorithm is implemented in a program developed with Matlab 2017a function library.

Extract sample HOG and LBP features:

Sample HOG feature calculation steps: For each positive and negative sample set, each corresponds to a 32*32 grayscale image (in this case, the grayscale picture is used to reduce computation cost by 3 while retaining fine detection when compared to color images, and it does not alter the final detection result significantly), and the rectangular HOG feature is calculated. New paragraph: The set cell size is 8*8, the size of the block is 16*16, and the slide step size is the width of a Cell. The specific process of HOG feature calculation is as follows: To reduce the influence of lighting, the sample is first Gamma standardization of images. We then calculate the gradient of x and y directions for each pixel in the grayscale image, and use the [-1,0,1] template to calculate the direction and amplitude of the gradient. In each Cell, we set the projection direction to 9 bins, and use the gradient magnitude of each pixel as the weight, and vote to count the weighted histograms of gradient directions of each Cell. The dimension of this histogram is 9. Four cells in a Block (with overlaps between blocks) are normalized using L2-norm, and the gradient histograms of four cells are counted, resulting in a dimension of 36. Finally, all blocks in the image are concatenated, and the dimension of the obtained HOG feature vector is calculated.

Sample LBP feature calculation steps: For each positive and negative sample set, each size is a 32*32 grayscale image, and LBP feature extraction based on a sliding window is used. The general description of the sliding window for the image algorithm is as follows: In an image of size W*H, the w*h window (W»w, H»h) is moved according to a certain rule, and a pixel in the window is performed. In the series operation, the window moves one step to the right or down after the operation is completed, until the entire image is processed. Set the size of the window to 16*16, and set the window's horizontal and vertical sliding steps to half the width of the window. The specific process of the LBP feature calculation is as follows: For one pixel in each window, an LBP feature value is calculated using an operator LBP (representing a radius 1, a ring containing 8 neighborhoods, a uniform mode). According to the LBP eigenvalue calculated in the window, the histogram of each window is calculated, that is, the number of occurrences of each LBP eigenvalue, and then L2-norm is used for normalization. The statistical histograms of all windows in the tandem image, the dimensions of the resulting LBP eigenvectors.

Finally, the sample HOG feature vector and the LBP feature vector are connected in series to form a joint feature vector. We use the SVM classifier to transform the linearly inseparable, low-dimensional space into a linearly separable, high-dimensional space by using a kernel function. The cross-validation method is used to select the optimal SVM parameters so that the classifier pairs the input training samples. The highest classification accuracy. The experimental test image size is 384*288. The algorithm is modified based on Matlab 2017a and runs on an Inter Core i5-5250 (1.60 GHz), 4 GB RAM computer. The experimental results are shown in table 6.1.

Table 6.1 – Algorithm performances shown by 3 experiments.

Detection algorithm	Test sequence	False number	Detection rate
Dalal HOG	1	39	92.2%
T.Ojala LBP	2	32	93.6%
Joint HOG+LBP	3	17	96.6%

The second part of the experiment:

Here, we implement the adaptive filtering to optimize the Gaussian filtering, and the median filtering replaces the Gaussian filtering and we propose to optimize Canny operator. At the same time, we also added the noise factor to the target image. We conducted comparisons of different sets of experimental data. The results are as follows figure 6.15:

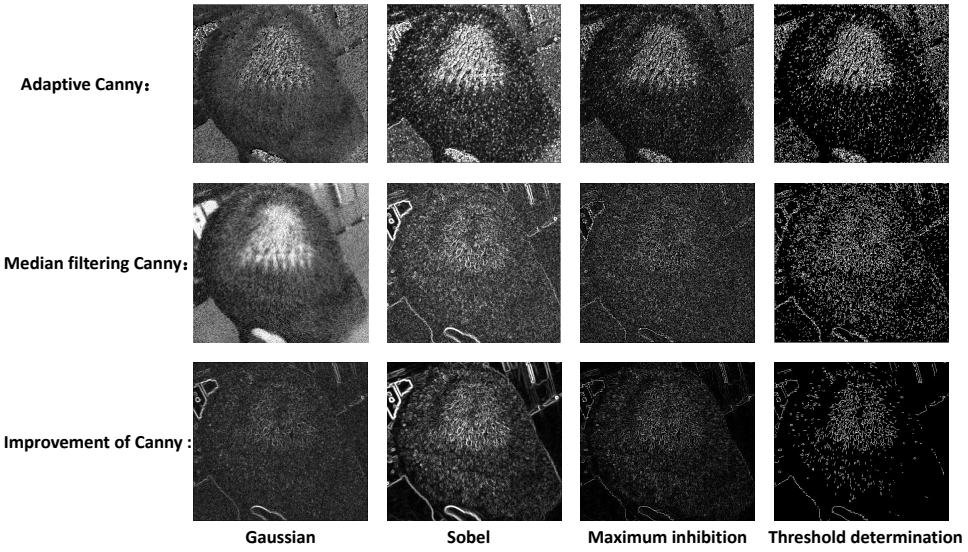

Figure 6.15 – Edge detection using different operators and comparison with the Canny operator.

Through the experimental results: the optimization of the new Canny operator is proposed. By performing Gaussian filtering, the purpose is to smooth the original picture to remove or reduce noise in the image. Then, the first order partial derivative of finite difference to calculate the gradient magnitude and direction of the expansion, the calculation of four directions, effectively improve the canny edge detection accuracy. Then, non-maxima suppression through inhibition of gradient direction on all non-roof peak gradient magnitude to refine the gradient magnitude. Finally, non-maxima suppression processing after the gradient amplitude array threshold value of the high and low threshold edge points, reduce the false edge segments Number, get the image. we can see that the improved canny algorithm has a better edge detection effect compared to the traditional canny algorithm. It has a certain improvement in the image target extraction. It can improve the performance of canny operator in extracting image edge detail information and suppressing false edge noise. The obtained edge contour map has a high signal-to-noise ratio and good connectivity. Experiments prove that this is a better improved algorithm.

The third part of the experiment:

Here, we cite the data set of Experiment 1: HOG, LBP samples. At the same time, we also extract sample CANNY head contour edge detection. We use high and low thresholds to find edge points, reduce the number of false edge segments, and get the sample head contour image.

Firstly, we add CANNY edge detection to extract the outline of the head. Then, the sample HOG feature vector and the LBP feature vector are connected in series to form a joint feature vector. We use the SVM classifier to transform the linear indivisible, low-

dimensional space into a linearly separable, high-dimensional space by using a kernel function. The cross-validation method is used to select the optimal SVM parameters so that the classifier pairs the input training samples for a higher classification accuracy. At the same time, we also use Sobel, Roberts, Prewitt, Log, Laplace instead of the optimized canny operator to extract the head contour and compare the experimental results. The experimental test image size is 384×288. The experimental results are shown in Table 6.2:

Table 6.2 – Comparison of experimental results based on joint different edge detection operators.

Detection algorithm	Test sequence	False number	Detection rate
Joint HOG+LBP+Sobel	1	69	86.6%
Joint HOG+LBP+Roberts	2	75	85.1%
Joint HOG+LBP+Prewitt	3	96	80.9%
Joint HOG+LBP+Log	4	112	78.1%
Joint HOG+LBP+Laplace	5	108	78.1%
Joint HOG+LBP+CANNY	6	14	97.2%

Through the comparison of experimental data, we can find that Joint HOG+LBP+CANNY has a very good effect on the detection and extraction of head features. Then, we once again compare the HOG human detection proposed by Dalal et al. And the LBP human detection proposed by T. Ojala et al. The experimental results are shown in Table 6.3:

Table 6.3 – Algorithm performances shown by 4 experiments.

Detection algorithm	Test sequence	False number	Detection rate
Dalal HOG	1	39	92.2%
T.Ojala LBP	2	32	93.6%
Joint HOG+LBP	3	17	96.6%
Joint HOG+LBP+CANNY	4	14	97.2%

We divided the images of the tested crowd into different regional patches. For example, for a group of 256*512 pixel crowd pictures, we can set each area module as: 16 * 16 pixels, 32 * 32 pixels, 64 * 64 pixels and 128 * 128 pixels, as shown in the figure 6.16:

The crowd density is between 0-500 people, we have tried many times and got the results given in Table 6.4. In the table 6.4 are given the number of people detected, number of people actually present in the scene, the difference between the detected number of people and the actual number of people and time. Based on the above results, the precision calculated is 92.85%.

Then, we once again test the crowd density image between 500-1500 people.

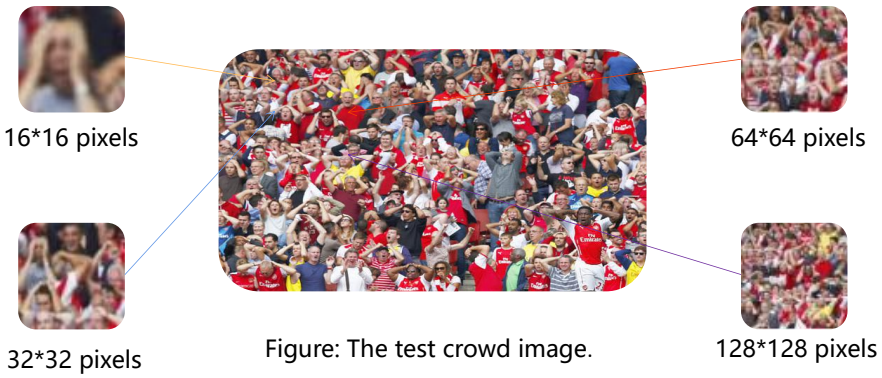

Figure 6.16 – This figure shows the size of different pixel blocks used in these experiments.

Table 6.4 – The experimental result of the crowd density is between 0-500 people.

Test sequence	Number of people detected	Actual	Difference	Time	Detection rate
1	220	280	60	6.13s	78.57%
2	231	280	49	8.56s	82.50%
3	260	280	20	14.25s	92.85%
4	257	280	23	13.13s	91.78%
5	245	280	35	9.51s	87.50%

Table 6.5 – The experimental result of the crowd density is between 500-1500 people.

Test sequence	Number of people detected	Actual	Difference	Time	Detection rate
1	536	829	293	7.13s	64.65%
2	889	1256	367	10.28s	70.78%
3	938	1480	542	11.38s	63.37%
4	624	963	339	8.23s	64.79%
5	689	1045	356	9.25s	65.93%

Through the experimental results, we can clearly see that as the number of people increases, the detection accuracy rate decreases. This shows that our algorithm has achieved good results in the crowd density image of 0-500 people. For high-density crowd images, when 500-1500 people, our detection rate is decreasing.

6.5/ CONCLUSION

Image feature extraction and texture feature analysis methods, data obtained from multiple sources are used to count people in the crowd. Therefore, we use multiple sources of information, namely, HOG, LBP and Canny's edge detector to extract such features. These sources provide separate estimates and other combinations of statistical measurements. Using the support vector machine (SVM) classification technique, and regression

analysis, we count the density crowd. The approach adopted is easy and fast in processing.

Our experiments show that when the crowd density is 0-500 people, this method has a good effect in crowded scenes. When the crowd density is 500-1500 people, as the number of people increases, the detection accuracy rate decreases. Next, we propose a convolutional neural network based method to achieve high-density crowd counting.

III

CONTRIBUTION B

7

MULTI-FEATURE COUNTING OF DENSE CROWD IMAGE BASED ON MULTI-COLUMN CONVOLUTIONAL NEURAL NETWORK

The concept of deep learning stems from the study of artificial neural networks. Multilayer perceptron with multiple hidden layers is a deep learning structure. Deep learning forms a more abstract high-level representation attribute category or feature by combining the underlying features to discover the distributed feature representation of the data. The concept of deep learning was proposed by Hinton et al. In 2006. Based on the Deep Belief Network (DBN), an unsupervised greedy layer-by-layer training algorithm is proposed, which brings hope for solving optimization problems related to deep structures, and then they proposed a multi-layer autoencoder deep structure. In addition, the convolutional neural network proposed by LeCun et al. is also the first true multi-layer structure learning algorithm, which uses spatial relative relationships to reduce the number of parameters to improve training performance.

In the field of deep learning, algorithms that have established great progress include deep convolutional networks (DNN) and recursive networks (RNN). Great success has been achieved in the areas of image recognition, video recognition, and speech recognition. It is precisely these successes that can contribute to the current deep learning boom. In the field of deep learning research, the most popular are production network architectures such as AutoEncoder, RBM, and DBN. For example, the application of AutoEncoder in the field of image and video search, and the processing of unstructured data by RBM. The DBN network combines the two major schools of connectionism and symbolism in the field of artificial intelligence, and they have great prospects.

7.1/ DEFINITION OF DEEP LEARNING

Convolutional neural networks (CNN) is one of the most successful areas for the application of deep learning algorithms. Convolutional neural networks include 1-D, 2-D and 3-D variants. One-dimensional convolutional neural networks are mainly used for sequence-type data processing. Two-dimensional convolutional neural networks are often used for 2D conv net should be generic enough for general images recognition. Three-dimensional

convolutional neural networks are mainly used for medical image and video data recognition.

So what is a neural network? The neural network here, also referred to as Artificial Neural Networks (ANNs), is an algorithmic mathematical model that mimics the behavior characteristics of biological neural networks. It consists of neurons, nodes and connections (synapses) between nodes, as shown below:

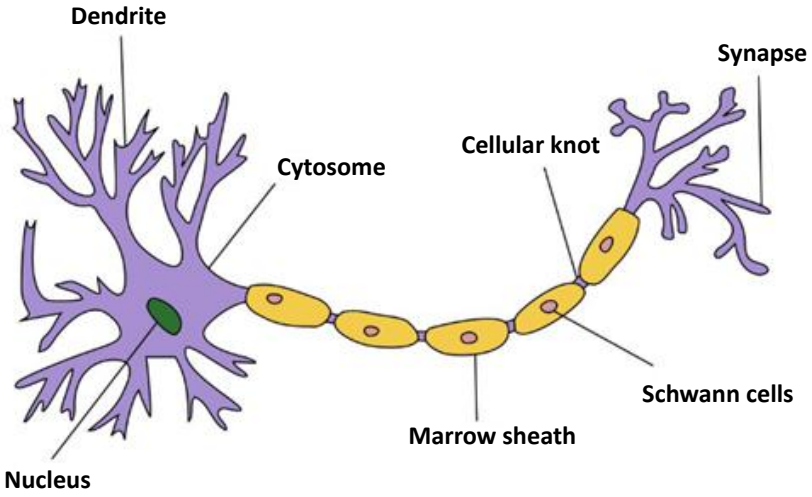

Figure 7.1 – Neural Networks.

The mathematical model abstracted by each neural network unit is as follows, also known as a perceptron, which receives multiple inputs (x1, x2, x3 ...) and generates an output, which is like the nerve endings feeling various external environmental changes (eg, external stimuli), and then generate electrical signals to facilitate transduction to nerve cells (also called neurons). An example is shown below:

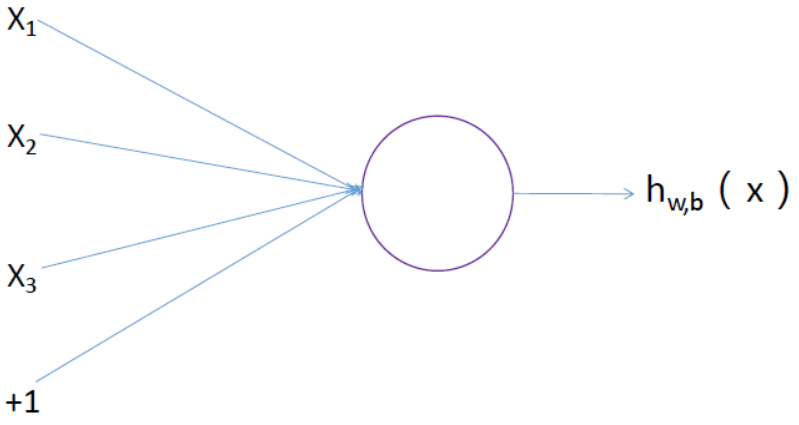

Figure 7.2 – Output of the neural network unit.

A single perceptron can constitute a simple model. However, in the real world, the actual decision-making model is much more complicated, often a multi-layer network composed of multiple perceptrons, as shown in the figure below. This is also a classic neural network model. It consists of an input layer, a hidden layer, and an output layer, as shown on Figure 7.3:

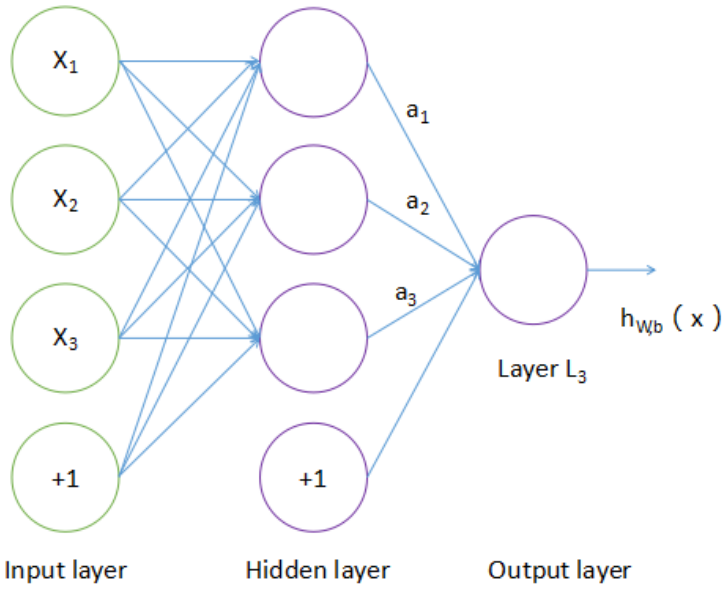

Figure 7.3 – Classic neural network model.

Artificial neural networks can map arbitrarily complex non-linear relationships, have strong robustness, memory ability, self-learning capabilities, and have a wide range of applications in classification, prediction, and pattern recognition.

7.2/ CONVOLUTIONAL NEURAL NETWORK

Convolutional neural networks have been developed in recent years and have attracted widespread attention as an efficient identification method. In the 1960s, Hubel and Wiesel, while studying neurons in the cerebral cortex of cats for local sensitivity and direction selection, found that their unique network structure can effectively reduce the complexity of feedback neural networks [61]. Therefore, a convolutional neural network (CNN) was proposed. Now, CNN has become one of the research hotspots in many scientific fields, especially in the field of pattern class. Because the network avoids complex preprocessing of the image and can directly input the original image, it has been more widely used. The new recognition machine proposed by K. Fukushima in 1980 was the first implementation of a convolutional neural network [16]. Subsequently, more researchers have improved the network.

It sounds like a strange combination of biology and mathematics, but these networks have become one of the most influential innovations in computer vision. 2012 was the first year of the growth of neural networks. Alex Krizhevsky used them to win the ImageNet competition of the year, reducing the classification error record from 26% to 15%. Since then, many companies have been doing deep learning with service at the core. Facebook uses an auto-tagging algorithm based on neural networks, Google's photo search, Amazon's product recommendations, and Instagram's search infrastructure.

Generally, the basic structure of a CNN includes two layers, one of which is a feature extraction layer. The input of each neuron is connected to the local receptive field of the previous layer, and the local features are extracted. After the local feature is extracted, the

positional relationship between it and other features is also determined. The second is a feature mapping layer. Each computing layer of the network consists of multiple feature maps, and each feature map is a plane. All neurons in the plane have equal weight. The feature map structure uses the sigmoid function with a small influence function kernel as the activation function of the convolution network, so that the feature map has displacement invariance. In addition, because the neurons on a mapping surface share weight, the number of free parameters of the network is reduced. Each convolutional layer in a convolutional neural network is followed by a calculation layer used to find local averages and secondary extractions. This unique feature extraction structure reduces feature resolution.

CNN is mainly used to identify two-dimensional graphics with the invariance of displacement, scaling, and other forms of distortion. This part of the function is mainly implemented by the pooling layer. Since the CNN feature detection layer learns from the training data, when using CNN, explicit feature extraction is avoided, and learning is implicitly performed from the training data. Furthermore, because the neurons on the same feature map have the same weight, the network can learn in parallel, which is also a great advantage of the convolutional network over the network of neurons connected to each other. Convolutional neural networks have unique advantages in speech recognition and image processing due to their special structure of local weight sharing. Its layout is closer to the actual biological neural network, and the weight sharing reduces the complexity of the network. In particular, the feature that multi-dimensional input vectors can be directly input to the network avoids the complexity of data reconstruction during feature extraction and classification.

7.2.1/ Convolution

It is more intuitive when seeing a small block randomly selected from a large image. For example, 8x8 is used as a sample, and some features are learned from this small sample. At this time, we can use the features learned from this 8x8 sample as a detector and apply it to any place in this image. In particular, we can use the features learned from the 8x8 sample to convolve with the original large-size image, so as to obtain an activation value for a different feature at any position on the large-size image.

As shown in the figure below, we show the process of a 3 * 3 convolution kernel doing convolution on a 5 * 5 image. Each convolution is a feature extraction method. It is like a sieve, filtering out the eligible parts of the image.

7.2.2/ Pooling layer

A feature of the pooling layer is feature invariance, which is the scale invariance of features that we often mention in image processing. The operation of the pooling layer is to adjust the size of the image. For example: the size of an image of a person has been reduced by a factor of two, we can still recognize that this is a photo of a person. This shows that the most important features of the character are still retained in this image, and we judge that the person is drawn in the image. The information removed during pooling is only some irrelevant information, and the remaining information is a feature with scale invariance, which is also the feature that can best express the image.

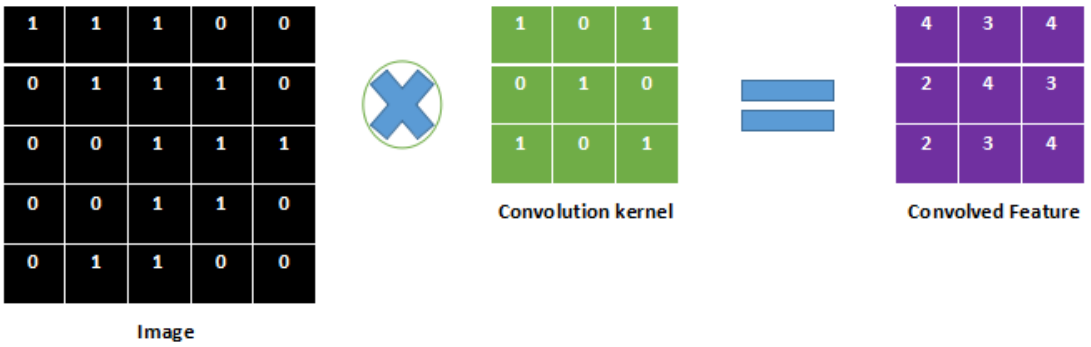

Figure 7.4 – Convolution feature example diagram.

Another feature of the pooling layer is feature dimensionality reduction. We know that an image contains a lot of information and many features, but some information is not useful or repetitive for our image tasks. We can remove this kind of information and extract the most important features, which is also a big role in the pooling operation.

7.2.2.1/ COMBINATION OF CONVOLUTIONAL LAYER AND POOLING LAYER

The figure below shows a typical CNN model structure.

As can be seen from the figure above, each node in the convolutional and pooling layers of the CNN is only connected to some nodes in the previous layer. The input of each node of the convolution layer is only a small block in the previous layer. The size of this small block is determined by the window size of the convolution kernel. In general, the node matrix processed by the convolution layer will become deeper, and the specific depth is determined by the number of convolution kernels in the convolution layer. The parameter of the convolution kernel is that sharing can make the content of the image independent of the position. At the same time, sharing the parameters of the convolution kernel can greatly reduce the parameters in the network model and reduce the complexity of the operation.

The input of each node of the pooling layer is also a small piece of the previous layer (usually a convolutional layer). The size of this small block is determined by the window size of the pooling kernel. The pooling layer does not change the depth of the node matrix, but it can change the size of the matrix. In general, for image processing, the pooling operation in the pooling layer can be understood as converting a high-resolution picture into a low-resolution picture. Common pooling operations include maximum pooling, average pooling, and so on. After the convolution layer and the pooling layer, the number of parameters in the network model can be further reduced. Let us analyze how CNN can reduce the parameters in the network model.

7.2.2.2/ DIMENSIONAL CHANGE PROCESS

Convolutional layers are one of the important concepts in CNN. We perform the convolution operation on the previous layer through the convolution kernel to complete the feature extraction. Here, we mainly analyze the dimensional change process of the convolutional

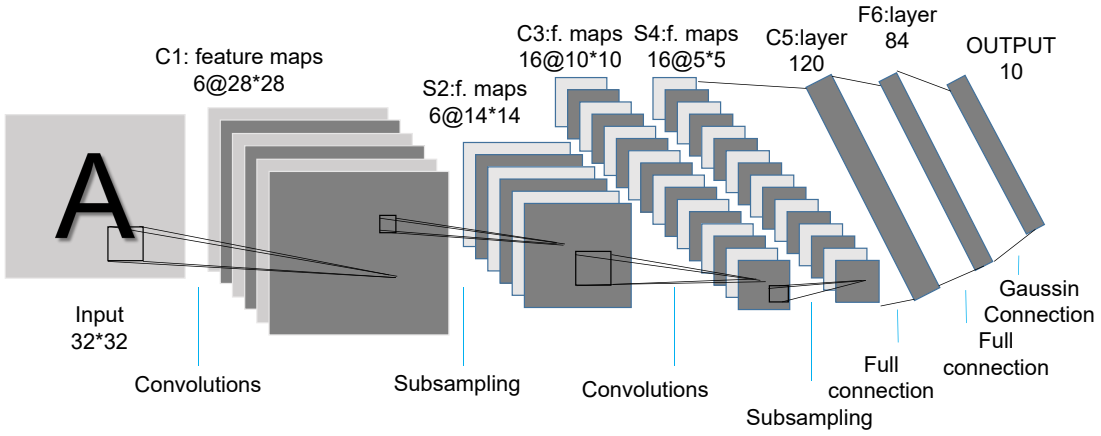

Figure 7.5 – Typical CNN model structure diagram.

layer and the pooling layer. When using all 0 padding (if the stride size is 1, the size of the node matrix can be prevented from changing after passing through the convolutional layer), the output dimension calculation formula of the convolutional layer and the pooling layer is:

$$out_{longth} = \left[\frac{in_{length}}{stride_{length}} \right] \tag{7.1}$$

$$out_{width} = \left[\frac{in_{width}}{stride_{width}} \right] \tag{7.2}$$

out (height) represents the length of the output matrix of the convolution layer. It is equal to the rounded value of the input layer matrix length divided by the stride size in the height direction. out (width) represents the width of the output matrix of the convolution layer, which is equal to the rounded value of the input layer matrix width divided by the stride size in the width direction.

If you do not use all 0 padding, the output dimension of the convolution layer and pooling layer is calculated as:

$$out_{length} = \left[\frac{in_{length} - filter_{length} + 1}{stride_{length}} \right] \tag{7.3}$$

$$out_{width} = \left[\frac{in_{width} - filter_{width} + 1}{stride_{width}} \right] \tag{7.4}$$

filter (height) represents the size of the convolution kernel/pooling kernel in the height direction, and filter (width) represents the size of the convolution kernel/pooling kernel in the width direction.

We assume that the dimensions of the input layer matrix are 32 * 32 * 1, 1 represents a grayscale image, the size of the first layer of the convolution kernel is 5 * 5, and the depth is 6 (that is, there are 6 convolution kernels), without all 0 padding and with a step size of 1. The output dimension of the convolution layer is (32-5+1=28), and the parameters of the convolution layer are 5 * 5 * 1 * 6 + 6 = 156. Here we can find that the number of

parameters of the convolution layer has nothing to do with the size of the picture, it is only related to the size of the convolution kernel, the depth, and the depth of the node matrix of the current layer. This allows CNN to be extended to image data of any size. The output dimension of the convolution layer is the input dimension of the next layer (usually the pooling layer) with 28 * 28 * 6 = 4704 nodes. The pooling kernel size of the pooling layer is 2 * 2, the step size is 2, the output dimension of the pooling layer is (28-2+1)/2=13.5, and the rounding result is 14. The pooling layer does not affect the depth of the node matrix, so the output dimension of the pooling layer is 14 * 14 * 6.

7.2.3/ Activation function ReLU (Rectified Linear Units)

The role of the activation function is to add non-linear factors and make the output of the convolution layer a non-linear mapping. ReLU is common in convolutional layers.

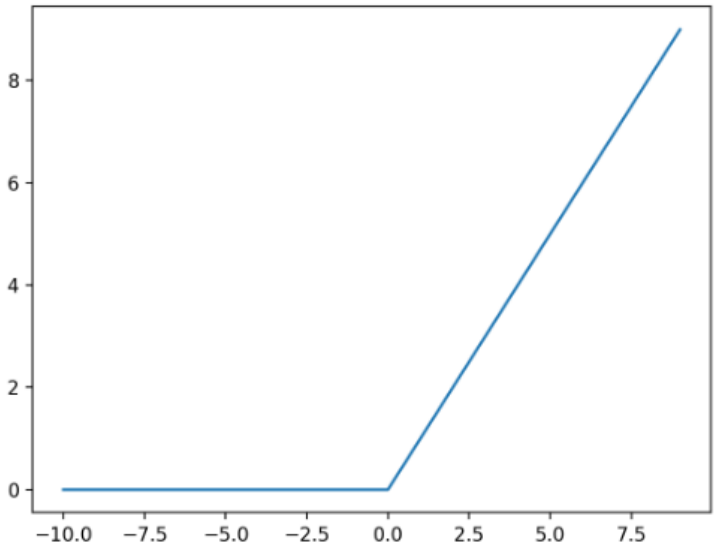

Figure 7.6 – Activation function Relu.

In convolutional neural networks, the activation function generally uses ReLU, which is characterized by fast convergence and simple gradient calculation. The calculation formula is also very simple, max (0, T), that is, if the input is negative, the output is then all 0; if the input is positive or 0, the output remains.

7.2.4/ Fully connected layer

A fully connected layer in a convolutional neural network is equivalent to a hidden layer in a traditional feedforward neural network. The fully connected layer is located at the last part of the hidden layer of the convolutional neural network, and only transmits signals to other fully connected layers. Feature maps lose their spatial topology in fully connected layers; instead, they are expanded into vectors and pass activation functions.

The convolutional layer and pooling layer in the convolutional neural network can perform feature extraction on the input data. The role of the fully connected layer is to perform a nonlinear combination of the extracted features to obtain the output, that is, the fully

connected layer itself is not expected to have feature extraction capabilities, but tries to use the existing high-order features to complete the learning goal.

In some convolutional neural networks, the function of the fully connected layer can be replaced by global mean pooling. Global mean pooling averages all values of each channel in the feature map. That is, if there is a 7 * 7 * 256 feature map, the global mean pooling will return a 256 vector. Then each of these elements is 7 * 7, the step size is 7, and the mean is pooled without padding.

7.2.5/ LOCAL RECEPTIVE FIELD

If a classic neural network model is used, we need to read the entire image as the input to the neural network model (that is, the fully connected way). When the size of the image is larger, its connected parameters will become more, which results in a very large amount of calculation.

Convolutional neural networks have two modes to reduce the number of parameters. The first mode is called a local receptive field. It is generally believed that people's perception of the outside world is from local to global, and the spatial relationship of the image is also close to the local pixels, while the pixels with longer distances are weaker. Therefore, it is not necessary for each neuron to perceive the global image, only the local area. Then at a higher level, the local information is integrated to obtain the global information. The idea that the network is connected is also inspired by the visual system structure in biology. Neurons in the visual cortex receive information locally (that is, these neurons respond to stimuli in only certain areas), as shown in the following figure:

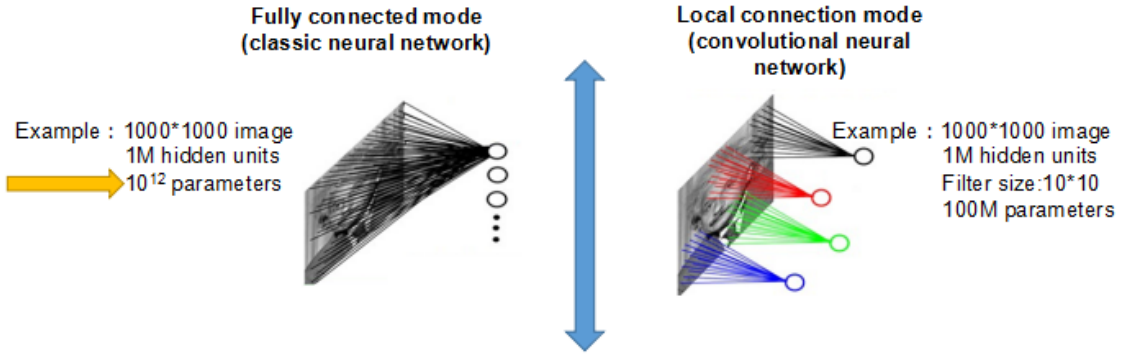

Figure 7.7 – The left picture is full connection, and the right picture is partial connection.

In the upper right figure, if each neuron is only connected to 10*10 pixel values, the weight data is 1000000*100 parameters, which is reduced to one ten thousandth of the original. The 10*10 parameters corresponding to the 10*10 pixel values are actually equivalent to the convolution operation.

7.2.6/ MULTI-CONVOLUTION KERNEL

When there are only 100 parameters mentioned above, it means that there is only one 10 * 10 convolution kernel. Obviously, feature extraction is insufficient. Then we can add multiple convolution kernels, such as 32 convolution kernels, which can learn 32 features.

Each convolution kernel generates an image as the result of applying a particular pattern template. For example, two convolution kernels can generate two images, and these two images can be regarded as different channels of an image.

7.2.7/ MULTIPLE CONVOLUTIONAL LAYERS

In practical applications, multi-layer convolution is often used, followed by fully connected layers. The purpose of multi-layer convolution is that the features learned by one layer of convolution are often local. The higher number of layers, the more global the learned features.

Convolutional networks are essentially an input-to-output mapping that can learn a large number of mappings between inputs and outputs. It does not require any precise mathematical expression between input and output. As long as the convolutional network is trained with a known pattern, the network has the ability to map between input and output.

A very important feature on CNN is that the more inputs the smaller the weight, the more outputs the larger the weight. It takes the form of an inverted triangle. This is a good way to avoid gradient loss too quickly during back propagation.

7.3/ CONVOLUTIONAL NEURAL NETWORK APPLIED TO COMPUTER VISION

With the progress of global urbanization, intelligent monitoring has gradually become a research focus in the field of computer vision. Crowd counting is one of the core problems of intelligent monitoring. It is of great significance in application scenarios such as crowd flow restriction and drainage. Great progress has been made in the research of population counting. However, in different scenarios, the research on solving the problem of inconsistent scales of crowd images still has great challenges. In recent years, deep convolutional neural networks have achieved outstanding results in the field of computer vision research. It has outstanding performance in image feature extraction and model generalization, and effectively solves the problem of feature extraction for crowd counting under complex backgrounds. In order to extract scale-related features [87], current crowd counting neural network models have adopted a multi-column or multi-network structure.

7.3.1/ CROWD COUNTING BASED ON CONVOLUTIONAL NEURAL NETWORK

The crowd counting task is an important research problem. Now more and more people are concerned about safety issues. When the population density reaches a very high peak, the population density counts, the alarm is sent out, and the crowds are diverted.

7.3.1.1/ MCNN METHOD

In 2016, Zhang et al. Proposed a single-image crowd counting method using a multi-column convolutional neural network [90].

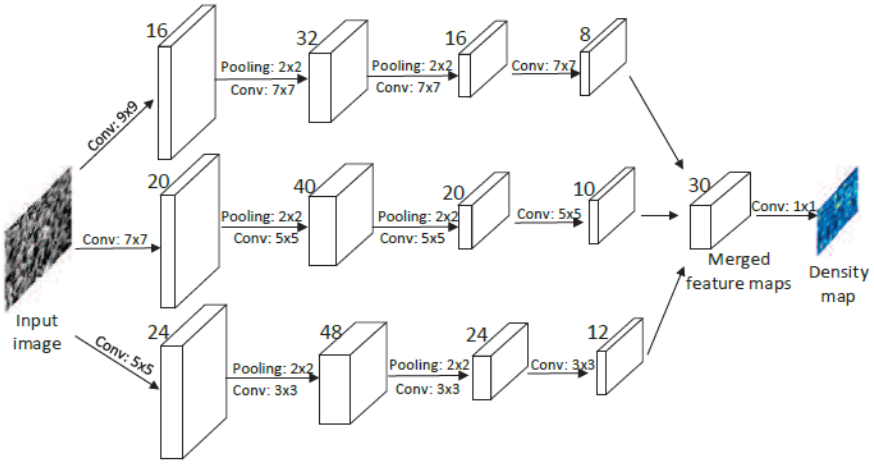

Figure 7.8 – Multi-column convolutional neural network framework [90].

This model estimates the number of people without segmenting the foreground. It needs to comprehensively utilize features of different scales to accurately estimate the number of people in different images, and the model can automatically learn effective features. Here, why is multi-coumn convolution used? The reason is that due to perspective distortion, pictures often contain human heads of different sizes, and only one kind of convolution kernel is not enough. So multiple columns of different convolutions are proposed to accommodate multiple sizes. The network uses a convolutional layer to turn the multi-channel tensor into a plane, replacing the original fully connected layer. In this way, the size of the input picture can be arbitrarily selected, which makes the network model most widely used.

- ***Conclusion:*** Zhang et al proposed a Multi-column Convolution Neural Network which can estimate crowd number accurately in a single image from almost any perspective. To better evaluate performances of crowd counting methods under practical conditions, they have collected and labelled a new dataset named Shanghaitech which consists of two parts with a total of 330,165 people annotated. The three major contributions of the MCNN model. (1) Created a new ShanghaiTech dataset. (2) A multi-column full convolutional network is proposed to count and the performance is improved. (3) Use 1x1 convolution kernel instead of a fully connected layer, so that the input image size can be any size.

7.3.1.2/ CNNs METHOD

In 2017, Vishwanath A. Sindagi and Vishal M. Patel proposed a cascade multi-task learning method based on CNN's advanced prior and density estimation [66] [76]. By merging a high-level prior with the network [30] [89], Sam learn a model that meets various density levels in the data set. The high-level prior is divided into different labeled groups according to the number of people in the picture [67]. Then, using tags, through this high-level prior, the number of people in the entire picture can be roughly estimated without being affected by scale changes [54]. As a result, the network can learn more discriminating global features. The high-level prior and CNN network [25] are used to estimate the density map together.

So here, this network inputs an image of task size and outputs a crowd density map. The network consists of two parts, the first part learns high-level prior, and the second part estimates density maps. The first part contains a set of convolutional layers, a pyramid-shaped pooling layer, and a fully connected layer. The second part consists of a set of convolutional layers, followed by a small-step convolutional layer, which is used to up-sample the output of the previous layer to compensate for the loss of detail caused by the early pooling layer.

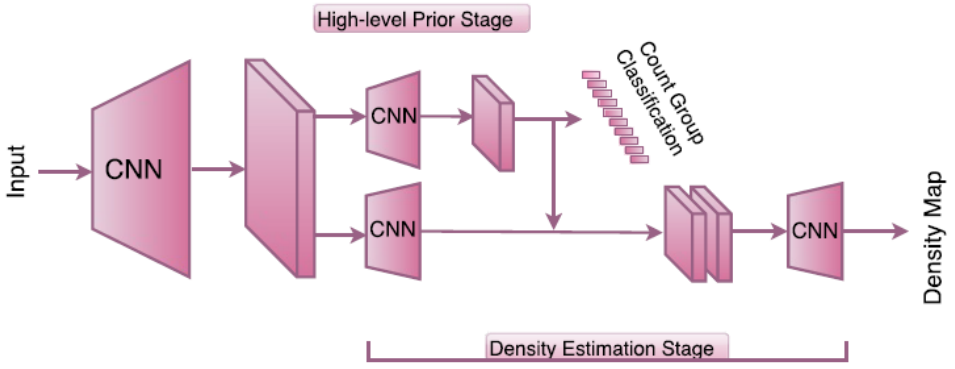

Figure 7.9 – Cascaded architecture for learning high-level prior and density estimation [66].

- ***High-level prior part:***

Dividing the population into groups is much easier than classifying or regressing the entire image directly [74] [86], because not much training data are needed. Therefore, here we quantify the crowd count into 10 groups and learn a crowd count group classifier. The classifier also performs the task of incorporating high-level priors into the network. The high-level prior part accepts the feature maps obtained previously as input. This part contains 4 convolutional layers, each of which uses PReLU as the activation function. After starting the two convolutional layers, there is a maximum pooling layer with a step size of 2. Finally, it includes 3 fully connected layers, while the activation function is still PReLU, and the number of neurons respectively is 512, 256, and 10. To be able to train with images of any size, Spatial Pyramid Pool (SPP) is used because it eliminates the fixed size constraints of deep networks containing fully connected layers. The SPP layer combines the features of the convolutional layer to produce a fixed-size output and can provide it to a fully connected layer. The cross-entropy error serves as a loss layer at this stage.

The cross-entropy loss function of the high-level prior part is:

$$L_c = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M}\left[\left(y^i = j\right) F_c\left(X_i, \theta\right)\right] \tag{7.5}$$

N represents the number of training examples, θ is a set of network parameters, Xi is the ith training example, $F_C\ (X_i,\ \theta)$ represents the classification of the output, y_i is the groundtruth classification, and M is the number of categories.

- ***Density estimation:***

This part of the network still contains 4 convolutional layers. After each layer there is a PReLU function as the activation function. After the first two layers, there is a maximum pooling layer with a step size of 2. The first convolution layer is 7*7, 20 channels. The second convolution layer is 5*5, 40 channels. The third is 5*5, 20 channels. The fourth layer is 5*5 with 10 channels. The output of this network is combined with the output of the high-level prior through two convolutional layers and two small-step convolutional layers. The first two convolutional layers are 3*3, 24 and 32 channels, and the small step convolutional layers are 16 and 18 channels. In addition to integrating the priors, these small-step convolutional layers can also promote feature maps to the original input size [5]. The reason for this is to recover details that were lost by the largest pooling layer before. The use of these layers has increased the upsampling rate of CNN output by 4 times. This enables regression on a full-resolution density map. Standard Euclidean loss is used as the loss layer.

The density estimation loss function is:

$$L_d = \frac{1}{N}\sum_{i=1}^{N} \|F_d\left(X_i, C_i, \Theta\right) - D_i\|_2 \tag{7.6}$$

F_d is the estimated density map, D_i is the true density map, and C_i is the feature map obtained from the last convolutional layer at the advanced prior stage. Then the total loss function is:

$$L = \lambda L_c + L_d \tag{7.7}$$

- ***Conclusion:*** Vishwanath A. Sindagi and Vishal M. Patel proposed a multi-task cascaded CNN network for jointly learning crowd count classification and density map estimation. By learning to classify the crowd count into various groups, they are able to incorporate a highlevel prior into the network which enables it to learn globally relevant discriminative features thereby accounting for large count variations in the dataset. Additionally, they employed fractionally strided convolutional layers at the end so as to account for the loss of details due to max-pooling layers in the earlier stages there by allowing them to regress on full resolution density maps. The entire cascade was trained in an end-to-end fashion.

7.3.1.3/ SWITCH-CNN NETWORK ARCHITECTURE

In the same year, D.B Sam and others proposed switch convolutional neural networks [62]. The network uses changes in crowd density in the image to improve the accuracy and localization of predicted crowd counts. First, several CNNs with different convolution kernel sizes are used as regression maps for density map prediction, and then a trained selection classifier is used to select the optimal CNN regressor for each input image, and the results are used as the final results.

The Switch-CNN network structure includes three CNN regressors with different structures and a classifier that selects the optimal regressor. For each input picture, first cut it into 9 parts that do not overlap each other, and each part is 1/3 length and width of the original image. The purpose is to make the input small picture be regarded as a single unit of density, scale, and perspective as the smallest unit for the selective regressor.

The CNN regressor selects the network structure in MCNN. Each column includes 4 convolutional layers and 2 pooling layers. The convolution kernels in the three columns have different sizes.

The classifier uses a VGG-16-based structure, removing the last fully-connected layer, replacing it with a global average pooling layer (GAP), two smaller fully-connected layers, and a softmax classification layer. The result of the softmax layer is the selected regressor. The procedure is as follows:

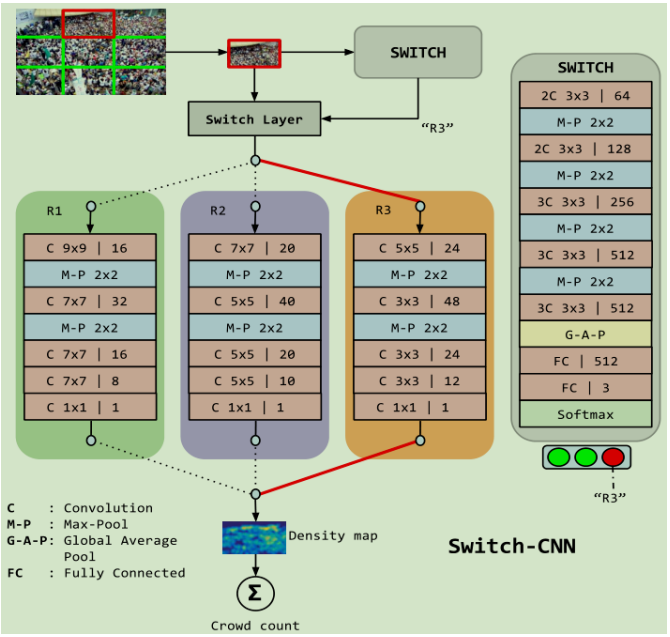

Figure 7.10 – Swith-CNN architecture [62].

Regarding the use of true density maps, the labeled images in the data set are represented in the form of points, so a method is needed to convert the labeled points into a density image. Here they cite the method of geometric adaptive kernel in MCNN network architecture by setting the parameters of the Gaussian transformation as the average distance from the labeled points to the k nearest neighbors. This method can better simulate the perspective information. For denser populations, a geometrically adapted Gaussian kernel is used, while for sparse populations a fixed Gaussian transform is used.

Switch-CNN training is divided into three steps: pre-training, differential training, and integrated training. In the pre-training phase, the three CNN regressors are first trained separately, and the loss function is the Euclidean distance between the real density image and the predicted image. This is followed by the difference training phase, where the results obtained from each of the three regressors trained on each training picture are evaluated. D.B Sam select the regressor with the smallest total number of errors to train the image again [62]. The effect of this process is that all training pictures are divided into three groups, and each group is fitted with a network. In this way, when the obtained test picture is correctly classified, a more accurate density image can be calculated by a regressor that best matches the picture characteristics. In order to improve the training effect, the process of differential training and training of the classifier is alternated.

- ***Conclusion:*** D.B Sam et al proposed a novel crowd counting model that maps a given crowd scene to its density. Crowd analysis is com-pounded by myriad of

factors like inter-occlusion between people due to extreme crowding, high similarity of appear-ance between people and background elements, and large variability of camera view-points. Current state-of-the art approaches tackle these factors by using multi-scale CNN architectures, recurrent networks and late fusion of features from multi-column CNN with different receptive fields. They propose switching convolutional neural network that lever-ages variation of crowd density within an image to improve the accuracy and localization of the predicted crowd count. Patches from a grid within a crowd scene are relayed to independent CNN regressors based on crowd count predic-tion quality of the CNN established during training. The independent CNN regressors are designed to have different receptive fields and a switch classifier is trained to relay the crowd scene patch to the best CNN regressor. The Switch-CNN network architecture solves the mutual occlusion among people, the high similarity the high similarity in appearance among people and background elements caused by extremely crowded environments.

7.3.1.4/ CSRNET ARCHITECTURE

In 2018, Li et al. Proposed a network for crowded scene recognition called CSRNet. A way to understand highly crowded scenarios by providing data-driven deep learning. This allows accurate count estimates and constructs high-quality density maps [42].

The CSRnet network model is mainly divided into front-end and back-end networks. Here, they use VGG-16, which removes the fully connected layer as the front-end network of the CSRnet to extract the features of the image. The size of the output density map is 1/8 of the original input image. A hollow convolutional neural network is used as the back-end network to expand the receptive field while maintaining the resolution to generate a high-quality crowd distribution map. Its network structure is as follows figure 7.11:

Configurations of CSRNet			
A	B	C	D
input(unfixed-resolution color image)			
front-end (fine-tuned from VGG-16)			
conv3-64-1 conv3-64-1			
max-pooling			
conv3-128-1 conv3-128-1			
max-pooling			
conv3-256-1 conv3-256-1 conv3-256-1			
max-pooling			
conv3-512-1 conv3-512-1 conv3-512-1			
back-end (four different configurations)			
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-512-1	conv3-512-2	conv3-512-2	conv3-512-4
conv3-256-1	conv3-256-2	conv3-256-4	conv3-256-4
conv3-128-1	conv3-128-2	conv3-128-4	conv3-128-4
conv3-64-1	conv3-64-2	conv3-64-4	conv3-64-4
conv1-1-1			

Figure 7.11 – CSRNet architecture [42].

Li et al. Mainly used a VGG-16 network excluding the fully connected layer, and used a 3

× 3 convolution kernel. Through experiments, it can be obtained that for the same size of the receptive field, the smaller the convolution kernel, the more the number of convolution layers is, the better the model is. To balance accuracy and resource overhead, the VGG-16 network here uses a combination of 10 convolutional layers and 3 pooling layers. The back-end network uses six layers of hole convolution layers with the same hole ratio. Finally, a layer of 1 × 1 ordinary convolution layer is used to output the results.

- ***Conclusion:*** Li et al. Proposed a network for Congested Scene Recognition called CSRNet to provide a data-driven and deep learning method that can understand highly congested scenes and perform accurate count estimation as well as present high quality density maps. The proposed CSRNet is composed of two major components: a convolutional neural network (CNN) as the front-end for 2D feature extraction and a dilated CNN for the back-end, which uses dilated kernels to deliver larger reception fields and to replace pooling operations. CSRNet is an easy-trained model because of its pure convolutional structure. They demonstrate CSRNet on four datasets (ShanghaiTech dataset, the UCF CC 50 dataset, the WorldEXPO'10 dataset, and the UCSD dataset). In the ShanghaiTechPart_B dataset, CSRNet achieves 47.3% lower Mean Absolute Error (MAE) than the previous state-of-theart method.

7.3.1.5/ SFCN NETWORK STRUCTURE

In 2019, Junyu Gao et al. Proposed two performance methods that use synthetic data to improve population counting [77]. First, a supervising strategy is used to reduce overfitting. Specifically, a large-scale synthetic data are first used to pre-train a crowd counter, which is a spatial full convolutional network (SFCN). The actual counter was then used to correct the counter. This method can effectively improve the performance of actual data. Some layers in traditional models have random initialization or regular distribution. Compared with them, the SFCN network structure can provide more complete and better initialization parameters.

Secondly, Junyu Gao et al. Proposed a domain adaptive crowd counting method which improves cross-domain transfer capabilities. Through a SSIM embedded (SE) loop GAN, it can effectively transform the synthesized crowd scene into a real scene. During the training process, they introduced the Structural Similarity Index (SSIM) loss. This is a loss between the original image and the image reconstructed by the two generators. Compared with the original circular GAN, this method effectively retains the local pattern and texture information, especially in crowded areas and certain backgrounds. Finally, the synthesized data are converted into a realistic image. Based on this data, they trained a crowd counter without real data labels, which works well in the wild. The following figure 7.12 shows two flow charts of the proposed SFCN network:

- ***Conclusion:*** The three major contributions of the SFCN model. (1) This is the first data collector and tagger to manually develop crowd counting. It can automatically collect and annotate images without any labor costs. By using them, the first large-scale, comprehensive, and diverse population counting data set was created. (2) A new pre-training method is proposed to improve the performance of the original method of real data. At the same time, compared with the random initialization and ImageNet model, it can reduce the estimation error more effectively. (3) A crowd

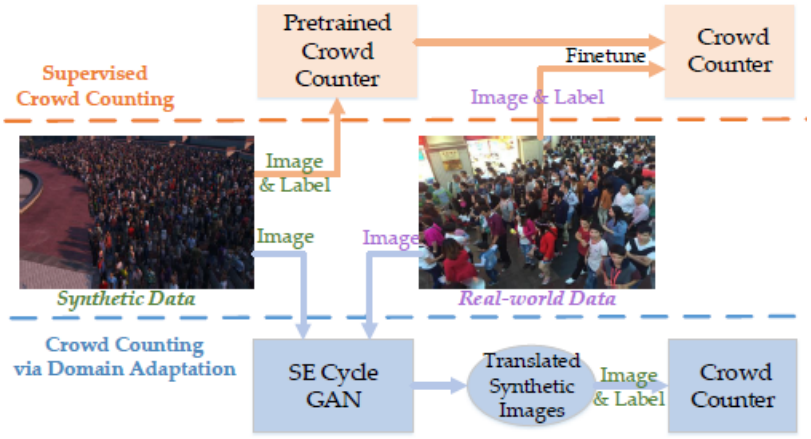

Figure 7.12 – SFCN network structure [77].

counting method based on adaptive field. This method does not use any labels for real data. By designing the SE loop GAN, the domain gap between synthetic data and actual data can be significantly reduced.

7.3.2/ MULTI-FEATURE COUNTING OF DENSE CROWD IMAGE BASED ON MULTI-COLUMN CONVOLUTIONAL NEURAL NETWORK

Here, in order to effectively solve the problem of extracting scale-related features in crowd counting, we propose a new framework M-MCNN based on MCNN for crowd counting on any single image. M-MCNN not only contains the original three columns of convolutional neural networks with different filter sizes, but replaces the fully connected layers with a convolutional layer of 1*1 filters , so the input image of the model can be of any size to avoid distortion. Moreover, in a single individual sample, we greatly improve the learning of sample features by extracting the texture features of a single human head , and better use it for datasets.

Given an image, we use M-MCNN to estimate the number of people. Here we use the population density map of the output (how many people per square meter), and then we use the total number of points.

The main reason is that the density map retains more information. The density map gives the spatial distribution of the population in a given image compared to the total number of people, and such distribution information is useful in many applications. For example, if the density of a small area is much higher than the density of other areas, it may indicate that some anomalies have occurred there, promptly alerting and evacuating the crowd.

When passing through M-MCNN model density map, the filter is more adaptable to the heads of different sizes and is therefore more suitable for any input with significant changes in perspective. These filters have more semantics and also improve the accuracy of crowd counting.

M-MCNN was mainly inspired by the success of MCNN in image classification. Each column of the M-MCNN network has the same depth of parallel sub-networks, but the filters are different in size (large, medium, and small) and can capture the characteristics of different sizes of human heads. Here we also incorporate texture features and target

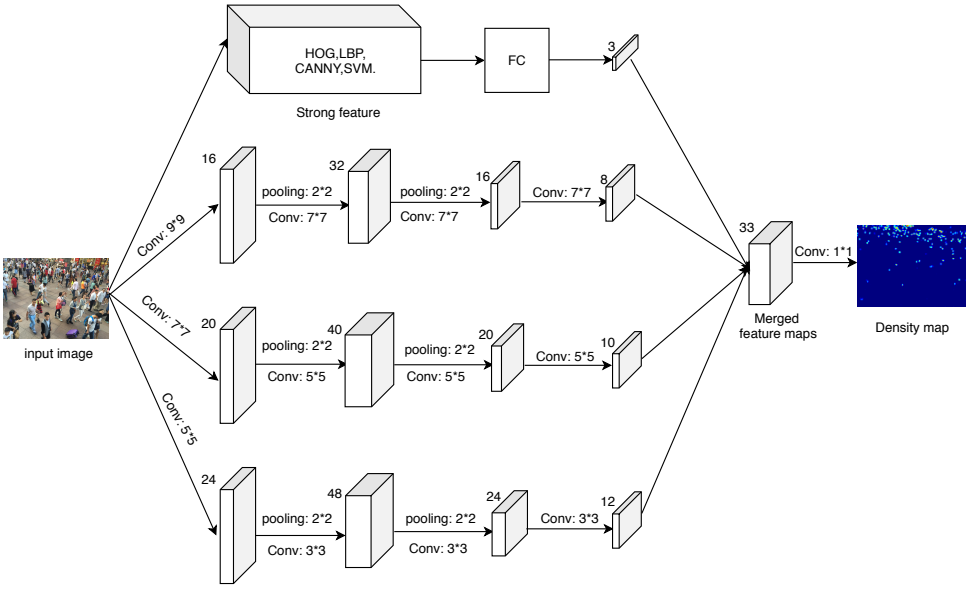

Figure 7.13 – The structure of the proposed multi-feature multi-column convolutional neural network for density map estimation.

edge detection to further refine the head features. Finally, the feature maps of the three columns of subnetworks are linearly weighted (completed by 1x1 convolution) to obtain the population density map of the image, following the concept of feature fusion.

7.3.2.1/ TEXTURE FEATURE AND TARGET EDGE DETECTION

- ***First part*** : Feature extraction.

The number of extracted local HOG features depends on our definition of the image size of each person, whose feature image can extract hundreds of local features through dense sampling. HOG performs the same processing on each element of the person's feature image. The directional gradient histogram is actually a feature vector. Next, we will use these HOG features to train the classifier. For the image of the head size and direction of the crowd, further processing is performed to summarize the set of local features into concise feature vectors to describe the characteristics of the crowd.

The LBP operator is calculated pixel by pixel.Here we perform pixel extraction of different sizes for each person's head feature , compare it to neighboring pixels and then follow neighboring pixels clockwise or counterclockwise. If other values are greater than the center pixel, we record "1". If not, we record "0". When passing through all the neighbors around, we can get 8-bit binary numbers. We convert it to decimal. The histogram is then normalized. Finally, the obtained statistical histograms of each region are connected into a feature vector.

Through Chapter 6, we put forward a kind of improved algorithm based on Canny algorithm, that is, first of all with the edge of the keep combined with filtering and anisotropic partial differential equation smoothing filter adaptive edge preserving filtering instead of the traditional Canny algorithm of Gaussian filtering method, which can for subsequent

Canny algorithm to detect edge linking to do a good job in basic, of image is good to the effect of noise reduction while preserving image edges.

Research on the human visual system has shown that image boundaries are particularly important, and that objects can often be identified by simply their outlines. This fact provides important insights for machine vision research, that is, the objects available the boundary represented by the grayscale image discontinuous points consisting of basic original carrying the original image of the most useful information. Here, we obtain clear head contours by edge detection, and fuse LBP and HOG texture feature analysis, thereby getting more head feature details.

7.3.2.2/ STRONG FEATURES AND DENSITY MAPS

- ***Second part*** : Output of strong features and density maps.

We use the method of HOG, LBP, CANNY, SVM feature fusion to get the head features. Here we define it as a strong feature. Then, through a fully connected layer, the output is a feature map of 3 channels. The purpose is also to prepare for the construction of M-MCNN network.

7.3.3/ M-MCNN NETWORK MODEL CONSTRUCTION

Here, we will integrate the training results of HOG, LBP, CANNY and SVM classifiers into the MCNN model to construct a new M-MCNN model. First, we input the image and add our fusion feature extraction and texture feature analysis. Then we make a fully connected layers, in order to make the network learn the importance of fusion features. The output is merged with MCNN's 3-column subnetwork, each parallel sub-network of MCNN can capture the characteristics of different sizes of human heads. A newly added fusion feature (texture feature and edge feature) can be regarded as a strong feature-assisted network learning with manual extraction. Finally, the feature map of the four-column sub-network is linearly weighted (completed by 1x1 convolution) to obtain the population density map of the image.

7.3.3.1/ CROWD COUNT BASED ON DENSITY MAP

Multi-feature fusion convolutional neural network (M-MCNN) works by estimating the number of people in a given image. It has two configurations. One is that the input is a picture and the output is an estimated number of people. In fact, in this part, we did a lot of algorithm deduction and experiments in the multi-feature fusion part of Chapter 6 of this article. Then, the other is to output the density map of the population, and then obtain the number of people through integration. Here, we choose the output as a population density map. There are several reasons for this:

The first reason is that the density map retains more information. By comparing the total number of people, the density map can give the spatial distribution of the people in a given image. Such distribution information is useful in many applications. For example, if the density of a small area is much higher than the density of other areas, it may indicate that anomalies have occurred there.

For the second reason, we obtained strong features of the human head through multi-feature fusion technology. Then through the fully connected layer, a feature map of 3 channels is outputted. This part not only allows us to get all the features of a single person's head image, but also greatly improves the resolution. It is more helpful to integrate MCNN network structure.

The third reason is in learning the density map via an M-MCNN, the learned filters are more adapted to the heads of different sizes, hence more suitable for arbitrary inputs whose perspective effect varies significantly. Thus the filters are more semantic meaningful, and consequently improves the accuracy of crowd counting.

7.3.3.2/ DENSITY MAP VIA GEOMETRY ADAPTIVE KERNELS

To generate a density map, we use a pulse function convolution Gaussian kernel to define the density map. Here, we briefly introduce the impulse function and Gaussian kernel function.

- ***Impulse function*** :

The impulse function was proposed by British physicist Dirac in the 1920s and was used to describe a physical quantity in a moment or a geometric point in space [63]. For example, instantaneous impact forces, pulse currents or voltages, and other rapidly changing physical quantities, as well as mass distributions of point masses, point-to-point radial charge distributions, and other physical quantities that are highly concentrated in space or time. The impulse function is also called δ function. If placed in a dimensional space, the independent variable is a function $\delta(t)$ of time t, which satisfies two conditions [63]:

$$\delta(t) = \begin{cases} +\infty, t = 0, \\ 0, t \neq 0; \end{cases} \tag{7.8}$$

$$\int_{-\infty}^{+\infty} \delta(t)\, dt = 1. \tag{7.9}$$

Then, a function that satisfies the above two conditions is called δ function, and it is written as $\delta(t)$. The δ function is a generalized function and can also be extended to multi-dimensional space. This is also one of the main reasons why we choose the pulse function. Its exact meaning should be understood under the integral operation: its integration curve height is "infinite height" and its width is "infinite narrow". The area under the curve is equal to 1. Therefore, the δ function has the following relationship:

$$\int_a^b \delta(t)\, dt = \begin{cases} 1, & a < 0 < b, \\ 0, & other. \end{cases} \tag{7.10}$$

Several basic properties of the δ function are given directly below, and they are also used in our subsequent calculations.

The first is the screening property. Let f (t) be a bounded function defined on the real number domain and continuous at t_0, then [63]:

$$\int_{-\infty}^{+\infty} \delta(t - t_0) f(t) dt = f(t_0) \tag{7.11}$$

In particular, when t_0 = 0, then:

$$\int_{-\infty}^{+\infty} \delta(t) f(t) dt = f(0) \tag{7.12}$$

The second property: the δ function is an even function, that is,

$$\delta(-t) = \delta(t). \tag{7.13}$$

The third property, when u (t) is a unit step function, that is,

$$u(t) = \begin{cases} 1, & t > 0, \\ 0, & t < 0. \end{cases} \tag{7.14}$$

Then:

$$\int_{-\infty}^{t} \delta(t) dt = u(t), \frac{du(t)}{dt} = \delta(t) \tag{7.15}$$

According to the screening properties of the δ function, we can get the Fourier transform of the δ function as:

$$F[\delta(t)] = \int_{-\infty}^{+\infty} \delta(t) e^{-jwt} dt = e^{-jwt}|_{t=0} = 1. \tag{7.16}$$

That is, $\delta(t)$ and F (w) = 1 constitute a Fourier transform. According to the Fourier integral formula:

$$\frac{1}{2\pi} \int_{-\infty}^{+\infty} e^{jwt} dw = \delta(t). \tag{7.17}$$

This is an important formula for the δ function. The Fourier transform of $\delta(t)$ is a generalized Fourier transform. In engineering technology, there are many functions that do not meet the conditions of absolute integrability, such as symbolic functions, unit step functions, and sine and cosine functions. However, using the δ function can find their Fourier transform. From this perspective, we can also see the importance of the δ function.

- ***Gaussian kernel*** :

A Gaussian kernel function, in simple terms, is used to map finite-dimensional data to high-dimensional space. It is usually defined as a monotonic function of the Euclidean distance between any point x in space and some center point x'. Here, we can write it as $k(||x - x'||)$, and its effect is often local. When x is larger, the value of the function is small [90].

Defined as:

$$k\left(x, x^{'}\right) = e^{-\frac{\|x-x^{'}\|^2}{2\sigma^2}} \tag{7.18}$$

$x^{'}$ is the center of the kernel function, and $\|x - x^{'}\|$ is the Euclidean distance between the vectors x and $x^{'}$. As the distance between the two vectors increases, the Gaussian kernel function decreases monotonically. σ controls the range of the Gaussian kernel function. The larger its value, the larger the local influence range of the Gaussian kernel function. Of course, σ cannot be too small, otherwise it is easy to overfit in the classification task.

Gaussian kernel functions have several typical characteristics. These properties make it particularly useful in image processing. These characteristics show that the Gaussian smoothing filter is a very effective low-pass filter in both the spatial and frequency domains, and has been effectively used by engineers in practical image processing. These qualities are:

First, the two-dimensional Gaussian function has rotational symmetry, that is, the degree of smoothness of the filter in all directions is the same. Generally, we don't know the direction of the edges of an image at first. Therefore, it is impossible to determine that one direction requires more smoothing processing than the other direction before filtering. Rotational symmetry means that the Gaussian smoothing filter will not be biased in any direction in subsequent edge detection.

Second, the Gaussian kernel function is a single-valued function. This shows that the Gaussian filter replaces the pixel value of the point with the weighted average of the pixel neighborhood. Therefore, the weight of each neighborhood pixel monotonously increases or decreases with the distance between the point and the center point. This trait is important because the edges are a local feature of the image. If the smoothing operation still has a great effect on pixels that are far away from the center of the operator, the smoothing operation will distort the image.

Third, the Fourier transform spectrum of the Gaussian kernel function is a single-lobed. Images are often disturbed by high-frequency signals, including: noise and fine textures. Image features, such as image edge features, contain both low-frequency and high-frequency components. The single lobe of the Fourier transform of the Gaussian kernel function means that the smooth image is not disturbed by unwanted high-frequency signals. At the same time, most of the required signals are retained.

Fourth, the width of the Gaussian filter determines the degree of smoothing. It is characterized by the parameter σ, and the relationship between σ and smoothness is very simple. The larger σ, the wider the band of the Gaussian filter and the better the smoothness. By adjusting the smoothness parameter σ, a better effect can be obtained between image features that are too blurred and excessive smoothness caused by noise and fine texture in smoother images.

Finally, the separability of Gaussian kernel functions. Two-dimensional Gaussian function convolution can be performed in two steps. First, the image is convolved with a one-dimensional Gaussian kernel function, and then the same one-dimensional Gaussian kernel function whose convolution result is perpendicular to the direction is convolved. The results show that the amount of computation of the two-dimensional Gaussian filter increases linearly with the width of the filter template, rather than squared.

Here, we assume that the position of the label point is x_i, then a label with N heads can be expressed as:

$$H(x) = \sum_{i=1}^{N} \delta(x - x_i). \tag{7.19}$$

Here, we convolve it with a Gaussian function into a continuous function. However, this density function assumes that each x_i is independent in image space. In fact, each x_i is a sample of the crowd density on the ground in a 3D scene. Due to the effect of perspective distortion, the pixels associated with different samples xi correspond to regions of different sizes in the scene. In order to estimate the population density more accurately, we need to consider perspective transformation. We assume that the density of the crowd is uniform around a head area, and its nearest neighbor gives a reasonable estimate of the geometric deformation. In order to make the density map better correspond to images with different perspectives (head sizes of different sizes) and dense crowds, we have improved the traditional Gaussian kernel-based density map and proposed a geometrically adapted Gaussian kernel-based density map. It is expressed by the following formula [90]:

$$\begin{cases} F(x) = \sum_{i=1}^{N} \delta(x - x_i) * G_{\sigma_i(x)}, \\ \sigma_i = \beta \widetilde{d^i}. \\ \widetilde{d^i} = \frac{1}{m} \sum_{j=1}^{m} . \end{cases} \tag{7.20}$$

In fact, the density map is obtained by convolving the δ pulse function with a Gaussian function. Here, the convolution is performed before the sum.

Here x_i represents the pixel position of the human head in the image, $\delta(x - x_i)$ represents the pulse function of the human head position in the image, N represents the total number of human heads in the image, d_j^i is the average distance between the m head closest to the head of x_i and the head at that point. Usually, the size of the head is related to the distance between the centers of two adjacent people in a crowded scene. $\widetilde{d^i}$ is approximately equal to the size of a human head in a crowded situation. Through subsequent experiments, the variance of the density map generated in this way is scattered or concentrated in the Gaussian kernel at large or small positions of the human head. This can better represent the characteristics of human head size.

In Equation 7.19, the δ function is an impulse function. The integral sum value within the range is 1, and then the N heads are summed to obtain the N head labels.

For the x_i point of each head, the average value d_i of the k nearest neighbor distances is given. Then, the pixel related to x_i corresponds to a region on the ground in the scene. The radius of this area is proportional to d_i. Therefore, in order to estimate the crowd density around the pixel x_i, we need to convolve $H(x)$ with an adaptive Gaussian kernel. The variance σ_i of this Gaussian kernel is variable and proportional to d_i.

Here, we convolve the label H with a Gaussian kernel function of an adaptive kernel. The variance of this Gaussian kernel function is a product of the average distances between β and x_i's K nearest neighbors. The implementation of the subsequent program is to extract x_i points from the ground truth and perform the Gaussian convolution.

In order to map the feature map to the density map, we set a filter with a size of 1 × 1. The difference between the estimated density map and the ground truth is measured by using the Euclidean distance. The loss function is defined as follows:

$$L(\Theta) = \frac{1}{2N} \sum_{i=1}^{N} \|F(X_i, \Theta - F_i)\|_2^2 . \tag{7.21}$$

Where θ is the network parameter to be optimized, and N is the total number of training images. Here, X_i is the image we need to enter, and F_i expresses the ground truth density map of image X_i. $F(X_i; \theta)$ stands for the estimated density map generated by M-MCNN architecture, which is parameterized with θ for sample X_i. Definition L expresses the loss difference between the estimated density map and the ground truth density map.

7.3.3.3/ Optimization of M-MCNN architecture

We optimize the M-MCNN structure from two aspects.

On the one hand: We optimize batch-based stochastic gradient descent and backpropagation through a loss function. We use a loss function to evaluate the degree of difference between the model's predicted and true values. In addition, the loss function is also an optimized objective function of the neural network. The process of neural network training or optimization is the process of minimizing the loss function. The smaller the loss function, the closer the predicted value of the model to the true value, the higher the accuracy of the model.

Here, we introduce the cross-entropy loss function [66]. The definition of the cross-entropy loss function is as follows :

$$L_i = -\left[y^{(i)} log \hat{y}^{(i)} + \left(1 - y^{(i)}\right) log\left(1 - \hat{y}^{(i)}\right)\right] \tag{7.22}$$

Take the binary classification problem as an example. For models such as logistic regression and neural networks, the real sample labels are [0,1], which represent negative and positive classes, respectively. Our model M-MCNN will finally pass a Sigmod function to output a probability value. This probability value reflects the probability that the prediction is positive: the larger the probability value, the more likely the sample is positive.

The expression of the Sigmod function is as follows:

$$g(s) = \frac{1}{1 + e^{-s}} \tag{7.23}$$

In Equation 7.23, s is the output of the layer above the model. The characteristics of the Sigmod function are: when s is 0, g (s) = 0.5; when $s \gg 0$, g≈1. When $s \ll 0$, g≈0. Obviously, g (s) maps the linear output of the previous stage to a numerical probability between [0,1]. Here g (s) is the model prediction output in cross entropy. The model prediction output characterizes the probability that the current sample is positive (that is, the label value is 1):

$$\hat{y} = P(y = 1 | x) \tag{7.24}$$

Therefore, the probability that the current sample is negative can be expressed as:

$$1 - \hat{y} = P(y = 0 | x) \tag{7.25}$$

We put the two situations above together:

$$P(y|x) = \hat{y}^y * (1-\hat{y})^{1-y} \tag{7.26}$$

Here we look at it from another perspective:

When the real sample label is y = 0, the first term of the above formula is 1, and the probability equation is transformed into:

$$P(y=0|x) = 1-\hat{y} \tag{7.27}$$

When the real sample label is y = 1, the second term of the above formula is 1, and the probability equation is transformed into:

$$P(y=1|x) = \hat{y} \tag{7.28}$$

In fact, the probability expression in both cases is exactly the same as before. We put the two situations together. Let's look at the probability expression after integration. What we want is that the larger the value of probability P(y|x), the better. First, we introduce the log function to P(y|x), because the log operation does not affect the monotonicity of the function itself. When P(y|x) is maximized, logP(y|x)is also the largest. As follows:

$$logP(y|x) = log\left(\hat{y}^y * (1-\hat{y})^{1-y}\right) = ylog\hat{y} + (1-y)\,log\,(1-\hat{y}) \tag{7.29}$$

Instead, we need the smaller the negative value of logP(y|x). Then we introduce the loss function, let loss=-logP(y|x), and the resulting loss function is:

$$Loss = -\left[ylog\hat{y} + (1-y)\,log\,(1-\hat{y})\right] \tag{7.30}$$

We calculate the loss function of a single sample. If we want to calculate the total loss function of N samples, we only need to combine the N losses:

$$Loss = -\sum\left[ylog\hat{y} + (1-y)\,log\,(1-\hat{y})\right] \tag{7.31}$$

In fact, we generally use the cross entropy function instead of the mean square error in classification problems. On the other hand, in regression problems, we often use mean square error (MSE) as the loss function [66], the formula is as follows:

$$loss = \frac{1}{2m}\sum_{i=1}^{m}(y_i - \hat{y}_i) \tag{7.32}$$

Because regression problems require fitting real numeric values, and the error between the predicted and actual values is measured by MSE. Then, it can be optimized by gradient descent. When we face the classification problem, we need a series of activation functions, such as: sigmod, softmax to map the predicted value to 0-1. When using MSE at this time, we need to think carefully. Because of the activation function, the gradient of the loss function with respect to the parameters becomes complicated, and the use of MSE brings difficulty to the optimization. The process is as follows:

$$\frac{\partial C}{\partial \omega} = (a - y)\,\sigma^{'}(z)\,x \tag{7.33}$$

$$\frac{\partial C}{\partial b} = (a - y)\,\sigma^{'}(z) \tag{7.34}$$

It can be seen from the above formula that the gradients of w and b are proportional to the gradient of the activation function. The larger the gradient of the activation function, the faster the resizing of w and b, and the faster the training convergence.

Recalculate the gradient: In this way, the partial derivative of the loss function with respect to the parameters no longer contains the derivative of the sigmod function. It has the difference between the value of sigmod and the actual value, which also satisfies the larger error we said before, the faster the decline. This is why in the classification problem, we use the cross entropy loss function instead of MSE!

Here we make a small summary. Because neural networks, logistic regression, etc. generally have sigmod functions as activation functions, if MSE is used as the loss function, the derivative of sigmod will appear in the derivative of the loss function with respect to the parameter to be obtained. The derivative of the sigmod function is a quadratic function about the original function. When $\sigma(z)$ is the independent variable, the derivative is $\sigma(z)(1-\sigma(z))$. This complicates the partial derivative and is not conducive to the parameter update.

When using cross-entropy derivation, due to the existence of the log function, the corresponding quadratic term will appear on the scorer. After elimination, the gradient is a linear function about y-$\hat{y}$. That is, the larger the error, the greater the magnitude of the parameter update.

It can be more intuitively understood that the difference (y-a) between the predicted value and the true value is A. When y=1 is taken as an example, then $\sigma^{'}$(z)=σ(z)(1-σ(z)) is converted into A(1-A). So the gradient (a-y) $\sigma^{'}$(z)x of the one-sample loss function is transformed into a function A^2(1-A)x about the error A. It is a cubic function about A. The larger A cannot be achieved, the larger the gradient. In contrast, the gradient of cross entropy is proportional to A. The larger the A, the larger the gradient, and the faster the parameter update. Therefore, we incorporate the cross-entropy loss function.

On the other hand: We add deep neural network training based on the M-MCNN structure. By optimizing the network structure, smaller errors can be obtained from the significantly increased depth of the neural network, improving accuracy.

When deeper networks can begin to converge, the degradation problem has been exposed. As deeper network levels increase, the relative accuracy decreases rapidly. This degradation is not caused by overfitting, but adding more layers to the appropriate depth model results in higher training errors.

First, we have to solve the problem of degradation. Here, we introduce a deep residual network [27]. If the layers behind the deep network are identity mappings, the model degenerates into a shallow network. So, what we have to solve is to learn the identity mapping function. It is however more difficult for some layers to directly fit a potential identity mapping function $H(x) = x$. This may be the reason why deep networks are difficult to train. However, if the network is designed as $H(x) = F(x) + x$, as shown in the figure 7.14 below, we can convert to learning a residual function $F(x) = H(x) - x$. As long

as $F(x) = 0$, an identity map $H(x) = x$ is formed. Then, fitting the residuals is definitely easier.

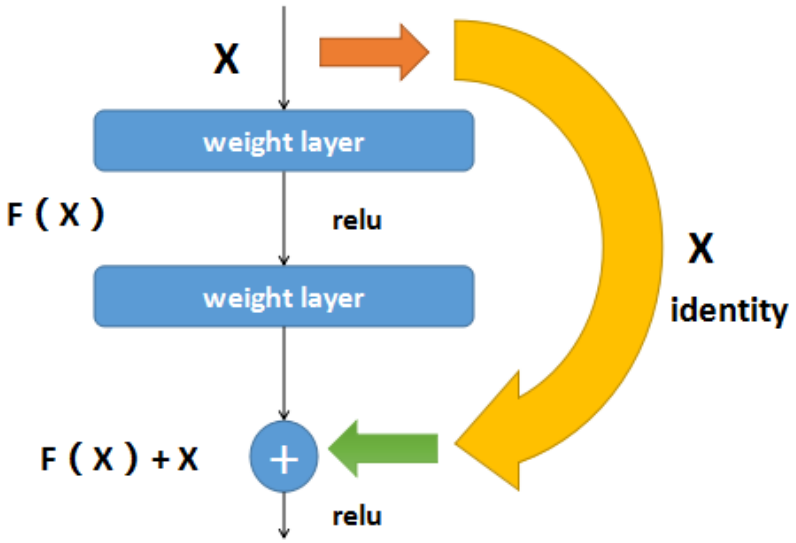

Figure 7.14 – Residual learning [27].

F is the network map before the summation, and H is the network map from the input to the summation. Here, we take an example to map 2 to 2.1, then it is $F'(2) = 2.1$ before introducing the residual. After introducing the residual is $H(2) = 2.1$, $H(2) = F(2) + 2, F(2) = 0.1$. Here F' and F both represent the network parameter mapping. The mapping after the introduction of residuals is more sensitive to changes in output. For example, the output of s changed from 2.1 to 2.2, and the output of the mapping increased by 2%. For the residual structure output from 2.1 to 2.2, the mapping F is from 0.1 to 0.2, an increase of 100%. Obviously, the output change of the latter has a greater effect on weight adjustment, and the effect is better. The idea of the residual is to remove the same main part, so as to highlight the small changes.

This residual learning structure can be implemented by a forward neural network + shortcut connection. First, the shortcut structure connection is equivalent to simply performing the equivalent mapping without generating additional parameters and without increasing the computational complexity. Second, the entire network can still be trained through port-to-port backpropagation. Here, we briefly introduce the shortcut structure.

The idea of the shortcut structure was introduced in order to solve the problem of gradient divergence and difficult training in deep networks. We know that for the original CNN model (called "plain networks", which does not specifically refer to a certain model framework), there is only a connection between two adjacent layers. As shown in the figure below, x and y are two adjacent layers, connected by W_H, and a deep network is formed by concatenating multiple such layers back and forth. The relationship between adjacent layers is as follows [27]:

$$y = H(x, W_H) \tag{7.35}$$

Where H represents the transformation in the network.

In order to solve the problem of gradient divergence of deep networks, we have added a

Figure 7.15 – A simple network structure.

shortcut structure between the two layers. The structure between the two layers is shown below:

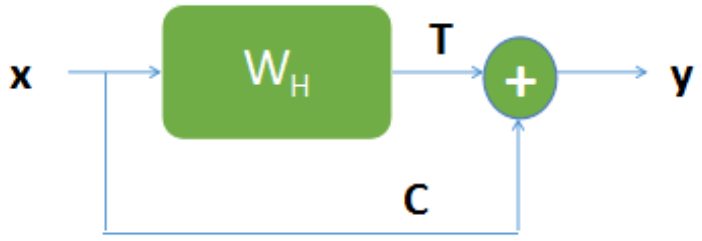

Figure 7.16 – Shortcut structure.

The relationship between x and y is as follows:

$$y = H(x, W_H) * T(x, W_T) + x * C(x, W_C) \tag{7.36}$$

Among them we set C = 1-T, which can be rewritten as:

$$y = H(x, W_H) * T(x, W_T) + x * (1 - T(x, W_T)) \tag{7.37}$$

Here, we call T a "transform gate" and C a "carry gate". The input layer x is connected to the output layer y by a weight of C. Through the improvement of this connection method, the problem of gradient divergence in deep networks is alleviated.

Our deepened residual network can be more easily optimized than a deep network produced by a simple overlay. Moreover, because of the increase in depth, the later experimental results have been significantly improved.

7.4/ EXPERIMENT

We evaluated our M-MCNN model on five different datasets compared with most MCNN-based methods in the literature, the proposed M-MCNN model is not only convenient to construct, but also integrates multiple feature source information. Above all, our proposed method has a competitive advantage and usually have superior performance when tested on the provided datasets.

We evaluate the performance via the mean absolute error (MAE) and mean square error (MSE) commonly used in previous works.

Table 7.1 – Statistics of the five real-word datasets

Dataset		Resolution	Num	Max	Min	Ave	Total
UCSD		158*238	2000	46	11	24.9	49885
WorldExpo'10		576*720	3980	253	1	50.2	199923
GCC		1080*1920	15212	3995	0	501	7625843
Shanghaitech	Part_A	different	482	3139	33	501.4	241677
	Part_B	768*1024	716	578	9	123.6	88488
CHDP		different	11585	19083	0	587	6800395

$$MAE = \frac{1}{N}\sum_{1}^{N}\left|w_i - w_i^{'}\right|. \tag{7.38}$$

$$MSE = \sqrt{\frac{1}{N}\sum_{1}^{N}\left(w_i - w_i^{'}\right)^2}. \tag{7.39}$$

where N is total number of test images, w_i is the actual number of people detected in the ith image, and w_i' is the estimated number of people in the ith image. So here, the MAE we use represents the accuracy of the estimate, and the MSE represents the robustness of the estimate.

7.4.1/ SHANGHAITECH DATASET

Here, we have introduced a large-scale population count dataset to test the assessment of crowd counting tasks. The Shanghai dataset, which contains 1,198 annotated images, totals 330, with 165 heads annotated. The dataset is mainly composed of two parts: 482 images in Part_A are randomly selected from the Internet, and 716 images in Part_B are taken from the bustling streets of Shanghai. Table 6.1 gives the statistics of Shanghaitech dataset and its comparison with other datasets.

At the same time we increase the training set of M-MCNN for training. We use the texture feature (HOG, LBP) to randomly crop each input image according to a 32×32 pixel block with BLOC. All patches are used to train our M-MCNN model. For Part_A, since the population density is very high, we use our geometric adaptive kernel to generate a density map and calculate the predicted density of the overlap region by averaging. For Part_B, because the population is relatively sparse, we use the same distribution in the Gaussian kernel to generate a density map.

We compare our method with the work of Zhang et al. they proposed a method uses MCNN for crowd counting and achieved state-of-the-art accuracy at the time. We added a fusion feature based on the original MCNN, which can be regarded as the stronger feature-assisted learning of artificial extraction. The image of the head edge is found by image target detection. Then we use the texture feature (HOG, LBP) to analyze the characteristics of each head target and statistics. Here, we divide the detected head image into small cells. Each cell size is $1 * 1$ pixels, and the gradient histogram of each cell (the number of different gradients) is counted, and each cell is composed into a block ($3*3$ cells/block). The characteristics of all cells in a block are connected in series to obtain

HOG and LBP header features. Then, we integrate the training results of the combined HOG, LBP, CANNY, and SVM classifiers into our M-MCNN model. The output is merged with MCNN's three-column sub-network, and each parallel sub-network of MCNN can capture different sizes of human head's characteristics. Finally, the feature map of the four-column sub-network is linearly weighted (completed by 1x1 convolution) to obtain the population density of the image.

In this figure, we compare the MCNN population count method on the Shanghaitech dataset.

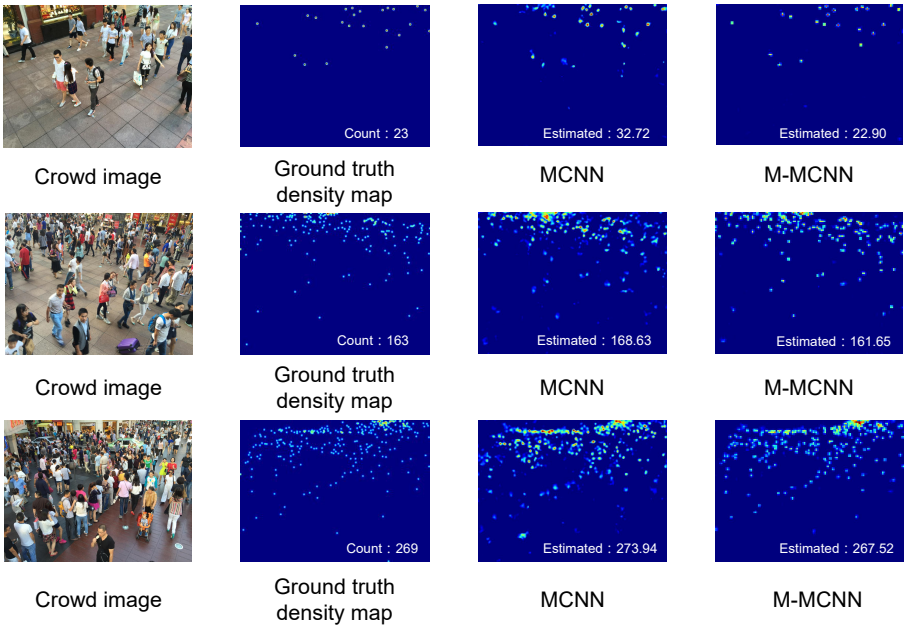

Figure 7.17 – Comparison of estimated density map of MCNN and our method of three test images in part_B.

- ***Conclusion:*** By comparing MCNN's dataset in Shanghai Part_B, we can find some details in the pictures of the people tested. For example, the first crowd test picture, this picture was taken on a pedestrian street, the actual number is 23 people. The density map obtained through the MCNN network architecture shows that the estimated number of people is 32.72, which is approximately equal to 33 people, which is 10 people different from the actual number. From the density map obtained from MCNN, it can be found that some places without people are also counted by wrong estimates. Compared with the density map obtained by our proposed M-MCNN network architecture, the estimated number of people is 22.9, which is approximately equal to 23 people, which is the same as the actual number of people. From the density map obtained from M-MCNN, we can find that the characteristics of each person counted are well preserved in the density map. Through a lot of experiments, our architecture M-MCNN better preserves the details of the generated density map. This will allow us to better count the crowd and get better results.

7.4.2/ UCSD DATASET

We also evaluated our approach on the UCSD dataset. The UCSD data set is the first dataset created for statistical population. The dataset is collected from the camera of the pedestrian channel, which consists of a frame of 238 x 158 frames of 2,000 x 158 from the video sequence and a ground truth annotation for each pedestrian in every fifth frame. For the remaining frames, linear interpolation is used to create annotations. An area of interest is also provided to ignore unnecessary moving objects, such as trees, telephone pole and so on. The dataset contains a total of 49,885 pedestrian instances and is divided into training and test sets. The training set contains frames with an index of 601 to 1400, while the test set contains the remaining 1200 images. The population of the dataset is relatively low, with an average of about 20 people in a frame.The dataset provides the ROI of each video frame. The pixel intensity outside the ROI is set to zero, and we also use the ROI to modify the final convolutional layer. Table 7.2 shows the results of our and other methods on this dataset.

Table 7.2 – Comparing results of different methods on the USCD dataset.

Method	MAE	MSE
MCNN [90]	1.07	1.35
CSRNet [42]	1.16	1.47
Swith-CNN [62]	1.62	2.10
M-MCNN (proposed)	1.02	1.23

- ***Conclusion:*** The UCSD dataset has a relatively low-density crowd, with an average of about 20 people in a frame, and because the dataset is collected from a single location, the perspective of the scene in the image has not changed. We evaluate the effectiveness of our model through MSE (Mean Squared Error) and MAE (Mean Absolute Error). MAE is the sum of absolute values of the difference between the target value and the predicted value. MSE can evaluate the degree of change of data. The smaller the value of MSE, the better the accuracy of the prediction model in describing experimental data. Here, the values of MAE and MSE that we obtained through experiments are lower than MCNN, CSRNet, Switch-CNN models. We proposed M-MCNN model is superior to foreground segmentation and background subtraction than CNN and MCNN-based methods. This indicats that our model can not only estimate a very dense population images, but also estimate relatively sparse population images.

7.4.3/ WORLDEXPO'10 DATASET

WorldExpo'10 dataset includes 3980 frames, which are from the Shanghai 2010 World-Expo. The dataset contains 1132 annotated video sequences that are captured by 108 surveillance cameras. In this dataset, 199923 pedestrians are annotated in the center of their heads. 3380 frames are used as training while the rest is taken as a test. The test set includes five different scenes , each with 120 frames. Regions of interest (ROI) are provided in each scene so that crowd counting is only conducted in the ROI in each frame. Some statistics of this dataset can be found in Table 7.3.

Table 7.3 – Mean absolute errors of the WorldExpo'10 crowd counting dataset.

Method	Sence1	Sence2	Sence3	Sence4	Sence5	Average
MCNN[[90]]	3.4	20.6	12.9	13.0	8.1	11.6
Zhang et al.[[86]]	9.8	14.1	14.3	22.2	3.7	12.9
Using our solution	3.2	19.2	11.5	11.9	7.2	10.6

- ***Conclusion:*** As demonstrated in Table 7.3, our method achieves state-of-the-art performance with respect to the average MAE of five scenes. Specifically, our model gets the lowest MAE in Scene 1, in Scene 3 and Scene 4, which are the three most challenging scenes in the testing set.

7.4.4/ GCC DATASET

Here we use the large-scale synthetic dataset GCC. GCC is a virtual game dataset. The GCC dataset contains 15,212 images with a resolution of 1080×1920 and contains 7,625,843 people. Compared to existing datasets, GCC is a larger population count dataset in terms of number of images and number of people. GCC is more diverse than other real-world datasets, GCC dataset consists of 400 different scenes, for example: convenience store, pub, mall, street, plaza, stadium and so on.

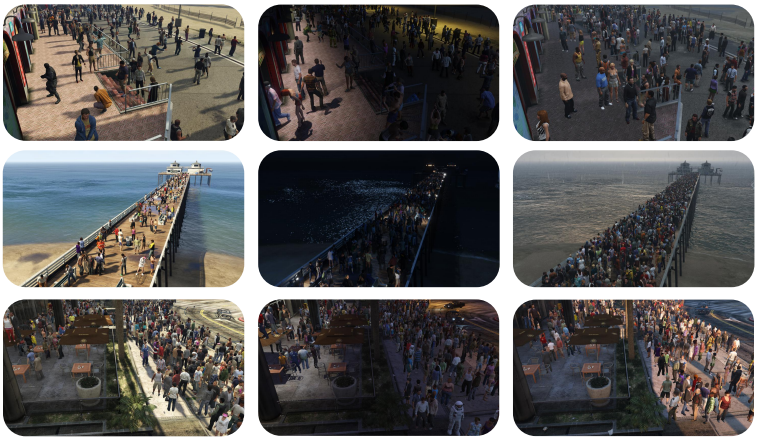

Figure 7.18 – Example for the GCC dataset

Not only that, we also added weather factors in the dataset, for exemple: clear, clouds, rain, foggy, thunder, overcast and extra sunny.

Table 7.4 reports the performance of our M-MCNN and three popular methods (MCNN [90] and CSRNet[42] and SFCN[77]) on the proposed GCC dataset.

- ***Conclusion:*** In the still crowd image, our model M-MCNN is used for testing in different environments. First, in the case of very few people (e.g., 2-3 people) then, in a very dense group, hundres and even thousands of people. Then, weather factors, such as: sunny, cloudy, rain, fog, etc. Finally, 400 different scenes, such as: streets, squares, casinos and other crowded places. Our model has achieved

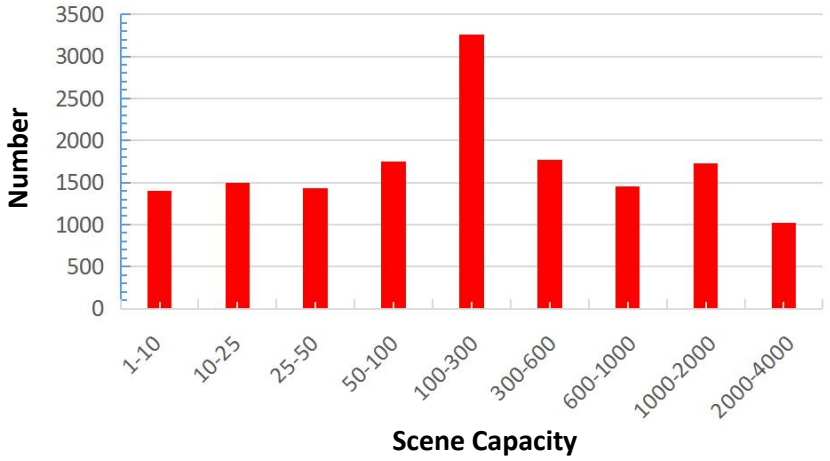

Figure 7.19 – The statistical histogram of crowd counts in different aeras.

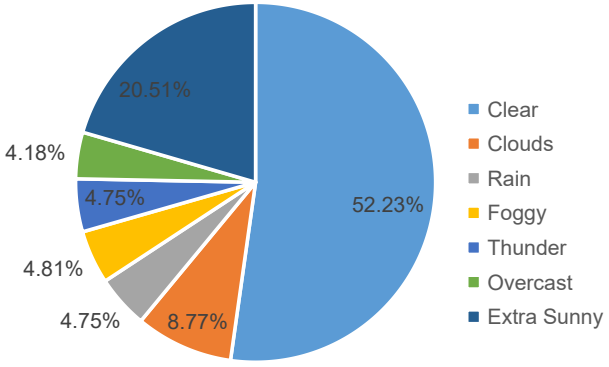

Figure 7.20 – The weather condition distribution on GCC dataset.

Table 7.4 – The results of our proposed M-MCNN and the three classic methods on GCC dataset.

Method	MAE	MSE
MCNN[[90]]	100.9	217.6
CSRNte[[42]]	38.2	87.6
SFCN[[77]]	36.2	81.1
M-MCNN(proposed)	35.5	79.3

good results in all test datasets used for evaluation. The GCC dataset is a multiple dataset, and our model has achieved good results in the GCC test dataset.

7.4.5/ CHDP DATASET

This data set is called the CHDP (high-density population) data set. It contains image of high-density people that we collected from Google, Baidu, and other major websites. The size of each high-density crowd image varies. Its purpose is to better detect the M-

MCNN crowd counting algorithm we have proposed. At the same time, we have added several existing data sets, which include: ShanghaiTech, USCD, WorldExpo'10 and GCC datasets. So as to make this composite data set more abundant. Of course, here we also introduced PSNR and SSIM image quality evaluation indicators [78]. Here we briefly introduce these two image quality evaluation indicators.

- ***PSNR (Peak Signal-to-Noise Ratio):***

  Given a clean image I and a noise image K of size m × n, the mean square error MSE is defined as [78]:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2 \tag{7.40}$$

  Then PSNR (dB) is defined as:

$$PSNR = 10 \cdot log_{10}\left(\frac{MAX_I^2}{MSE}\right) \tag{7.41}$$

  Where MAX_1^2 is the maximum possible pixel value of the image. If each pixel is represented by 8-bit binary, then it is 255. Generally, if the pixel value is represented by B-bit binary, then $MAX_I = 2^B$-1. Generally, for unit8 data, the maximum pixel value is 255; for floating-point data, the maximum pixel value is 1.

  The above is the calculation method for grayscale images. If it is a color image, there are usually three methods to calculate it.

  (1) Calculate the PSNR of the three RGB channels separately, and then take the average.

  (2) Calculate the MSE of the RGB three channels, and then divide by 3.

  (3) Convert the picture to YCbCr format, and then calculate only the Y component, which is the PSNR of the luminance component.

  Among them, the second and third methods are more common. For hyperspectral images, we need to calculate PSNR separately for different bands, and then take the average value, this indicator is called MPSNR.

- ***SSIM (Structural SIMilarity):***

  The SSIM formula is based on three comparative measures between the samples x and y: luminance, contrast and structure.

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{7.42}$$

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{7.43}$$

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{7.44}$$

  Here, we generally take c3 = c2/2. μ_x is the mean of x, μ_y is the mean of y, σ_x^2 is the variance of x, σ_y^2 is the variance of y, σ_{xy} is the covariance of x and y, c_1=$(k_1L)^2$ and

c_2=$(k_2L)^2$ are two constants, in order to avoid division by zero. L is the range of pixel values: $2^B - 1$, $k_1 = 0.01$, $k_2 = 0.03$ are the default value.

Then:

$$SSIM(x,y) = \left[l(x,y)^a \cdot c(x,y)^\beta \cdot s(x,y)^\gamma \right] \tag{7.45}$$

Set α, β, γ to 1, you can get [78]:

$$SSIM(x,y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{7.46}$$

Each time we calculate, we take an N * N window from the image. Then continuously sliding the window for calculation, and finally take the average as the global SSIM.

Of course, for hyperspectral images, we need to calculate SSIM separately for different bands, and then take the average. This indicator is called MSSIM.

The following figure shows the results of data comparison between M-MCNN and multiple crowd counting algorithms in a composite data set.

Participation Methods:

- ***CSRNet [42]:*** Congested Scene Recognition Network. CSRNet is a classical and efficient crowd counter, proposed by Li et al. in 2016. The authors design a Dilatation Module and add it to the top of the VGG-16 backbone. This network significantly improves performance in the field of crowd counting.
- ***PCC Net [17]:*** Perspective Crowd Counting Network. It is a multi-task network, which tackles the following tasks: densitylevel classification, head region segmentation, and density map regression. The authors provide two versions, a lightweight from scratch and VGG-16 backbone.
- ***MCNN [90]:*** Multi-Column Convolutional Neural Network. In 2016, it is a classical and lightweight counting model, proposed by Zhang et al.
- ***SCAR [18]:*** Spatial-/Channel-wise Attention Regression Networks. SCAR utilizes the self-attention module on the spatial and channel axis to encode the large-range contextual information. The well-designed attention models effectively extracts discriminative features and alleviates mistaken estimations.
- ***CANNet [43]:*** Context-Aware Network.CANNet combines the features of multiple streams using different respective field sizes. It encodes the multi-scale contextual information of the crowd scenes.
- ***Our method M-MCNN:*** Multi-feature multi-column convolutional neural network. A new framework M-MCNN based on MCNN for crowd counting on any single image. M-MCNN not only contains the original three columns of convolutional neural networks with different filter sizes, but replaces the fully connected layers with a convolutional layer of 1*1 filters , so the input image of the model can be of any size. To avoid distortion. Moreover, in a single individual sample, we greatly improve the learning of sample features by extracting the texture features of a single human head , and better use it for datasets.

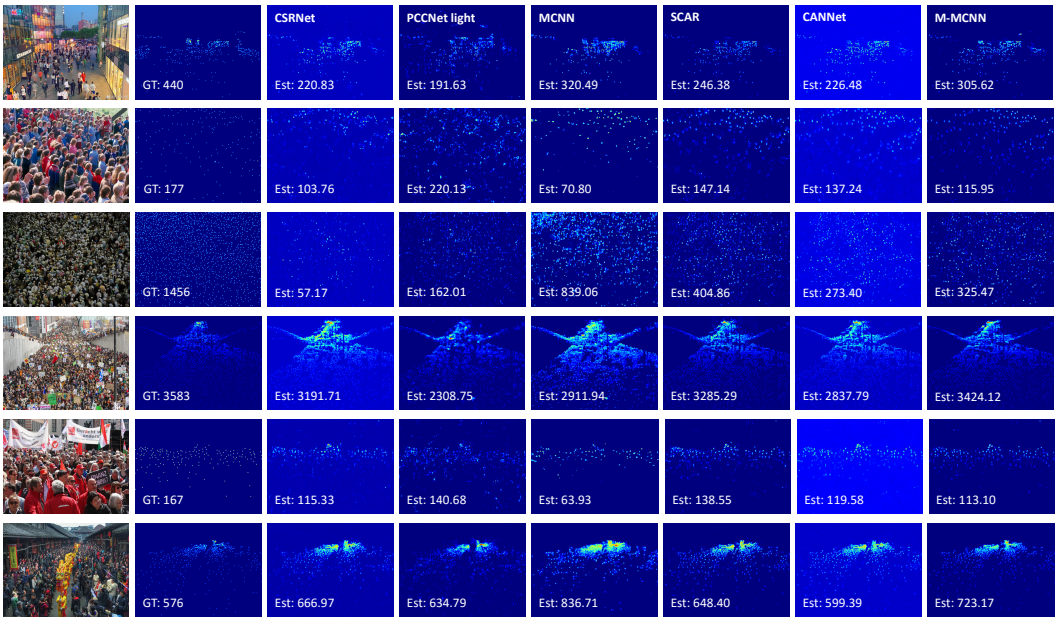

Figure 7.21 – The six groups of visualization results of some selected methods on the validation set.

- ***Conclusion:*** Here, we list the counting performance and density quality of all participation methods in Table 7.5. For evaluating the quality of the density map, two popular criteria are adopted, Peak Signal-to-Noise Ratio (PSNR) and Structural Similarity in Image (SSIM) [78]. In the calculation of PSNR, the negative samples are excluded to avoid zero denominators.

  Table 7.5 – Six methods of counting performance and density quality.

Method	Overall			
	MAE	MSE	PSNR	SSIM
CSRNet [42]	115.76	458.19	21.52	0.928
PCC Net [17]	102.76	589.12	20.42	0.877
MCNN [90]	225.83	732.65	19.48	0.865
SCAR [18]	93.86	490.57	21.22	0.913
CANNet [43]	95.58	514.36	21.01	0.895
M-MCNN	89.63	433.43	21.76	0.932

  From the table, we find M-MCNN attains the best counting performance, MAE of 89.63 and MSE of 433.43. At the same time M-MCNN produces the most high-quality density maps, PSNR of 21.76 and SSIM of 0.932.

7.5/ CONCLUSION

We propose a multi-feature multi-column convolutional neural network that can count the number of people. We extract feature maps from different layers and simultaneously resize them to have the same output dimension. After that, we fuse features so that the

fused features can be used to generate the density map. We also used texture features and target edge detection to reduce the loss of density map detail to better integrate with our convolutional neural network. We have performed a lot of experiments on ShanghaiTech, USCD, WorldExpo'10, GCC and CHDP datasets. Our model, M-MCNN is superior to other new population counting methods in the five datasets used to evaluate population counts.

Not only that, our model M-MCNN solves the serious occlusion problem between people well. Moreover, our model distinguishes human head features more finely, so as to better achieve high-density population counting. At the same time, we face another challenge. When some venues cannot be equipped with an intelligent video surveillance system, how can we estimate the high-density crowd area and avoid crowd trampling accidents? Then, we propose to use drones to estimate and predict high-density crowd areas in real time. In the next chapter, we detail its implementation process.

IV

CONTRIBUTION C

8

IMPLEMENTATION OF REAL TIME RECONFIGURABLE EMBEDDED ARCHITECTURE FOR PEOPLE COUNTING IN A CROWD AREA

In recent years, FPGAs have been increasingly used in areas such as speech recognition, machine learning, and cloud computing. This is due to the FPGA's powerful parallel computing capabilities and lower power consumption compared to general-purpose processors. However, these applications are mainly concentrated on large-scale FPGA clusters, which have extremely powerful processing capabilities to perform a large number of matrix or convolution operations. We will use a multi-column multi-feature crowd counting algorithm to implement its function through the FPGA.

8.1/ FPGAs

The FPGA is the field programmable gate array. It is the product of further development on the basis of programmable devices such as PAL, GAL, CPLD. It appears as a semi-custom circuit in the field of application-specific integrated circuits (ASIC), which not only solves the shortcomings of custom circuits, but also overcomes the shortcomings of the limited number of gate circuits of the original editable device.

Circuit design is completed by hardware description language Verilog or VHDL. It can go through the simple synthesis and layout and quickly test on FPGA. It is the mainstream technology of modern IC design verification. These editable elements can be used to implement some basic logic gate circuits, such as: AND, OR, XOR, NOT, or more complex functions such as decoders or mathematical equations. In most FPGAs, these editable components also contain memory components, such as Flip−flop or other more complete memory blocks. System designers can connect logic blocks inside the FPGA through editable connections as needed, as if a circuit test board was placed on a chip. The logic blocks and connections of a finished FPGA after leaving the factory can be changed according to the designer, so the FPGA can complete the required logic functions.

When the world 's first FPGA product, XC2064, was born in 1985, PCs destined to use a large number of chips had just stepped out of Silicon Valley laboratories and entered the

commercial market. The Internet was just a mysterious link for communication between scientists and government agencies. Wireless phones are as heavy as bricks. In the future, Bill Gates is struggling for livelihoods. Innovative programmable products seem to have little use.

This is indeed the case. Initially, FPGAs was only used for glue logic (Glue Logic), from glue logic to algorithm logic to digital signal processing, high-speed serial transceivers, and embedded processors. FPGA really changed from a supporting role to the protagonist. The idea that every electronic device will have a programmable logic chip in the next ten years is becoming a reality.

The FPGA uses the concept of logic cell array (LCA), which includes three parts: Configurable Logic Block (CLB), Input Output Block (IOB), and Interconnect. An FPGA is a programmable device. Compared with traditional logic circuits and gate arrays, such as PAL, GAL, and CPLD devices, FPGAs have different structures. FPGA uses a small lookup table 16*1RAM to implement combinational logic. Each lookup table is connected to the input of a D flip-flop. The flip-flop then drives other logic circuits or drives I/O, thus forming a combinational logic function, and can realize the basic logic unit module of sequential logic function. These modules are connected to each other or to I/O modules using metal wiring. The logic of the FPGA is implemented by loading programming data into the internal static storage unit. The value stored in the memory unit determines the logic function of the logic unit and the connection mode between modules or between modules and I/O. And finally researchers decided the function that the FPGA can realize, FPGA allows unlimited programming.

The main advantages of FPGA are:

- ***Advantage*** :
  - ***High programmable flexibility;***
  - ***Short development cycle;***
  - ***Parallel programming is highly programmable.***

FPGA and ASIC circuits are different. ASICs are fully custom circuits, while FPGAs is semi-custom circuits. In theory, if the gate circuit provided by FPGA can be satisfied, any logic function of ASIC and DSP can be realized by programming. In addition, the content of programming can also be changed repeatedly according to requirements. It is not like the programming of ASIC design cannot be modified after curing. Therefore, the application of FPGA is also more flexible. In actual programming design, FPGA has programmable performance, which allows developers to use software upgrade packages to run on the chip through software upgrade packages. At the same time, the original program of the chip is modified, thereby avoiding the replacement of the hardware chip. More convenient is the FPGA, which can also be upgraded remotely via the Internet. The parallel computing efficiency possessed by FPGAs is very high. The FPGA uses an algorithm that can execute multiple instructions in parallel at one time. Hence, the ASIC, DSP and CPU chips used in general life use serial computing. They can only process instructions individually. If you need to accerlerate the operation of ASIC and CPU in circuit design, a rule of thumb is to increase the frequency. Although the general frequency of FPGAs is typically low, for some special requirements designs, having a large number of relatively low-speed parallel units is more efficient than having a few of

high-efficiency units. The method of processing the final result is very similar to the effect of ASIC, so the execution efficiency is greatly improved.

8.1.1/ FPGA Reconfiguration

Reconfiguration technology is a design method for real-time reconfiguration of some or all logic resources on a programmable device during system operation. This technology can effectively improve the applicability of the FPGA hardware platform to system functions. While ensuring system performance and flexibility, it significantly reduces the cost of the system.

Reconfiguration computing technology depends on the editability of the device. This makes it easier to customize the device to the functional components needed for the computing task during the calculation. So, the programming technology of FPGA in the common configuration computing system mainly includes antifuse, Flash and SRAM.

First: The anti-fuse FPGA is programmed by a dedicated editor based on the anti-fuse array of the configuration data given by the design. It has the advantages of fast speed and good performance. However, its disadvantage is that it has only one-time programming capability.

Second: Flash's FPGA integrates SR-type storage structure. SRAM is used to control the system when the device is working normally. Flash is used to save configuration information. This type of FPGA makes full use of the reconfigurability of Flash, but it is not particularly ideal in terms of configuration speed.

Third: Each configuration point in the SRAM FPGA is connected to the SRAM. In fact, the configuration process is the programming process of these SRAM storage bits. This type of FPGA has the ability to be reprogrammed and is fast to configure. It is most widely used in the field of reconfiguration computing.

8.1.1.1/ Classification of reconfiguration systems

In terms of FPGA configuration, the reconfiguration system is divided into:

(1) Global reconfiguration, that is, reconfiguration of the entire FPGA can change the operating logic.

(2) Partial reconfiguration, that is, while part of the logic unit of the FPGA is running, it can configure the logic unit of the other part without affecting the running part.

From the perspective of FPGA configuration, reconfigurable systems are divided into:

(1) Being reconfigured, that is, the system controls the initialization of files and the occurrence of reconfiguration by external signals.

(2) Self-reconfiguration, that is, the system itself controls the occurrence of reconfiguration.

(3) Mix reconfiguration, a reconfiguration mode that combines the above two modes.

8.1.1.2/ STRUCTURE OF THE RECONFIGURATION SYSTEM

An architecture of the reconfiguration computing system is composed of one or more microprocessors, special processing modules, and reconfiguration modules. The dedicated processing module communicates with the microprocessor through a shared bus or some special internal network connection. Microprocessors and specialized processing modules may use some on-chip memory as a local cache. In addition, since the programmable logic device can support dynamic reconfiguration, the dedicated the dedicated processing module can be changed dynamically. Therefore, corresponding control and management components are needed to deal with those specialized modules that can be dynamically added or deleted. At the same time, it handle internal communication issues after reconfiguration. This tightly coupled architecture is shown below:

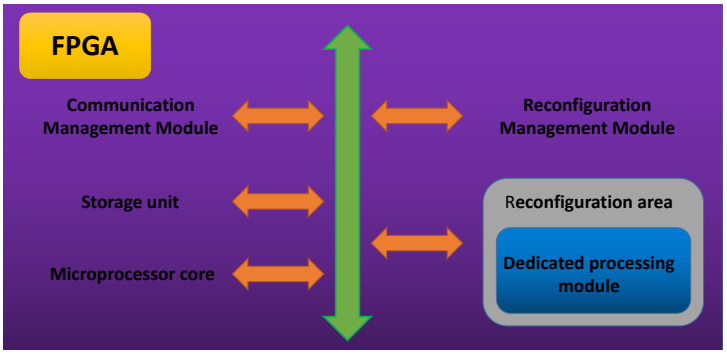

Figure 8.1 – Basic structure of the reconfiguration system.

8.1.1.3/ FPGA RECONFIGURATION TECHNOLOGY

Xilinx company first proposed to reconfigure computing technology in FPGA. At present, there are three types of reconfiguration technologies proposed by Xilinx: System ACE technology, ICAP technology, and the combination of System ACE and ICAP technology. From the aspect of reconfiguration, the first one belongs to global reconfiguration, and the other two belong to local reconfiguration.

System ACE technology:

System ACE technology can be used to solve the problem of dynamic reconfiguration of a single-chip FPGA system. The system consists of a System ACE controller and a general commercial CF card. The System ACE controller reads different configuration files stored on the CF card through its configuration address CFGADD [0 -2]. Global reconfiguration of the target FPGA device through the JTAG port. From a configuration perspective, this technology is reconfigured. Because there are only three configuration address lines, the System ACE controller can only support up to eight different configuration schemes at the same time. The specific process can be divided into 3 stages:

(1) Use a synthesis tool to synthesize different configuration schemes into a BIT file.

(2) Use the impact tool to convert the BIT file into a System ACE file, and then burn the System ACE file with the ace extension to the CF card.

(3) System ACE controls the occurrence of reconfiguration through an external controller.

ICAP technology:

ICAP (Internal Configuration Access Port) is an internal configuration interface provided on the FPGA chip on Virtex-II. The ICAP interface is a hard module, and its position on the FPGA chip is fixed in the lower right corner of the entire array. The FPGA internal logic implements reading and writing of configuration memory through the module's 32-bit input data bus and 32-bit output data bus.

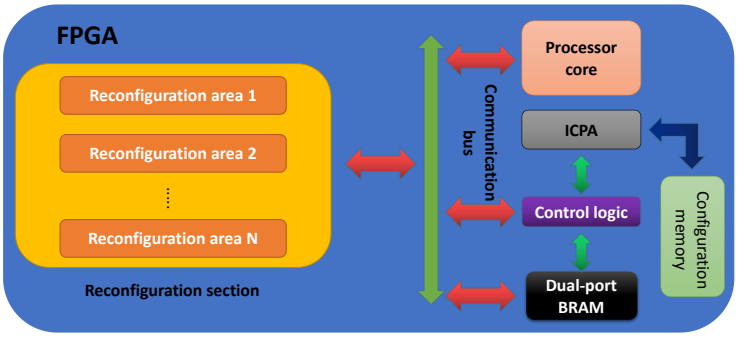

Figure 8.2 – System structure of ICPA self-reconfiguration technology.

The process of ICAP technology to achieve the local dynamic reconfiguration is as follows: the microprocessor communicates with the control logic through the communication bus. A program running on a microprocessor makes a request to read in certain configuration data. The reconfiguration control module reads the configuration data back into the dual-port block RAM through the ICAP element. When the readback is completed, the microprocessor modifies it appropriately. The modified configuration data is then written back to the device via the ICAP element. The microprocessor can be an internal processor of the FPGA or an external processor. If it is an internal processor, ICAP technology is a self-reconfiguration mode. If it is an external processor, ICAP technology belongs to the reconfigured mode.

System ACE and ICAP combined technology:

System ACE and ICAP combine the advantages of both reconfiguration technologies. Through static configuration generation, module-level reconfiguration features and local reconfiguration features are combined. This is a hybrid reconfiguration. When the system is powered on, the System ACE controller automatically reads the ACE file stored in the large-capacity CF card. Configure the target FPGA and start the hard core of the microprocessor embedded in the FPGA. Then use the microprocessor core to complete the partial reconfiguration of the target FPGA through the ICAP interface. System ACE technology, ICAP technology and System ACE, ICAP fusion technology can complete the reconfiguration of the FPGA. All three technologies have different characteristics. System ACE technology controls the occurrence of reconfiguration by external signals, which is suitable for systems with static configuration generation. Its advantage is that the control is simple, but the configuration takes a long time. ICAP technology can quickly complete the partial reconfiguration of the FPGA. The advantage of ICAP technology is that it is fast to configure. The disadvantage is that the controls are particularly complex. System ACE and ICAP fusion technology combine the advantages of the former two, which is more convenient and faster.

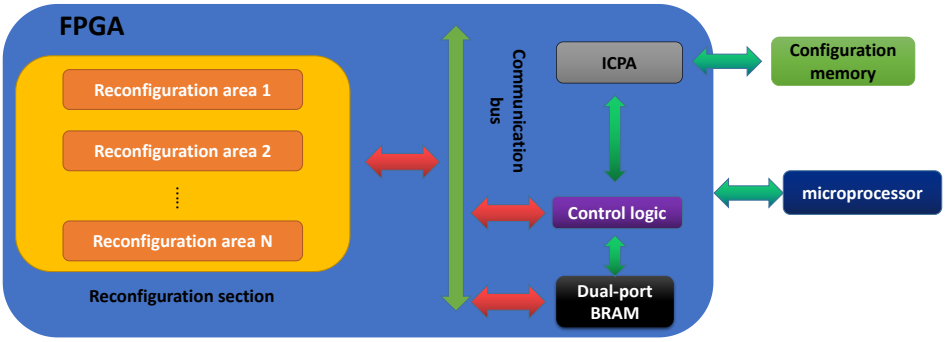

Figure 8.3 – System structure of ICAP reconfiguration mode.

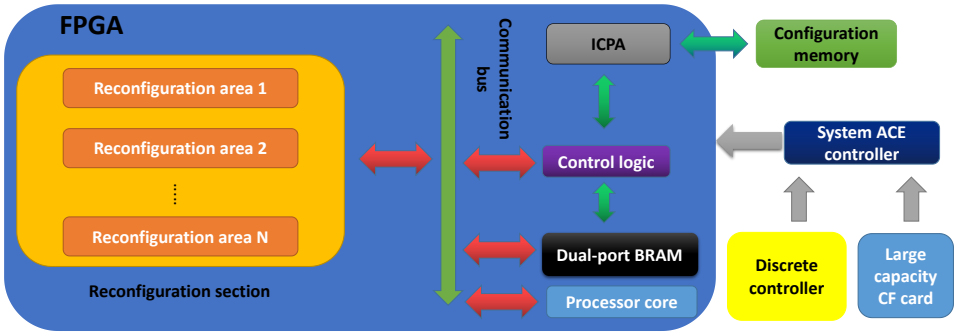

Figure 8.4 – System ACE and ICAP technology system structure.

8.1.1.4/ APPLICATION OF RECONFIGURATION TECHNOLOGY

FPGA reconfiguration computing technology can flexibly meet multiple functional requirements by changing the device configuration. This allows a single reconfiguration device to meet different design needs. Therefore, the reconfiguration technology has real-time processing capabilities, adaptive capabilities, and fault tolerance. At the same time, it can effectively reduce energy consumption, which has great advantages in terms of performance and flexibility.

Internet remote reconfiguration technology:

Internet remote reconfiguration technology is a new FPGA design concept proposed by Xilinx. Its core idea is to use reconfiguration technology to upgrade, debug, and monitor the hardware design and software programs of remote devices through the network. In the internet environment, it access the remote target machine through a browser, and implement data and file transfer through HT-TP, FTP protocol, Java Applet, and CGI. This design concept enables remote updates and dynamic reconfiguration of the software and hardware of the target system via the Internet.

Multi-bus standard bridging technology:

At present, in the automotive field, electronic systems have multiple bus standards, such as: LIN, CAN, 1394, and so on. The technical parameters of different standard buses are very different. The multi-bus standard bridging technology mainly uses reconfiguration technology to reconfigure the on-board vehicle electronic system. This makes it possible to flexibly implement bridging between various standard buses without changing

hardware resources. At present, the cost of in-vehicle electronic system accounts for an increasing proportion of the entire manufacturing cost, and the multi-bus standard bridge technology can very well improve the repeatability of the hardware platform.

FPGA self-healing technology:

In certain environments, logic circuits can be easily damaged or altered. As a result, the logic circuits in this area cannot operate normally. The FPGA self-healing technology is to relocate the logic structure of the faulty chip by using dynamic reconfiguration technology based on locating the logic circuit where the error occurred. At the same time, the configuration data corresponding to this part of the logic circuit function are configured to other valid FPGA areas through relocation technology. Hence, it skip the damaged part and continue working normally. FPGA self-healing technology can make the reconfiguration device self-adaptive and self-healing through dynamic reconfiguration, effectively extending the life of the chip.

FPGA reconfiguration technology can reconfigure logic functions at any time during system operation. It achieves the reuse rate of logical resources and also improves the utilization of resources. Reconfiguration technology combines the high performance of hardware with the flexibility of software. For computationally intensive applications, the reconfiguration technology further reflects the advantages of development parallelism. The development of very large-scale integrated circuits, a large number of high-speed and low-consumption FPGAs are used in various fields. The function of the embedded central processing unit in FPGA is getting stronger and stronger, and the reconfiguration technology has been more widely used.

8.1.2/ VIVADO: DESIGN TOOL FOR XILINX FPGA

Xilinx is a well-known developer, and vivado is the mainstream tool for Xilinx developer. We next introduce an overview of the process that Xilinx vivado uses to develop.

8.1.2.1/ VIVADO DEGISN FLOW

The guide provides an overview of working with the Vivado® Design Suite to create a new design for programming into a Xilinx® device. It provides a brief description of various use models, design features, and tool options, including preparing, implementing, and managing the design sources and intellectual property (IP) cores.

The Vivado Design Suite offers multiple ways to accomplish the tasks involved in Xilinx device design, implementation, and verification. We can use the traditional register transfer level (RTL)-to-bitstream FPGA design flow, as described in RTL-to-Bitstream Design Flow. We can also use system-level integration flows that focus on intellectual property (IP)-centric design and C-based design, as described in Alternate RTL-to-Bitstream Design Flows.

Design analysis and verification is enabled at each stage of the flow. Design analysis features include logic simulation, I/O and clock planning, power analysis, constraint definition and timing analysis, design rule checks (DRC), visualization of design logic, analysis and modification of implementation results, programming, and debugging.

RTL Design

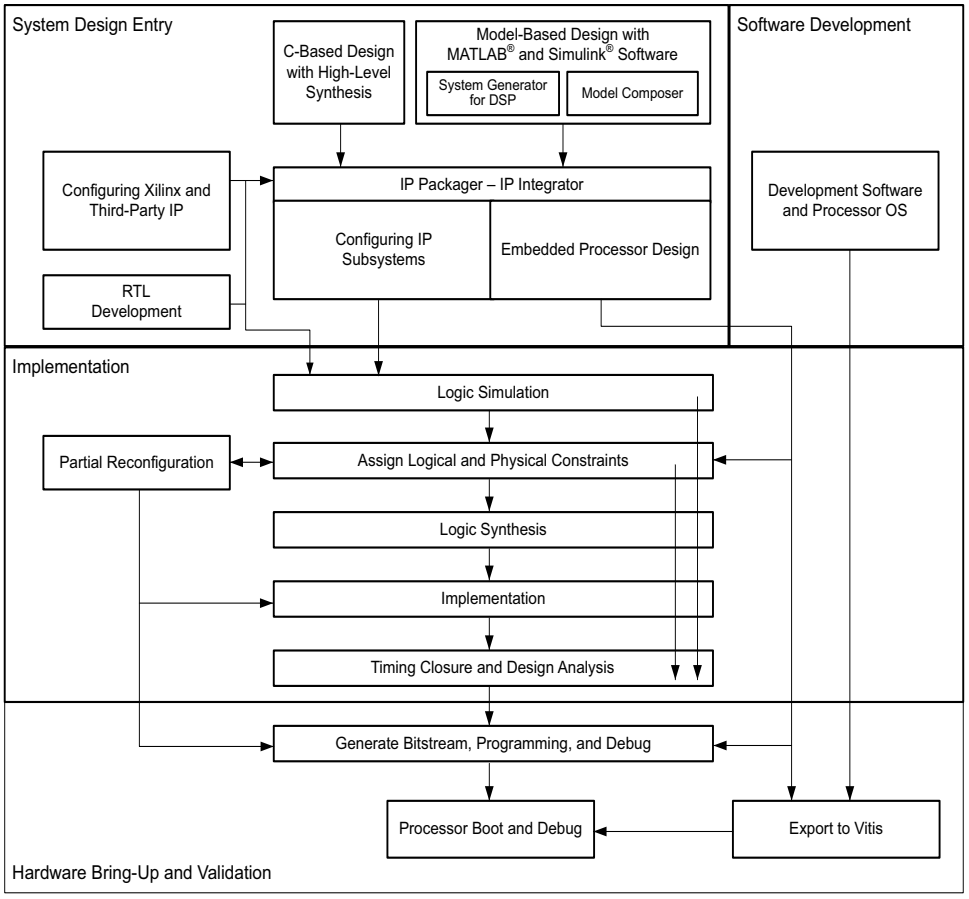

Figure 8.5 – High-level design flow in the Vivado Design Suite.

We can specify RTL source files to create a project and use these sources for RTL code development, analysis, synthesis and implementation. Xilinx supplies a library of recommended RTL and constraint templates to ensure RTL and XDC are formed optimally for use with the Vivado Design Suite. Vivado synthesis and implementation support multiple source file types, including Verilog, VHDL, SystemVerilog, and XDC.

The UltraFast Design Methodology Guide for the Vivado Design Suite (UG949) focuses on proper coding and design techniques for defining hierarchical RTL sources and Xilinx design constraints (XDC), as well as providing information on using specific features of the Vivado Design Suite, and techniques for performance improvement of the programmed design.

High-Level Synthesis C-Based Design

The C-based High-Level Synthesis (HLS) tools within the Vivado Design Suite enable us to describe various DSP functions in the design using C, C++, and SystemC. We create and validate the C code with the Vivado HLS tools. Use of higher-level languages allows us to abstract algorithmic descriptions, data type, specification, etc. We can create “what-if” scenarios using various parameters to optimize design performance and device area.

HLS lets us simulate the generated RTL directly from its design environment using C-based test benches and simulation. C-to-RTL synthesis transforms the C-based design into an RTL module that can be packaged and implemented as part of a larger RTL

design, or instantiated into an IP Integrator block design.

IP Design and System-Level Design Integration

The Vivado Design Suite provides an environment to configure, implement, verify, and integrate IP as a standalone module or within the context of the system-level design. IP can include logic, embedded processors, digital signal processing (DSP) modules, or C-based DSP algorithm designs. Custom IP is packaged following IP-XACT protocol and then made available through the Vivado IP catalog. The IP catalog provides quick access to the IP for configuration, instantiation, and validation of IP. Xilinx IP utilizes the AXI4 interconnect standard to enable faster system-level integration. Existing IP can be used in the design either in RTL or netlist format.

Xilinx Platform Board Support

In the Vivado Design Suite, we can select an existing Xilinx evaluation platform board as a target for us design. In the platform board flow, all of the IP interfaces implemented on the target board are exposed to enable quick selection and configuration of the IP used in your design. The resulting IP configuration parameters and physical board constraints, such as I/O standard and package pin constraints, are automatically assigned and proliferated throughout the flow. Connection automation enables quick connections to the selected IP.

Synthesis

Vivado synthesis performs a global, or top-down synthesis of the overall RTL design. However, by default, the Vivado Design Suite uses an out-of-context (OOC), or bottom-up design flow to synthesize IP cores from the Xilinx IP Catalog and block designs from the Vivado IP integrator. We can also choose to synthesize specific modules of a hierarchical RTL design as OOC modules. This OOC flow lets us synthesize, implement, and analyze design modules of a hierarchical design, IP cores, or block designs, out of the context of, or independent from the top-level design. The OOC synthesized netlist is stored and used during top-level implementation to preserve results and reduce runtime. The OOC flow is an efficient technique for supporting hierarchical team design, synthesizing and implementing IP and IP subsystems, and managing modules of large complex designs.

The Vivado Design Suite also supports the use of third-party synthesized netlists, including EDIF or structural Verilog. However, IP cores from the Vivado IP Catalog must be synthesized using Vivado synthesis, and are not supported for synthesis with a third-party synthesis tool. There are a few exceptions to this requirement, such as the memory IP for 7 series devices.

Placement and Routing

When the synthesized netlist is available, Vivado implementation provides all the features necessary to optimize, place and route the netlist onto the available device resources of the target part. Vivado implementation works to satisfy the logical, physical, and timing constraints of the design.

For challenging designs the Vivado IDE also provides advanced floorplanning capabilities to help drive improved implementation results. These include the ability to constrain specific logic into a particular area, or manually placing specific design elements and fixing them for subsequent implementation runs.

8.1.2.2/ VIVADO ENVIREMENT

The Vivado Design Suite also has specific versions of the HL system version, HL design version, and HL WebPACK ™ version. The HL System Edition and HL Design Edition support functions are almost the same. However, HL System Edition only supports System Generator for DSP. In addition, Vivado HL Design Edition and HL System Edition include Vivado® Design Suite HLx versions, including partial reconfiguration, with the advantage of avoiding additional costs.

The new Vivado® Design Suite HLx release provides the tools and methodologies required by design teams. It leverage C-based design and optimization reuse, IP subsystem reuse, integrated automation and accelerated design completion when combined with the UltraFast ™ High-Level Productivity Design Methodology Guide. This particular combination is proven not only to help designers work in a high-level abstraction, but also to promote reuse, which can accelerate productivity.

8.2/ MULTI-COLUMN MULTI-FEATURE CONVOLUTIONAL NEURAL NETWORK CROWD COUNTING ARCHITECTURE IMPLEMENTED IN REAL-TIME RECONFIGURABLE EMBEDDED SYSTEM

FPGA chip is a logic gate circuit element with programmable function. It has powerful parallel processing data capabilities and advantages. Convolutional neural networks have powerful feature extraction capabilities, making them widely used in image classification and recognition, target tracking and other fields. Next, we implemented a crowd counting algorithm based on a multi-column multi-feature convolutional neural network that was implanted on the FPGA chip. We ues an FPGA to realize real-time crowd counting function.

8.2.1/ FPGA-BASED CROWD DETECTION AND ESTIMATION

Here, we mainly study the application of FPGA in the field of video image processing. Video image processing is a hot technology in the multimedia field. The amount of data processed by video images is increasing. Based on this large amount of data, it can be divided into two categories: video codec and target recognition. One of our main research directions is target recognition. Object recognition is mainly used to extract related information, such as image edge extraction. Crowd detection and counting functions are implemented via FPGA.

FPGA implements the detection of certain parts of a person's body, such as the detection of the head, shoulders and other parts of the body:

In 2007, Gardel et al. proposed real-time head detection based on an embedded vision module [19]. The main process is divided into the following steps: 1. Dynamic background extraction and object detection. 2. Edge feature extraction. 3. Introduce a Huff counter by applying a pattern with a set of ring patterns. 4. Pass the candidate matching and tracking framework.

Images are processed in parallel through different ring patterns, and head features of different sizes are detected simultaneously. The developed system uses hardware processing and uses almost 100% of Spartan3 (1.5Mgates) resources [71]. Detection of human head features is achieved through a very low cost circuit design.

In 2012, Tomasz Kryjak et al. proposed a system for implementing head and shoulder detection in FPGAs [39]. It performs feature extraction based on local binary patterns and classification based on support vector machines. To reduce false alarm rates, foreground object detection is used as an additional verification criterion.

On the one hand: in a system for head and shoulder detection, for feature extraction, a local binary mode is used, and texture descriptors and support vector machines (SVM) are applied for classification. Additional background generation and foreground objects use the segmentation module. In the final implementation of the head and shoulders detection, only the objects that effectively reduce the false detection rate are performed in the foreground.

On the other hand: used in hardware systems for head and shoulder detection, it mainly contains three modules: NRULBP function, SVM classification and foreground object detection. Everything is integrated into a single FPGA device and implemented on a Xilinx ML605 evaluation board with Virtex 6 FPGA devices. The system processes video stream with resolution of 640x480 pixels @ 60fps. The described design can be incorporated within a video surveillance system, a PC accerlerator, or embedded into a smart camera.

In 2015, Megalingam and others proposed an FPGA-based solution for automatic navigation of indoor electric wheelchairs [50]. Here, the house is divided into virtual grids, and each grid is assigned specific coordinates. Through experiments, a predefined path is implanted into the system. The system runs faster when it arrives from the departure point to the destination. Instead, the limitation of this system is that it can only follow a fixed path.

Implementation of people counting function in FPGA:

In 2010, Gasparini et al. proposed an integrated camera processor node through which people are detected and counted [21]. They plan each frame segment as a feature model used in statistics. This camera-contrast based implementation is described as an independent counter for people crossing the gate. This simple application highlights the main functions of the sensor and provides examples of integration with the processing unit. All of these new features can be easily accommodated in current Flash-FPGA implementations.

During the following year, in 2011, Gasparini et al. Proposed the implementation of a person counter based on Dijkstra's algorithm on an FPGA [20]. The development application is a personnel counter based on Dijkstra's algorithm and on an FPGA-based node. The clock frequency is 10 MHz, and it has good classification performance when acquiring images at 30 fps. Despite the inherent complexity of the application, the node consumes only 5mW. The most critical factor is to hire a smart vision sensor that can perform pre-processing chips directly on the sensor. They use its inherent multi-tasking FPGA to implement an internally efficient architecture. This results in better counting of people.

In 2019, Ahmad et al. proposed an implementation based on an improved gradient histogram (HOG), support vector machine (SVM), and a new person counting system [1]. The proposed system is based on a new image processing technology with a simple

architecture. The algorithm is based on HOG and optimized SVM classifier. They use SVM classifier to classify between human and non-human. SVM is a training-based classifier, which means that it can be trained for various situations. After the algorithm was implemented on the PC, they adopted the architecture of the Vertex 6 series FPGA for testing. Because of the extensive pipeline design, the clock rate is 196.2MHz. With this high-definition frame rate video stream with a clock frequency of 6 frames per second, the experimental results reflect a reduction in resource usage and use only 3% slices and 49% bounded IOBs. The personnel counting function is well implemented on FPGA.

8.2.2/ Multiple hardware implementations of deep learning algorithms

The development of artificial intelligence is getting faster and faster. Deep learning has played a very important role in its development. In spite of its powerful simulation prediction capabilities, deep learning still faces the problem of huge computational load. At the hardware level, GPUs, ASICs, and FPGAs are the first solutions to solve the huge computational load.

8.2.2.1/ CPU hardware

In 2006, people still used serial processors to deal with machine learning problems. At that time, Mutch and Lowe developed a tool FHLib (feature hierarchy library) to deal with the hierarchical models. For the CPU, it requires less programming and has the benefit of migration. But the characteristics of serial processing have become its shortcomings in the field of deep learning, and this shortcoming is fatal. It is 2020 now, the development of integrated circuits in the past decade still follows Moore's Law, and the performance of the CPU has been greatly improved. However, this did not get the CPU into the perspective of deep learning researchers again. Although the CPU can perform a certain amount of computing power on small data sets, and multiple cores enable it to process in parallel, it is still not enough for deep learning.

8.2.2.2/ GPU hardware

GPUs have also come into sight of researchers. Compared to the CPU, the number of GPU cores has greatly increased. This also allows it to have more powerful parallel processing capabilities, and it also has more powerful ability to control data flow and store data. In 2008, Chikkerur carried out the difference between the CPU and the GPU in processing target recognition capabilities [9]. The final GPU processing speed is 3-10 times that of the CPU.

8.2.2.3/ Application Specific Integrated Circuit Chip (ASIC)

Application Specific Integrated Circuit Chip (ASIC) due to its customized characteristics, is a more efficient method than the GPU. But its customization also determines its low portability. When dedicated to a well-designed system, migration to other systems is not possible. Moreover, its high cost and long production cycle make it unconsidered in the

current research. Of course, its superior performance can still be used in some areas. In 2010, Kim, J. Y. et al. Applied ASIC to a bio-heuristic neural perception engine processor for real-time multi-target recognition. The recognition rate can reach 60 frames/second in a 640*480 pixel image [37].

8.2.2.4/ FPGA HARDWARE

FPGAs make a good trade-off between GPU and ASIC. FPGAs take into account processing speed and control capabilities. On the one hand, the FPGA is programmable reconfigurable hardware, so it has more powerful controllability than the GPU. On the other hand, the increasing gate resources and memory bandwidth make it have more design space. More conveniently, FPGA also eliminates the tape-out process required in the ASIC solution. One disadvantage of FPGAs is that they require users to be able to program them using a hardware description language. However, technology companies and research institutes have developed easier-to-use languages such as Impulse Accelerated Technologies Inc. They have developed C-to-FPGA compilers to make FPGAs more user-friendly. Yale's E-Lab developed the Lua scripting language [34]. These tools have shortened the researcher's development time limit to a certain extent, and made the research easier and more convenient.

8.2.2.5/ THE ADVANTAGES AND DISADVANTAGES OF FPGA IMPLEMENTATION OF DEEP LEARNING

Computational power is generally characterized by two parameters:

- ***Peak GOPs***
- ***Real GOPs measured performance (for specific networks)***

The FPGA can achieve high Real GOPs / Peak GOPs in the inference process.

- ***FPGA computing power advantages:***
  - ***Low latency during inference, especially when batch size is 1.***
    The advantages of GPUs are block processing, batch data, and batch computing. This can use his massive computing unit, as well as external storage. But when inference, the batch size is 1. The advantage of FPGA pipeline design is obvious.
  - ***Customized calculation engine.***
    This refers to the array-style, reconfigurable data flow engine (weighting, data inflow, reasonable coordination of calculations). With the design of a large number of distributed RAM, FPGAs can be adapted to specific neural networks. The Real GOPs / Peak GOPs ratio for application scenarios is high. In comparison, the Real Gops of FPGAs in some neural networks may indeed exceed GPUs.

  - ***Continuously promoted software and hardware fusion technology.***
    The algorithm optimizes the compression network, compression weights, and the adaptive NPU structure.

- ***Disadvantages of FPGA devices:***

  - ***FPGA devices are not suitable for floating-point operations. The training process is basically floating-point arithmetic.***
    The arithmetic unit inside the FPGA is mainly DSP (no floating point unit), which is suitable for fixed-point calculation. If the accuracy is intercepted in the iterative calculation process, then in the back propagation process, the calculation error is accumulated layer by layer. The deeper the depth, the greater the cumulative error. The propagated weight parameter will either tend to be zero or saturated, which will cause training failure.

  - ***There are many types of calculations required during the training process, and FPGAs are expensive to implement certain operations.***
    The normalization of the middle node of the during the training process, the root operation in ADAM, and so on. If only the forward propagation multiply-add operation is placed in the FPGA, the reverse gradient calculation is placed on the CPU. Each iteration will cause a large number of parameters and intermediate calculation results to be repeatedly transferred between the CPU and FPGA, thereby offsetting the advantages obtained by hardware acceleration.

8.2.3/ FPGA Development Board

Currently, artificial intelligence and neural networks are finding new uses in many applications. The biggest and fastest growing of them is computer vision. Zynq UltraScale+ ZCU102 is one of the best platforms for developing embedded vision applications today for the following reasons: It has an HDMI input and HDMI output, and it has an FPGA architecture that can be used for hardware-accelerated image processing algorithms. Here, we use the Zynq UltraScale+ ZCU102 FPGA development board to implement a crowd counting algorithm based on a multi-column multi-feature convolutional neural network.

8.2.3.1/ Introduction to Zynq UltraScale+ ZCU102

The ZCU102 Evaluation Kit enables designers to jumpstart designs for automotive, industrial, video, and communications applications. This kit features a Zynq® UltraScale+™ MPSoC with a quad-core Arm® Cortex®-A53, dual-core Cortex-R5F real-time processors, and a Mali™-400 MP2 graphics processing unit based on Xilinx's 16nm FinFET+ programmable logic fabric. The ZCU102 supports all major peripherals and interfaces, enabling development for a wide range of applications.

- ***Key Features & Benefits***

  - ***Optimized for quick application prototyping with Zynq UltraScale+ MPSoC***

- ***DDR4 SODIMM – 4GB 64-bit w/ ECC attached to processing system (PS)***
- ***DDR4 Component – 512MB 16-bit attached to programmable logic (PL)***
- ***PCIe® Root Port Gen2 x4, USB3, Display Port, and SATA***
- ***4x SFP+ interfaces for Ethernet***
- ***2x FPGA Mezzanine Card (FMC) interfaces for I/O expansion, including 16 16.3Gb/s GTH transceivers and 64 user-defined differential I/O signals***

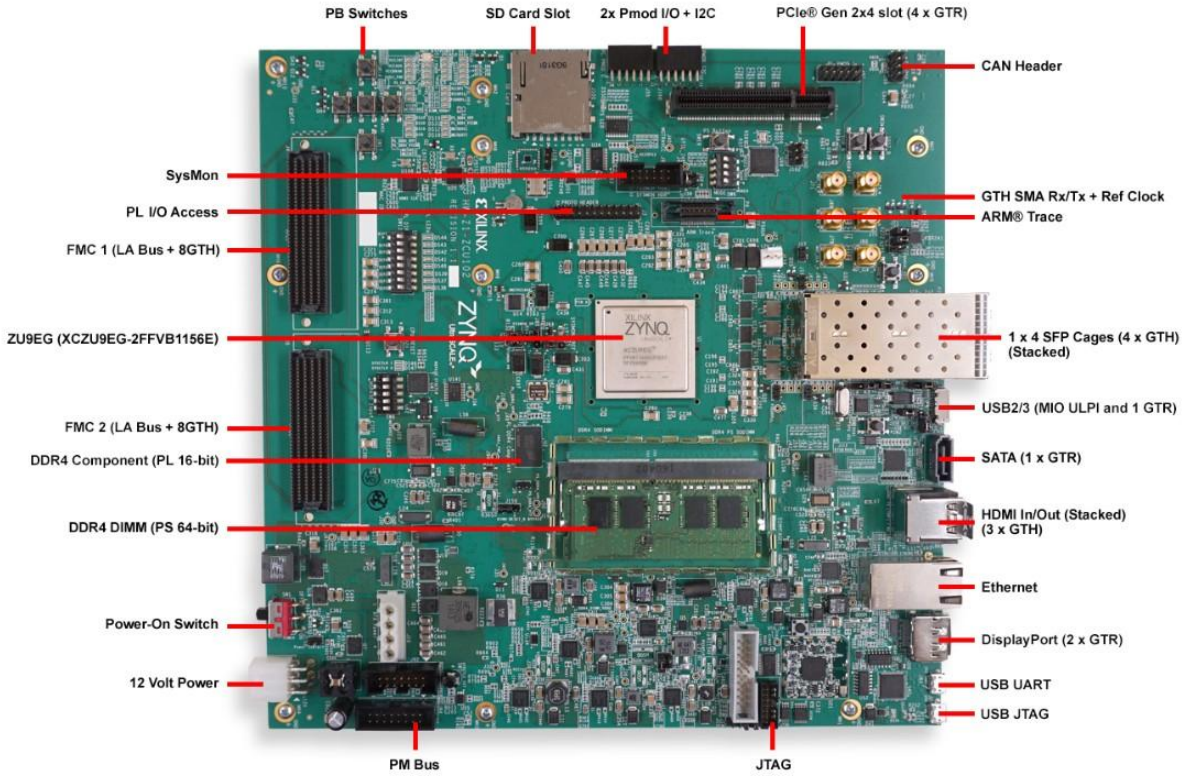

Figure 8.6 – Zynq UltraScale+ ZCU102.

Here we simple introduce the memory and communication network design of Zynq UltraScale+ ZCU102.

- ***Memory***
  - ***PS 4GB DDR4 64-bit SODIMM w/ ECC***
  - ***PL 512MB DDR4 component memory ([256 Mb x 16] devices) at 1200MHz / 2400Mbps DDR***
  - ***8KB IIC EEPROM***
  - ***Dual 64MB Quad SPI flash***
  - ***SD card slot***

- ***Communication & Networking***
  - ***RGMII communications at 10, 100, or 1000 Mb/s. Serial GMII interface-supports a 1 Gb/s SGMII interface***
  - ***4x SFP+ cage***
  - ***SMA GTH support (4x SMA Tx/Rx connectors)***
  - ***UART To USB bridge***
  - ***RJ45 Ethernet connector***
  - ***SATA (1 x GTR)*****
  - ***PCIe Gen2x4 Root Port*****

8.2.3.2/ VIVADO HLS

The programming language we use here is Vivado HLS. The Vivado High-Level Synthesis compiler enables C, C++ and SystemC programs to be directly targeted into Xilinx devices without the need to manually create RTL. Vivado HLS is widely reviewed to increase developer productivity, and is confirmed to support C++ classes, templates, functions and operator overloading.

8.2.3.3/ FRAMEWORK SPOONN

Framework spooNN is based on C and Vivado HLS programming language. It provides basic convolution operation modules such as image to matrix, matrix multiplication, max-pool, batch normalization, etc. Zhang et al. ranked second in the 2018 and 2019 DAC object detection network competition, delivering the highest FPS at lowest power consumption for object detection [88].

8.2.4/ EXPERIMENT

8.2.4.1/ EXPERIMENT DESIGN

Through previous experiments, the time for each high-density crowd image tested in our strong feature part is between 7-9 seconds, while the time for each high-density crowd image tested in the multi-column neural convolutional network part is about 20 seconds. Due to the long running time of the high-density crowd counting part of the multi-column convolutional neural network, the overall architecture runs too slow. In this case, we recommend using a parallel structure to speed up the calculation, that is, the heavier part of the algorithm (Multi-column neural convolutional network) runs in the accelerator. The number of layers of the multi-column convolutional neural network is uniformly set to (16.32.16.8), in order to better cooperate with the FPGA algorithm implementation without losing accuracy, while the other parts run in the CPU.

- ***Coefficient codage***

In the experimental part, we use 16 bits integer to represent each value in the image. Compared with 32 bits floating-point, using 16 bits integer under appropriate circumstances will not observe the lost of the accurary visible to the naked eye [38], and the model will be doubled. For exemple, the model that was able to run with 24GB of video memory now requires 12GB of video memory and is fast. We use the post-training quantization method proposed in TensorFlow [38] to quantize the well-trained fully floating-point network. According to our experiments, the accuracy of the quantization network reaches 97.2%. Compared with 32-bit floating point networks, the loss of accuracy is negligible. Not only can the size of the model be reduced, but also the latency of the CPU and hardware accelerator can be increased, and the accuracy of the model will hardly be reduced.

- ***Hardware design***

We use spoonn as the fpga programming framework [88]. Figure 8.7 shows the overall layout of the accelerator. At the beginning of the calculation, the cpu writes the weight into the fpgq through the channel. When the weight is written, the cpu resizes and preprocesses the image, and sends the image to the accelerator through the channel.

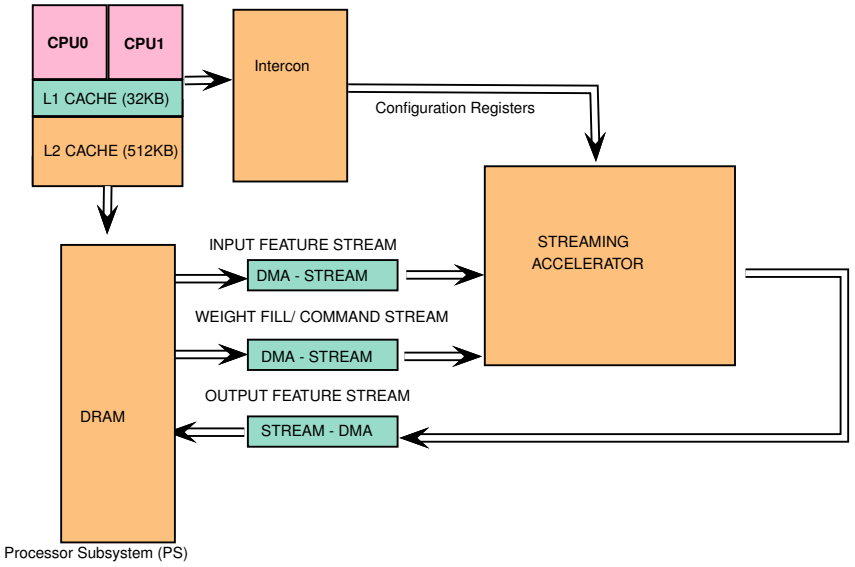

Figure 8.7 – Overall Hardware Architecture.

First, we perform image processing. We use images with a resolution of 512*512. The image is expressed using the RGB three-color method. Therefore, the size of the input matrix vector is 512*512*3. Here we use half floating point, which means using 16 bits floating-point, to represent each value in the image. Therefore, each pixel will be expressed using 3x16 bits. We use the axis stream with width = 512 bits to communicate between the CPU and the FPGA device. Therefore, we compress the value of the same pixel in 48 bits. Every time we send 8 pixels, that is 384 bits, a total of 512 * 512/8 = 32768 batches are sent.

The neural network uses 12 convolutional layers. The 12 convolutional layers are divided into three branches (branch), we named the branches: A, B, C. Set column A to use

large-scale convolution kernels: 9 * 9, 7 * 7, 7 * 7, 7 * 7, column B to use medium-scale convolution kernels: 7 * 7, 5 * 5, 5 * 5, 5 * 5. Column C uses small-scale convolution kernels: 5 * 5, 3 * 3, 3 * 3, 3 * 3. In the branch, the output of the previous convolution can be used as the input of the latter convolution. Each convolutional layer does not interfere with each other, and a pipeline architecture can be used to accelerate the operation of the neural network. This architecture is very suitable for FPGA. Therefore, we try to use FPGA to run the neural network.

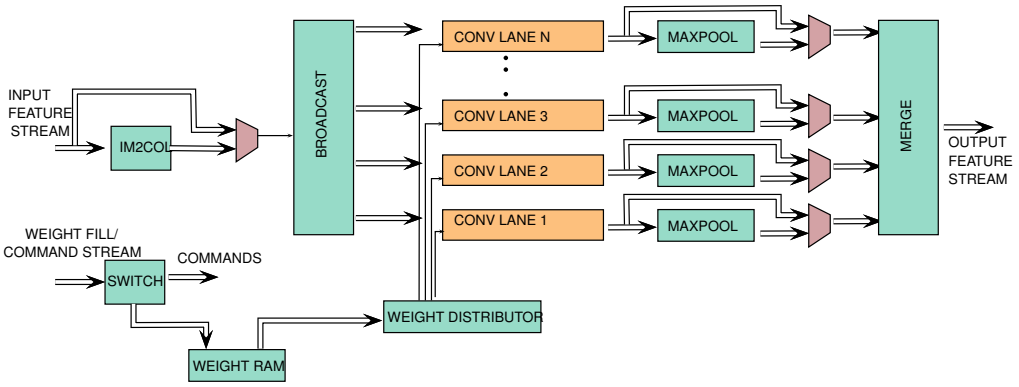

Figure 8.8 – Detailed architecture of one convolutional layer .

Figure 8.8 shows how we deal with each convolutional layer. The input feature map matrix for a convolutional layer is streamed into the accelerator in row major format. There is a configurable image-to-matrix transformation module to reorder the input image, the mechanism of image-to-matrix is shown as figure 8.9. By this module, the convolution is converted into matrices multiplication. Then the matrices generated by image-to-matrix is sent to the conv-lane subsequently. That is a module to calculate the dot product of two vector, which is the fundamental operator for matrix multiplication. In other words, the multiplication-accumulation (MAC) operator in convolutional layer is computed in this module. There are N conv-lane which means N MAC operators can be processed in parallel. The output of matrix multiplication is also the result of convolution. At the end of convolutional layer, a module implementing activation function and a max-pools are followed according to the configuration. Then the convolutional layer is completed and the output of these module is sent to the next layer.

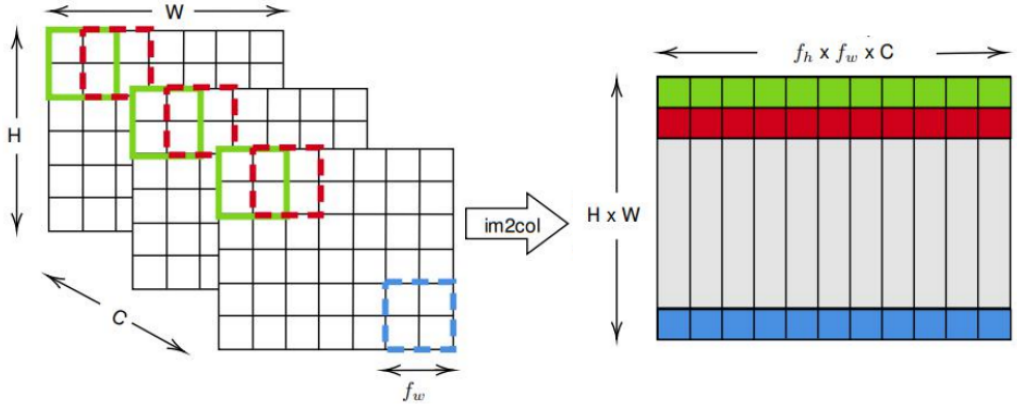

Figure 8.9 – Image to Matrix Transformation.

8.2.4.2/ EXPERIEMNTS RESULTS

The experiment is conducted in Zynq UltraScale+ ZCU102, and running at 100 MHz. We have built 16 conv-lane for each layer. The conv-lane for first layer in each branch can calculate 3 MAC, while 16 MAC can be calculated in parallel in other layers. That means 48 or 256 multiplication can be processed in parallel for convolutional layers. The utilization of resources is shown in Table 8.1:

Table 8.1 – The utilization rate of FPGA resources.

Name	BRAM_18K	DSP48E	FF	LUT	URAM
DSP	-	-	-	-	-
Expression	-	-	0	156	-
FIFO	-	-	-	-	-
Instance	1235	2244	47577	225910	-
Memory	464	-	0	0	-
Multiplexer	-	-	-	10962	-
Register	-	-	2382	-	-
Total	1699	2244	49959	237028	0
Available	1824	2520	548160	274080	0
Utilization(%)	93	89	9	86	0

The main limiting factor is the on-chip memory. Because of this limitation, these three branches share the same memory block for weights and other parameters. Therefore, the branches execute sequential. This caused the implementation of the other two branches idle when one branch was being calculated. This wastes computing resources. We will optimize it in the future work. Even if some branches are idle, this design still achieved the calculation of Multi-feature multi-column convolutional neural network within 0.85s seconds. Compared with CPU(Intel Core i5-5250u (1.60 GHz), 4 GB RAM computer.), it achieves 23.5x speed up and the accuracy of the model will hardly be reduced.

Table 8.2 – Comparison of experimental data between CPU and FPGA.

Detection algorithm	Detection rate	Time
CPU	97.2%	20.25s
FPGA	96.6%	0.85s

Compared with Gasparini and Ahmad, They proposed an integrated camera processor node through which people are detected and counted [1][21]. Most of the existing algorithms only detect and count a single target person. On the contrary, our network (M-MCNN) architecture can detect and count not only a single target person, but also high-density people. At the same time, we have also achieved good experimental results.

8.2.5/ CONCLUSION

In this chapter, we implement the multi-column convolutional neural network part of the M-MCNN high-density crowd counting algorithm. Comparing the performance of M-MCNN on CPU, FPGA: Compared with CPU, FPGA has a good running energy efficiency ratio. The FPGA is more flexible from a design perspective. As long as the internal logic structure is defined by Verilog or other description languages, the hardware accelerator function can be realized.

V

Conclusion, Perspectives

9

CONCLUSION, PERSPECTIVES

9.1/ CONCLUSION

The rapid development of FPGA-based virtual hardware accelerators facilitates neural network operations. We propose a real-time reconfigurable embedded architecture to implement a multi-column multi-feature convolutional neural network crowd counting algorithm. FPGAs provide higher computing performance and lower power consumption.

Artificial neural networks are an important component of artificial intelligence. With the advent of the age of artificial intelligence, more and more enterprises and scientific research units have invested in the research and development of intelligent products and tools. This helps companies and individuals by making convenient and efficient tools available. Neural networks have been widely used for their advantages of high speed and high accuracy. For example, continuous breakthroughs have been made in applications such as pattern recognition, information processing, and robotics. At present, the implementation of neural networks mostly uses traditional general-purpose computers. The traditional general-purpose computer uses a serial computing method. This processing method prohibits efficent operations for neural networks. Therefore, it is important to adopt a better method to implement neural network. And because of FPGA (Field programmable gate array) being hardware programmable, highly parallel computing characteristics, FPGA can effectively make up for the serial limitation of neural networks, and can provide a new way for the realization of artificial neural networks. The method of implementing neural networks through FPGA hardware is of great significance in solving crowd counting.

In this thesis we proposed a combination of software and hardware to achieve crowd counting:

In the first part, we use multi-feature fusion technology. The main purpose is to find head features. We extract image features from multiple information sources. And find the head features through texture feature analysis and crowd image edge detection. First, we connect the HOG function vector and LBP feature vector in series to form a feature vector through texture features. Then the feature vector is inputted into the SVM classifier. Here, in the classification process, linearly separable low-dimensional space is transformed into linearly separable high-dimensional space, passed the SVM kernel function and used cross-validation. The method of selecting the best parameters of the SVM so that the classifier has the highest input classification accuracy training sample. Next, we optimized the original canny operator to suppress false edges caused by noise, make target edges

thinner, and better obtain target edge features. Multi-feature fusion technology obtains clear head contours by extracting target edge detection features and analyzing texture features.

In the second part, we proposed a new framework M-MCNN based on multi-column multi-feature convolutional neural network for crowd counting on any single image. M-MCNN not only contains the original three columns of convolutional neural networks with different filter sizes, but also uses 1*1 filter convolution layers instead of fully connected layers. Therefore, the input image of the model can be of any size, avoiding distortion. And in a single individual sample, we use the first part to extract the texture features of a single human head, and detect the edge features of the head, which greatly improves the learning ability of the sample features. At the same time, the loss of details of the density map is also reduced, and it is better integrated with our convolutional neural network. We performed a lot of experiments on ShanghaiTech, USCD, WorldExpo'10 and GCC datasets. Our model M-MCNN outperforms the latest crowd counting methods on all data sets used for evaluation.

In the third part, we implemented the neural network architecture through FPGA hardware. It is mainly designed and implements the key parts of the hardware implementation of neural networks. The process of implementing the hidden layer is described in detail, and the calculation process of each stage of the network is analyzed and simulated. From the analysis of the experimental results, we can build a system that can well implement the function of crowd counting. And has higher accuracy and stability, so as to achieve the experimental purpose.

9.2/ PERSPECTIVES

In the past ten years, artificial intelligence has reached a stage of rapid development. Deep learning has played an important role in its development. In spite of its powerful simulation prediction capabilities, deep learning still faces the problem of huge computational load. At the hardware level, GPUs, ASICs, and FPGAs are all solutions to the huge computational load. Real-time reconfigurable embedded architecture is implemented using FPGAs for counting people in densely crowded areas. We also encountered a number of challenges.

Software aspect: we propose a multi-feature multi-column convolutional neural network that can count the number of people. We extract feature maps from different layers and simultaneously resize them to have the same output dimension. After that, we fuse features so that the fused features can be used to generate the density map. We also used texture features and target edge detection to reduce the loss of density map detail to better integrate with our convolutional neural network. At the same time, the problem is also coming. The crowd high-density image we obtained from the M-MCNN structure consumes a long time. There is no doubt that the M-MCNN framework needs to be improved in terms of time and detection rate.

Hardware aspect: we implement the M-MCNN crowd counting algorithm through FPGA. The main limiting factor is the on-chip memory. Because of this limitation, these three branches share the same memory block for weights and other parameters. Therefore, the branches execute sequential. This caused the implementation of the other two branches idle when one branch was being calculated. This wastes computing resources. We will

optimize it in the future work.

In future work, we will further optimize the convolutional neural network crowd counting algorithm. And transplant it in drone to realize its function. It includes crowd counting, crowd detection and so on.

BIBLIOGRAPHY

[1] AHMAD, I., ISLAM, Z. U., ULLAH, F., HUSSAIN, M. A., AND NABI, S. An fpga based approach for people counting using image processing techniques. In *2019 11th International Conference on Knowledge and Smart Technology (KST)* (2019), IEEE, pp. 148–152.

[2] AHONEN, T., HADID, A., AND PIETIKÄINEN, M. Face recognition with local binary patterns. In *European conference on computer vision* (2004), Springer, pp. 469–481.

[3] AKINLAR, C., AND CHOME, E. Cannysr: Using smart routing of edge drawing to convert canny binary edge maps to edge segments. In *2015 International Symposium on Innovations in Intelligent SysTems and Applications (INISTA)* (2015), IEEE, pp. 1–6.

[4] BANSAL, ANKAN AND VENKATESH, KS. People counting in high density crowds from still images. *arXiv preprint arXiv:1507.08445* (2015).

[5] BELHASSEN, H., FRESSE, V., AND BOURENNANE, E.-B. Comparative study of face and person detection algorithms: Case study of tramway in lyon. In *2019 International Conference on Advanced Systems and Emergent Technologies (IC_ASET)* (2019), IEEE, pp. 154–159.

[6] BUTENUTH, M., BURKERT, F., SCHMIDT, F., HINZ, S., HARTMANN, D., KNEIDL, A., BORRMANN, A., AND SIRMACEK, B. Integrating pedestrian simulation, tracking and event detection for crowd analysis. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)* (2011), IEEE, pp. 150–157.

[7] CHANG-YEON, J. Face detection using lbp features. *Final Project Report 77* (2008), 1–4.

[8] CHEN, L., WU, H., ZHAO, S., AND GU, J. Head-shoulder detection using joint hog features for people counting and video surveillance in library. In *2014 IEEE Workshop on Electronics, Computer and Applications* (2014), IEEE, pp. 429–432.

[9] CHIKKERUR, S. Cuda implementation of a biologically inspired object recognition system, 2008.

[10] CONTE, D., FOGGIA, P., PERCANNELLA, G., TUFANO, F., AND VENTO, M. A method for counting people in crowded scenes. In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance* (Aug 2010), pp. 225–232.

[11] CONTE, D., FOGGIA, P., PERCANNELLA, G., AND VENTO, M. A method based on the indirect approach for counting people in crowded scenes. In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance* (2010), IEEE, pp. 111–118.

[12] DALAL, N., AND TRIGGS, B. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)* (2005), vol. 1, IEEE, pp. 886–893.

[13] FERMI, D., FNB, R. R., AND KARTHA, S. As," a survey on different face detection algorithms in image processing,". *International Journal of Innovative Research in Science, Engineering and Technology 6* (2017), 151–156.

[14] FRADI, H., AND DUGELAY, J.-L. Low level crowd analysis using frame-wise normalized feature for people counting. In *2012 IEEE International Workshop on Information Forensics and Security (WIFS)* (2012), IEEE, pp. 246–251.

[15] FRADI, H., AND DUGELAY, J.-L. Crowd density map estimation based on feature tracks. In *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)* (2013), IEEE, pp. 040–045.

[16] FUKUSHIMA, K. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics 36*, 4 (1980), 193–202.

[17] GAO, J., WANG, Q., AND LI, X. Pcc net: Perspective crowd counting via spatial convolutional network. *IEEE Transactions on Circuits and Systems for Video Technology* (2019).

[18] GAO, J., WANG, Q., AND YUAN, Y. Scar: Spatial-/channel-wise attention regression networks for crowd counting. *Neurocomputing 363* (2019), 1–8.

[19] GARDEL, A., BRAVO, I., JIMENEZ, P., JL, L., AND TORQUEMADA, A. Real time head detection for embedded vision modules. In *2007 IEEE International Symposium on Intelligent Signal Processing* (2007), IEEE, pp. 1–6.

[20] GASPARINI, L., GOTTARDI, M., MASSARI, N., PETRI, D., AND MANDUCHI, R. Fpga implementation of a people counter for an ultra-low-power wireless camera network node. In *2011 7th Conference on Ph. D. Research in Microelectronics and Electronics* (2011), IEEE, pp. 169–172.

[21] GASPARINI, L., MANDUCHI, R., AND GOTTARDI, M. An ultra-low-power contrast-based integrated camera node and its application as a people counter. In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance* (2010), IEEE, pp. 547–554.

[22] GONZALEZ, R., AND WINTZ, P. Digital image processing, advanced book program. *World Science Division, Addison Wesley* (1977).

[23] GÜNDÜZ, A. E., TEMIZEL, T. T., AND TEMIZEL, A. Kalabalik videolarinda yoğunluk kestirimi density estimation in crowd videos.

[24] HADID, A., PIETIKAINEN, M., AND AHONEN, T. A discriminative feature space for detecting and recognizing faces. In *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* (2004), vol. 2, IEEE, pp. II–II.

[25] HAN, K., WAN, W., YAO, H., AND HOU, L. Image crowd counting using convolutional neural network and markov random field. *Journal of Advanced Computational Intelligence and Intelligent Informatics 21*, 4 (2017), 632–638.

[26] HAO, G., MIN, L., AND FENG, H. Improved self-adaptive edge detection method based on canny. In *2013 5th International Conference on Intelligent Human-Machine Systems and Cybernetics* (2013), vol. 2, IEEE, pp. 527–530.

[27] HE, K., ZHANG, X., REN, S., AND SUN, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 770–778.

[28] HOU, Y.-L., AND PANG, G. K. Human detection in crowded scenes. In *2010 IEEE International Conference on Image Processing* (2010), IEEE, pp. 721–724.

[29] HSU, W.-L., LIN, K.-F., AND TSAI, C.-L. Crowd density estimation based on frequency analysis. In *2011 Seventh International Conference on Intelligent Information Hiding and Multimedia Signal Processing* (2011), IEEE, pp. 348–351.

[30] HU, Y., CHANG, H., NIAN, F., WANG, Y., AND LI, T. Dense crowd counting from still images with convolutional neural networks. *Journal of Visual Communication and Image Representation 38* (2016), 530–539.

[31] HUANG, D., SHAN, C., ARDABILIAN, M., WANG, Y., AND CHEN, L. Local binary patterns and its application to facial image analysis: a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 41*, 6 (2011), 765–781.

[32] HUI.L, AND CHUANG.Z. Human head detection method based on head feature. *Optoelectronic Technology 34*, 1 (2014), 21.

[33] IDREES, H., SALEEMI, I., SEIBERT, C., AND SHAH, M. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2013), pp. 2547–2554.

[34] IERUSALIMSCHY, R., DE FIGUEIREDO, L. H., AND CELES, W. Lua 5.1 reference manual, 2006.

[35] KEJU.Z, FUQIANG.L, Z. Counting people based on multiple targets detection and tracking. *Journal of Image and Graphics. 14*, 10 (2009), 2106–2109.

[36] KIM, D., LEE, Y., KU, B., AND KO, H. Crowd density estimation using multi-class adaboost. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance* (2012), IEEE, pp. 447–451.

[37] KIM, J.-Y., KIM, M., LEE, S., OH, J., KIM, K., AND YOO, H.-J. A 201.4 gops 496 mw real-time multi-object recognition processor with bio-inspired neural perception engine. *IEEE Journal of Solid-State Circuits 45*, 1 (2009), 32–45.

[38] KRISHNAMOORTHI, R. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342* (2018).

[39] KRYJAK, T., KOMORKIEWICZ, M., AND GORGON, M. Fpga implementation of real-time head-shoulder detection using local binary patterns, svm and foreground object detection. In *Proceedings of the 2012 Conference on Design and Architectures for Signal and Image Processing* (2012), IEEE, pp. 1–8.

[40] LE, T.-S., AND HUYNH, C.-K. Human-crowd density estimation based on gabor filter and cell division. In *2015 International Conference on Advanced Computing and Applications (ACOMP)* (2015), IEEE, pp. 157–161.

[41] LI, M., ZHANG, Z., HUANG, K., AND TAN, T. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. In *2008 19th International Conference on Pattern Recognition* (2008), IEEE, pp. 1–4.

[42] LI, Y., ZHANG, X., AND CHEN, D. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 1091–1100.

[43] LIU, W., SALZMANN, M., AND FUA, P. Context-aware crowd counting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 5099–5108.

[44] LIU, X., TU, P. H., RITTSCHER, J., PERERA, A., AND KRAHNSTOEVER, N. Detecting and counting people in surveillance applications. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.* (2005), IEEE, pp. 306–311.

[45] MA, R., LI, L., HUANG, W., AND TIAN, Q. On pixel count based crowd density estimation for visual surveillance. In *IEEE Conference on Cybernetics and Intelligent Systems, 2004.* (2004), vol. 1, IEEE, pp. 170–173.

[46] MA, Z., AND CHAN, A. B. Crossing the line: Crowd counting by integer programming with local features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 2539–2546.

[47] MANDHALA, V. N., BHATTACHARYYA, D., AND KIM, T.-H. Face detection using image morphology–a review. *International Journal of Security and Its Applications 10*, 4 (2016), 89–94.

[48] MARANA, A. N., VELASTIN, S., COSTA, L., AND LOTUFO, R. *Estimation of crowd density using image processing*. IET, 1997.

[49] MATHIAS, M., BENENSON, R., PEDERSOLI, M., AND VAN GOOL, L. Face detection without bells and whistles. In *European conference on computer vision* (2014), Springer, pp. 720–735.

[50] MEGALINGAM, R. K., AND PILLAI, M. Fpga based wheelchair autonavigation for people with mobility issues. In *2015 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE)* (2015), IEEE, pp. 35–38.

[51] MEYNBERG, O., CUI, S., AND REINARTZ, P. Detection of high-density crowds in aerial images using texture classification. *Remote Sensing 8*, 6 (2016), 470.

[52] OJALA, T., PIETIKAINEN, M., AND HARWOOD, D. Performance evaluation of texture measures with classification based on kullback discrimination of distributions. In *Proceedings of 12th International Conference on Pattern Recognition* (1994), vol. 1, IEEE, pp. 582–585.

[53] PAPAGEORGIOU, C. P., OREN, M., AND POGGIO, T. A general framework for object detection. In *Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271)* (1998), IEEE, pp. 555–562.

[54] PENG, D., SUN, Z., CHEN, Z., CAI, Z., XIE, L., AND JIN, L. Detecting heads using feature refine net and cascaded multi-scale architecture. In *2018 24th International Conference on Pattern Recognition (ICPR)* (2018), IEEE, pp. 2528–2533.

[55] QIANG, S., GUOYING, L., JINGQI, M., AND HONGMEI, Z. An edge-detection method based on adaptive canny algorithm and iterative segmentation threshold. In *2016 2nd International Conference on Control Science and Systems Engineering (ICCSSE)* (2016), IEEE, pp. 64–67.

[56] RANJAN, R., PATEL, V. M., AND CHELLAPPA, R. Hyperface: A deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence 41*, 1 (2017), 121–135.

[57] REKHA, N., AND KURIAN, M. Face detection in real time based on hog. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 3*, 4 (2014), 1345–1352.

[58] RODRIGUEZ, M., LAPTEV, I., SIVIC, J., AND AUDIBERT, J.-Y. Density-aware person detection and tracking in crowds. In *2011 International Conference on Computer Vision* (2011), IEEE, pp. 2423–2430.

[59] RODRIGUEZ, M. D., AND SHAH, M. Detecting and segmenting humans in crowded scenes. In *Proceedings of the 15th ACM international conference on Multimedia* (2007), pp. 353–356.

[60] SABZMEYDANI, P., AND MORI, G. Detecting pedestrians by learning shapelet features. In *2007 IEEE Conference on Computer Vision and Pattern Recognition* (2007), IEEE, pp. 1–8.

[61] SAKATA, H. In the early 1960s hubel and wiesel (5) found simple cells and complex. *Strategy and Prospects in Neuroscience 10* (1987), 217.

[62] SAM, D. B., SURYA, S., AND BABU, R. V. Switching convolutional neural network for crowd counting. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), IEEE, pp. 4031–4039.

[63] SCHWEBER, S. S. The empiricist temper regnant: Theoretical physics in the united states 1920-1950. *Historical Studies in the Physical and Biological Sciences 17*, 1 (1986), 55–98.

[64] SHANMUGAVADIVU, P., AND KUMAR, A. Modified eight-directional canny for robust edge detection. In *2014 International Conference on Contemporary Computing and Informatics (IC3I)* (2014), IEEE, pp. 751–756.

[65] SHAO, J., CHANGE LOY, C., AND WANG, X. Scene-independent group profiling in crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 2219–2226.

[66] SINDAGI, V. A., AND PATEL, V. M. Cnn-based cascaded multi-task learning of high-level prior and density estimation for crowd counting. In *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)* (2017), IEEE, pp. 1–6.

[67] SINDAGI, V. A., AND PATEL, V. M. A survey of recent advances in cnn-based single image crowd counting and density estimation. *Pattern Recognition Letters 107* (2018), 3–16.

[68] SOOMRO, P. N., MEMON, U., AND MEMON, S. Human detection and counting in crowded scene. *Asian Journal of Engineering, Sciences & Technology 4*, 1 (2014).

[69] SUBBURAMAN, V. B., DESCAMPS, A., AND CARINCOTTE, C. Counting people in the crowd using a generic head detector. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance* (2012), IEEE, pp. 470–475.

[70] VAPNIK, V., GUYON, I., AND HASTIE, T. Support vector machines. *Mach. Learn 20*, 3 (1995), 273–297.

[71] VICENTE, A. G., MUNOZ, I. B., MOLINA, P. J., AND GALILEA, J. L. L. Embedded vision modules for tracking and counting people. *IEEE Transactions on Instrumentation and Measurement 58*, 9 (2009), 3004–3011.

[72] VIOLA, P., AND JONES, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001* (2001), vol. 1, IEEE, pp. I–I.

[73] VORA, A., AND CHILAKA, V. Fchd: Fast and accurate head detection in crowded scenes. *arXiv preprint arXiv:1809.08766* (2018).

[74] WANG, C., ZHANG, H., YANG, L., LIU, S., AND CAO, X. Deep people counting in extremely dense crowds. In *Proceedings of the 23rd ACM International Conference on Multimedia* (New York, NY, USA, 2015), MM '15, Association for Computing Machinery, p. 1299–1302.

[75] WANG, L., AND YUNG, N. H. Bayesian 3d model based human detection in crowded scenes using efficient optimization. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)* (2011), IEEE, pp. 557–563.

[76] WANG, Q., GAO, J., LIN, W., AND LI, X. Nwpu-crowd: A large-scale benchmark for crowd counting. *arXiv preprint arXiv:2001.03360* (2020).

[77] WANG, Q., GAO, J., LIN, W., AND YUAN, Y. Learning from synthetic data for crowd counting in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2019), pp. 8198–8207.

[78] WANG, Z., BOVIK, A. C., SHEIKH, H. R., AND SIMONCELLI, E. P. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing 13*, 4 (2004), 600–612.

[79] WANG, Z., LIU, H., QIAN, Y., AND XU, T. Crowd density estimation based on local binary pattern co-occurrence matrix. In *2012 IEEE International Conference on Multimedia and Expo Workshops* (2012), IEEE, pp. 372–377.

[80] WU, X., LIANG, G., LEE, K. K., AND XU, Y. Crowd density estimation using texture analysis and learning. In *2006 IEEE international conference on robotics and biomimetics* (2006), IEEE, pp. 214–219.

[81] XU, B., AND QIU, G. Crowd density estimation based on rich features and random projection forest. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2016), IEEE, pp. 1–8.

[82] YANG, D. B., GUIBAS, L. J., ET AL. Counting people in crowds with a real-time network of simple image sensors. In *null* (2003), IEEE, p. 122.

[83] YANG, H., AND ZHAO, H.-A. A novel method for crowd density estimations. In *IET International Conference on Information Science and Control Engineering 2012 (ICISCE 2012)* (2012), IET, pp. 1–4.

[84] YIN, J. H., VELASTIN, S. A., AND DAVIES, A. C. Image processing techniques for crowd density estimation using a reference image. In *Asian Conference on Computer Vision* (1995), Springer, pp. 489–498.

[85] YUAN, L., AND XU, X. Adaptive image edge detection algorithm based on canny operator. In *2015 4th International Conference on Advanced Information Technology and Sensor Application (AITS)* (2015), IEEE, pp. 28–31.

[86] ZHANG, C., LI, H., WANG, X., AND YANG, X. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2015), pp. 833–841.

[87] ZHANG, L., SHI, M., AND CHEN, Q. Crowd counting via scale-adaptive convolutional neural network. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2018), IEEE, pp. 1113–1121.

[88] ZHANG, X., LI, Y., HAO, C., RUPNOW, K., XIONG, J., HWU, W.-M., AND CHEN, D. Skynet: A champion model for dac-sdc on low power object detection. *arXiv preprint arXiv:1906.10327* (2019).

[89] ZHANG, Y., ZHOU, C., CHANG, F., KOT, A. C., AND ZHANG, W. Attention to head locations for crowd counting. In *International Conference on Image and Graphics* (2019), Springer, pp. 727–737.

[90] ZHANG, Y., ZHOU, D., CHEN, S., GAO, S., AND MA, Y. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2016), pp. 589–597.

[91] ZHANG, Z., YIN, W., AND VENETIANER, P. L. Fast crowd density estimation in surveillance videos without training. In *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance* (2012), IEEE, pp. 452–457.

[92] ZHU, J.-Q., AND CAI, C.-H. Real-time face detection using gentle adaboost algorithm and nesting cascade structure. In *2012 International Symposium on Intelligent Signal Processing and Communications Systems* (2012), IEEE, pp. 33–37.

List of Figures

List of Tables

Document generated with LaTeX and:
the LaTeX style for PhD Thesis created by S. Galland — http://www.multiagent.fr/ThesisStyle
the tex-upmethodology package suite — http://www.arakhne.org/tex-upmethodology/