Deep Learning and its application

Submitted by: Soreefan Mohammad Jaleel
Matriculation Number: U1522684L

Supervisor: Prof. Tan Yap Peng

School of Electrical & Electronic Engineering

A final year project report presented to the Nanyang Technological University
in partial fulfilment of the requirements of the degree of
Bachelor of Engineering

2019

Contents

Abstract

The applications of machine learning are numerous and are growing continuously. In this project, deep learning, a type of machine learning, will be used to construct a couple of these applications. These include crowd counting and path tracking from a live video feed. A convolution neural network will be trained and tested on a dataset such that it can detect people from the video feed. The network will then be applied to real-life cases such that the above-mentioned functions can be performed accurately.

Acknowledgement

I would like to thank Professor Tan Yap Peng for his help and advice throughout the entire project. I would also like to thank Ma Junjie for his help in preparing the datasets used during the project. The datasets were essential in the training and testing of the neural network.

Introduction

Machine learning is one of the main forces that drives modern technology. Deep learning is one aspect of machine learning. Over the past decades, machine learning technologies have been constantly improving [1]. The improvement in technologies, namely in CPUs and GPUs, have made the implementation of deep learning networks easier and quicker.

Projects that made use of the concept of deep learning are numerous. Many of them were used for surveillance purposes such as person re-identification [2], pedestrian detection [3] [4], tracking [5] and crowd segmentation. These projects benefited hugely because of the power of deep models.

Object tracking forms part of computer vision and has a long story spanning decades.
decades [6]. Following the rise of deep learning in computer vision, data-driven learning methods are being used for object tracking.

To the best of my knowledge, the closest application of deep learning to crowd counting is by using the density map to estimate the number of people in a crowd [7].Using this method, the crowd count is evaluated. However, the crowd cannot be tracked. The method proposed in this project is to track the individuals in a crowd first. After that, the crowd count is obtained.

This project will focus on applying the knowledge of deep learning to crowd tracking and counting. Some materials used during this project will be very similar to those used in previous projects particularly the datasets since the datasets required to train the designed algorithm are not readily available [8].

The findings from the research will provide a deeper understanding into the application of deep learning in object tracking and counting. The application of the algorithm in some real-life cases will also be discussed in detail.

Neural networks

Neural networks are the basis of every deep learning algorithms. They attempt to mimic the human neural system to solve complex problems.

The human neural system includes the brain, the spinal cord and cells called neurons. Neurons are connected by axons and dendrites. One of the functions of the neural system is to carry impulses around the body to achieve specific functions which include sensory and motor functions. Upon repeated simulation of a neuron by the same impulse, the synaptic connections between neurons are changed by the brain. This process is commonly known as learning. Throughout their lives, human beings learn by continuously modifying neural connections using sensory experiences.

A neural network tries to imitate the human neural network. It consists of units called perceptrons. These units are similar to neurons and are connected together by links which have an associated weight and bias. Like humans, neural networks try to learn by changing these weights and biases. This is done so by subjecting the neural network to datasets and improving the weights to attain the best result.

A neural network may consist of millions of perceptrons. They are arranged in layers; the first one being the input and the last one being the output. All the layers in between the input and the output are called hidden layers. Figure 1. shows an example of a neural network which consists of 2 hidden layers [9].

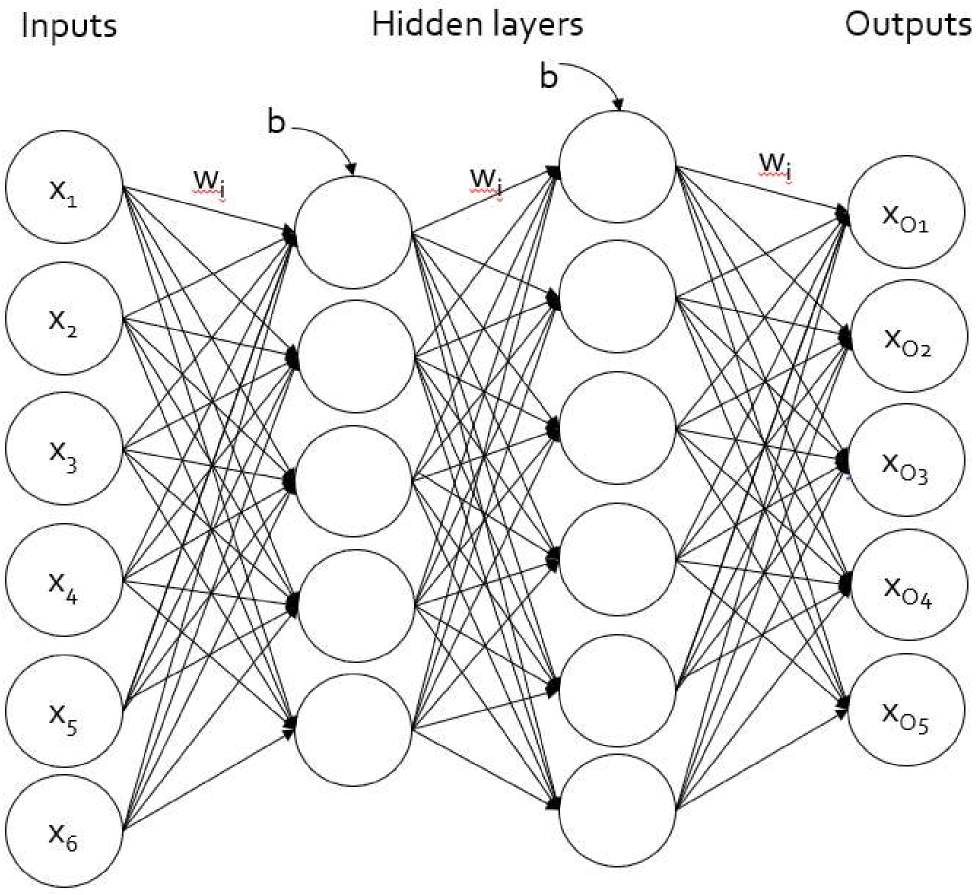


Figure 1. Example of a neural network

Each perceptron inside a neural network, except the input layer which has only outputs, may have several inputs and outputs. Figure 2. shows a perceptron with 5 inputs and 1 output.

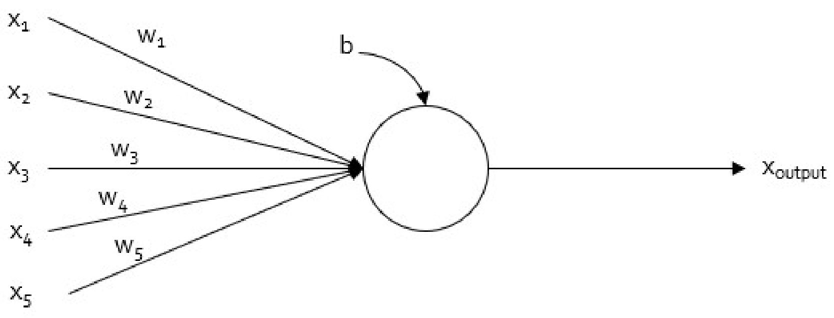


Figure 2. Example of a perceptron

Each input consists of the output of a previous perceptron, an associated weight and a bias. It uses these along with an activation function to get the output of the perceptron. This output is then sent to the subsequent perceptron in the neural network.

$$x_{output} = f(\sum_i (x_i . w_i) + b) \text{ [10]}$$

Activation function

As indicated above, the neural network uses an activation function to produce the output of the perceptron. The activation function is used to define the output. Some commonly used activation functions are the sigmoid function, the tanh function and the rectified linear unit (ReLU) function.

Even though one may think that non-linear functions are better suited as activation function during machine learning, some research show otherwise. In fact, the ReLU function has greater or equal quality [11]. Therefore, the activation function that will be used during the project is the ReLU function. The graph of the function is shown in figure. 3.

$$f(x) = \begin{cases} x, & x < 0 \\ 0, & x \geq 0 \end{cases}$$

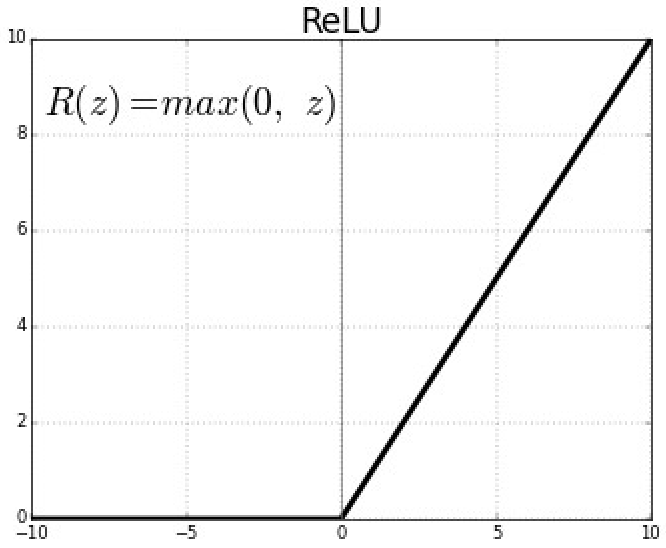


Figure. 3 Graph of ReLU function

Backpropagation algorithm

The backpropagation algorithm is the way by which the neural network learns. Through this algorithm, the weights and biases of the network are changed. At first, the weights and biases of the untrained network are randomized. The next step that the algorithm performs is to calculate the cost function. The cost function is based on the difference between the predicted output and the actual output. One of the most common cost functions is the mean squared error. It is defined as follows:

$$E = \frac{1}{N}\sum_{k=1}^{N}(t_k - o_k)^2$$

Where t_k: target output

o_k: output of system

The weights and biases of the network are modified in such a way that the cost function is minimized [10].

The change in weights, Δw, and biases, Δb, are calculated as follows:

$$\Delta w = -\eta_w \frac{\partial E}{\partial w}$$

$$\Delta b = -\eta_b \frac{\partial E}{\partial b}$$

Where η_w and η_b are the learning rates of the network. The learning rates are used to define the steps by which the weights and biases change. A large learning rate creates a fast way to find the minimum of the cost function but could lead to going past the minimum. Appropriate learning rates must be chosen to prevent overcorrecting the values.

A neural network consists of thousands of weights and biases. The values of the weights and biases are stored in a matrix and the change, Δw and Δb, is also stored in a corresponding matrix.

$$w_{final} = w_{initial} + \Delta w$$

$$\begin{bmatrix} w_{f1} \\ w_{f2} \\ \vdots \\ w_{fn} \end{bmatrix} = \begin{bmatrix} w_{i1} \\ w_{i2} \\ \vdots \\ w_{in} \end{bmatrix} + \begin{bmatrix} \Delta w_1 \\ \Delta w_2 \\ \vdots \\ \Delta w_n \end{bmatrix}$$

$$b_{final} = b_{initial} + \Delta b$$

There are many ways in which the neural network can implement the changes in weights and biases:

- Epoch updating

  In this method, the network updates the weights and biases only after all the samples in the dataset have been presented. Although this method will be very accurate, it is computationally slow since the whole training set must be passed through the algorithm.

- Stochastic updating

  This is a less computationally slow method that will be used in this project. Instead of passing the whole training set through the algorithm, the dataset is divided into several batches which are randomly chosen from the dataset. The weights and biases are updated after passing each batch through the algorithm and the process is repeated until all the batches are completed.

Convolution neural network

Convolution neural networks are mainly used in image processing. Typical neural networks are not efficient for images as it consists of a lot of pixels. If a conventional neural network is used, the number of weights and biases will be too big and thus, it will be computationally very slow and may lead to overfitting.

To speed up the process and reduce the number of variables, convolution filters, also known as kernels, are applied to the image. The weights and biases are then associated to the kernels instead of the individual pixels of the image.

Moreover, because neighboring pixels inside an image have very similar values, a down sampling can be performed on the image to reduce its special dimensions. As a result, the number of parameters and computations in the network will be reduced. There are different types of down sampling techniques, namely max pooling, where the pixel with the largest value is picked, and average pooling, where the average of the neighboring pixels is chosen.

A successful application of this type of neural network is the LeNet-5 network which was developed by Yann LeCun in 1998. This network could be used to read zip codes, digits, etc. the network is shown in Figure. 4 [12].

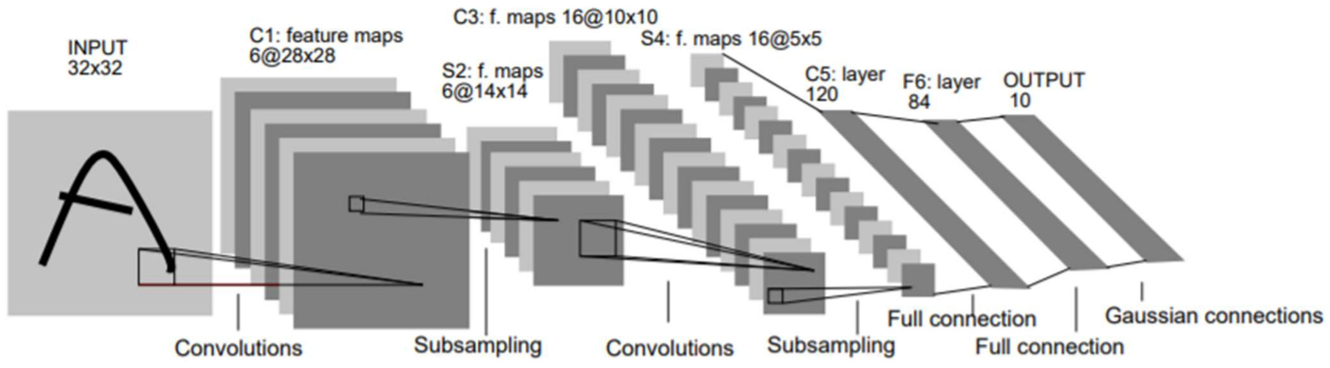
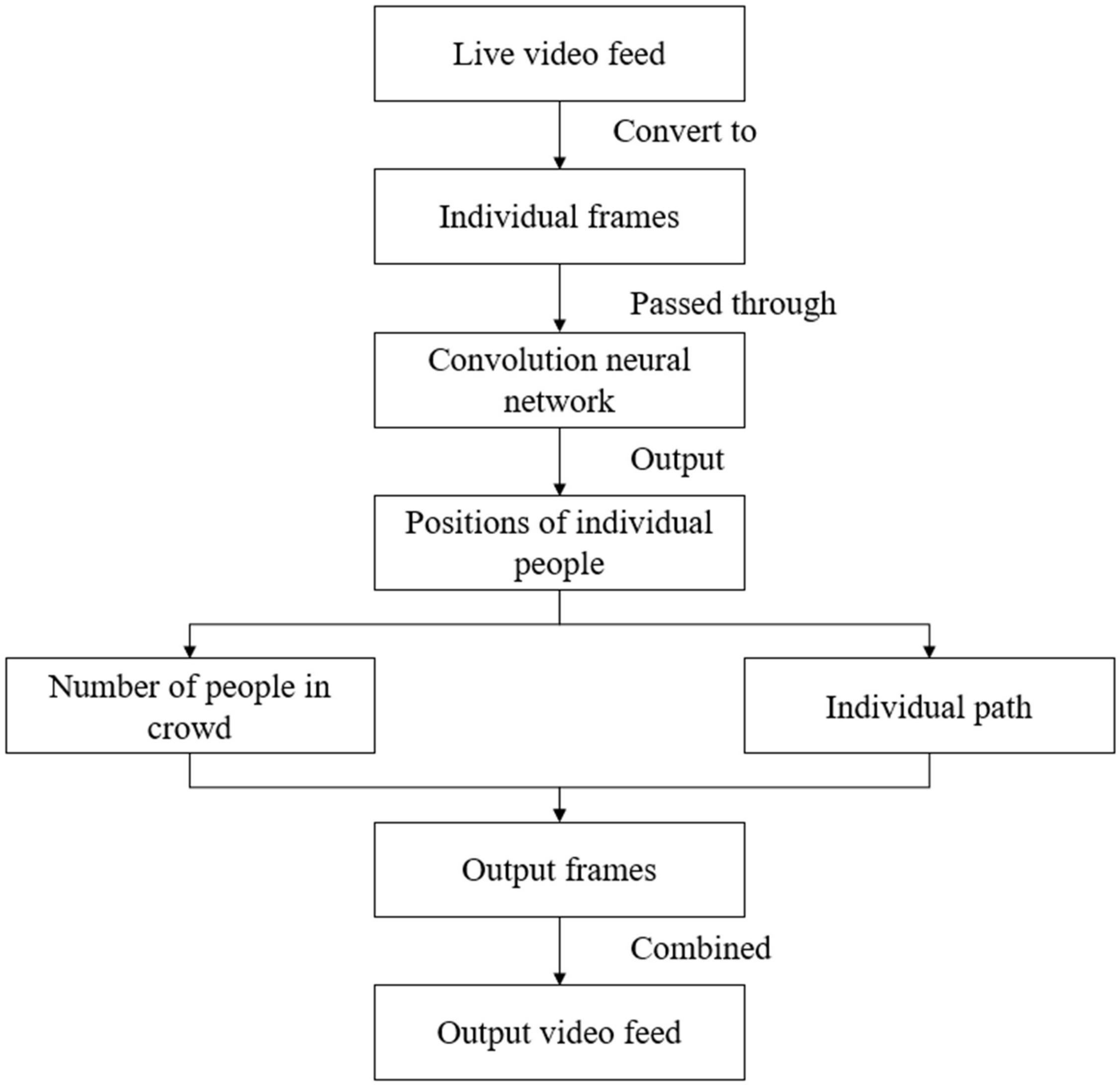


Figure. 4 LeNet-5 Convolution Neural Network

This network consists of 2 convolution layers with down sampling and 3 fully connected layers. Across the 2 convolution layers, kernel sizes of 5x5 were applied and the down sampling technique used was average pooling.

Objective

The main objective of this project is to perform the crowd detection and counting in a live video feed. The video will be separated into frames and each frame will be passed into a neural network to determine the positions of the people inside the frame which will then be used to perform the crowd counting. The resulting output frames will be joined together to form an output video feed. Additionally, a path detection algorithm will be implemented using the output from the neural network.



The aim of the project is to be able to distinguish every people in a crowd and find their position in the picture. The number of people in the image can then be determined. Another objective of the project is to be able to track the paths of the detected people from the video. Successive frames from the video feed will be compared and the overall path of each people will be shown on the output video feed. The locations where the project is expected to work is in mildly crowded places such as canteens or MRT stations.

Tools used

This project is mainly software oriented. Therefore, the main tools that were used were software. The whole project was built using the programming language python. Python is an interpreted, high level programming language. The most useful feature of the python programming language is the availability of libraries for use. In fact, several useful machine learning libraries such as pytorch, tensor flow and keras are available. Moreover, the OpenCV, open source computer vision, library as well as imutils which were used for image processing are also available.

Pytorch

Pytorch is a machine learning library which was used in this project. It was initially released in October 2016 and is primarily developed by Facebook's artificial-intelligence research group. In this project, the pytorch library was used to firstly build the neural network model. Secondly, it was used to convert the training and testing images into tensors which were then inputted into the network. The main computation required for a neural is during back propagation where the weights and biases are adjusted based on a loss function. Pytorch facilitates this process as it provides a specific function which performs this task. It also contains functions designed to calculate the loss function.

OpenCV

OpenCV was designed for computational efficiency and with a strong focus on real-time applications. In this project, OpenCV was used during the post processing phase. After going through the network, the output frames were processed using the OpenCV library to first, remove noise and second, apply some filters to the frames. The positions of the people in the crowd could then be obtained more easily.

Imutils

Imutils consists of a series of functions which make basic image processing easier. The main part in which imutils was used in this project was during the phase where the live video was converted into its individual frames. OpenCV contains a very similar function and was used in the earlier part of the project. However, using OpenCV was too slow and the output video stream could never even reach 1 frame per second. Using imutils enabled faster image processing due to threading and an image queue. It creates a thread separate from that of the python script which is solely responsible for the reading of image frames from the video stream. These frames are then stored in the image queue and are processed accordingly. As a result, the speed at which the network works is much faster when using imutils to retrieve the image frames compared to when using OpenCV.

Datasets

Datasets are very important when dealing with machine learning. They are the way through which the network learns, that is through which the weights and biases of the network are updated. In the project, the datasets that were available were the Shanghai tech dataset, mall dataset and the UCSD dataset.

Shanghai tech dataset

This dataset contains both colored as well as grayscale images. The size of the dataset is fairly good, containing about 1200 images of varying sizes. The images consisted of very dense crowds. The information about the positions of the people in the images is stored in a separate csv file. Each csv file mirrors each pixel of its corresponding image where the pixels containing a person in non-zero. The other pixels are fixed at zero. Figure. 5 shows some example of the images inside the dataset.

Figure. 5 Images from Shanghai tech dataset

Mall dataset

This dataset contains colored images from a mall. The dataset is quite small containing about 2000 images divided into training and testing samples. Each image is of size 640 x 480 pixels and are taken from a CCTV camera. The information about the positions of the people in each image is stored in a similar fashion as those from the Shanghai tech dataset in that the pixels containing people are non-zero and those without are zero. Figure. 6 shows some examples of the images in this dataset.

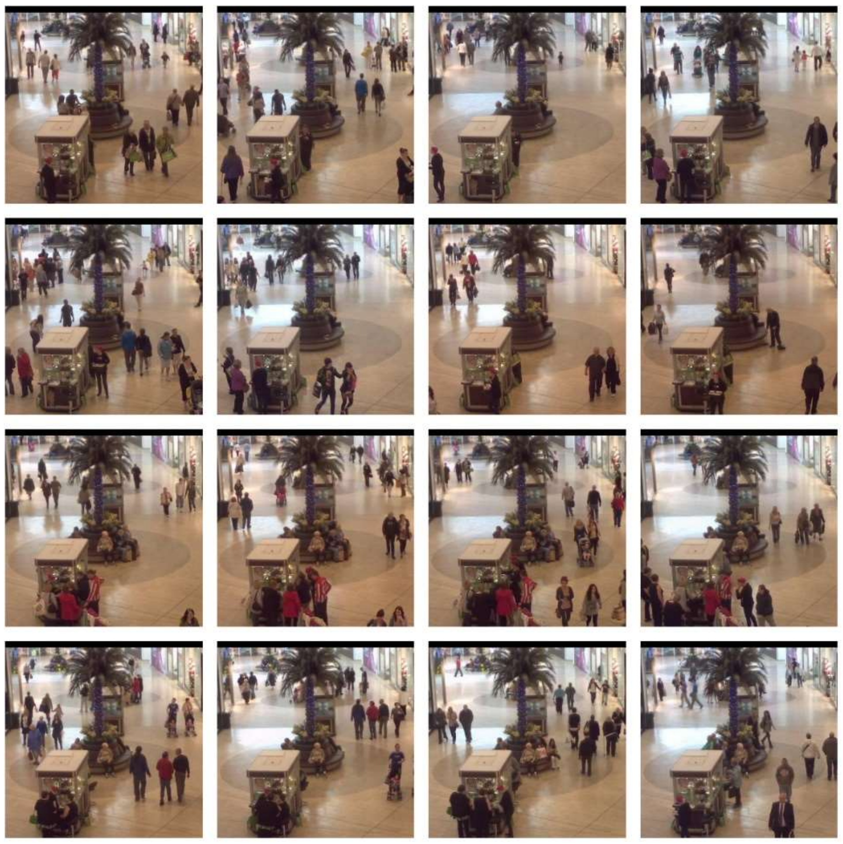


Figure. 6 Images from the mall dataset

UCSD dataset

This dataset consists of grayscale images. Each image consists of people on a walkway. The total number of samples in this dataset is 2000 and each image has a size of 238 x 158 pixels. The information about the positions of people in each image is stored in the same way as the previous two datasets. The information is stored in an array of identical size as the input image. The pixels where an object is present is indicated by a non-zero value and the pixels where an object should not be detected is indicated by zero. Some samples of the dataset are shown in Figure. 7.

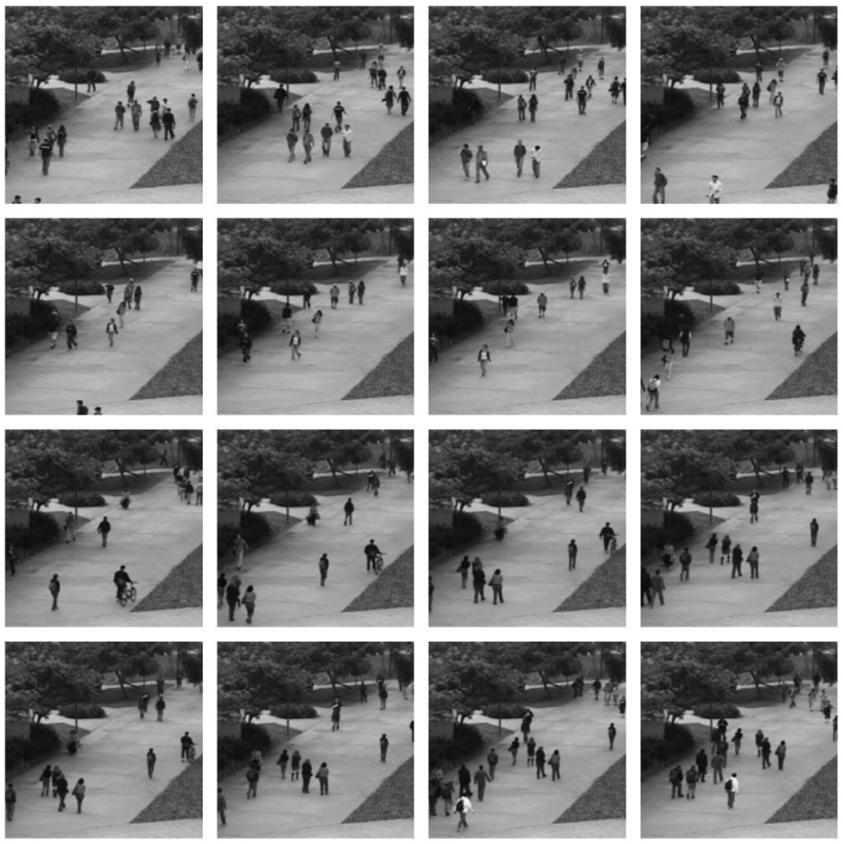


Figure. 7 Images from UCSD dataset

These are the 3 datasets that were used during the training and testing of the neural network. Since one of the desired features of the project was to monitor moderately crowded places, the Shanghai tech datasets was selected. The mall dataset reflected a more accurate representation of the places where the project could be used. The UCSD dataset was also used so that the network could detect objects under different lighting conditions.

The characteristics of the datasets reflect on the desired properties that the final network should possess. The optimal condition is that the final network can perform accurately in all the different conditions. The actual performance of the network will be monitored during the training and testing phase.

Neural network model

The model that was used during the experiments consisted of 3 different convolution networks. Each network is comprised of a sequence of several convolution layers and a couple of down sampling layers.

In this case, the down sampling is done using max pooling with a kernel size of 2. Moreover, contrary to the usual convolution layers during the shape of the image is reduced by (kernel size - 1), the convolution layers in these networks apply a padding to the output so that the original shape of the input is maintained.

The outputs of each convolution network are then combined and up sampled to form the output. The overall output has the same shape as the original input.

Convolution layers will be shown as conv [input channels, output channels, kernel size] and max pooling will be shown as max_pool (kernel size)

All the convolution networks consist of 6 layers, 4 of which are convolution layers and 2 are max pooling layers. However, each branch consists of convolution layers with different input channels, output channels and kernel size.

Branch1 =	conv [3, 16, 9], max_pool (2), conv [16, 32, 7], max_pool (2), conv [32, 16, 7], conv [16, 8, 7]
Branch2 =	conv [3, 20, 7], max_pool (2), conv [20, 40, 5], max_pool (2), conv [40, 20, 5], conv [20, 10, 5]
Branch3 =	conv [3, 24, 5], max_pool (2), conv [24, 48, 3], max_pool (2), conv [48, 24, 3], conv [24, 12, 3]

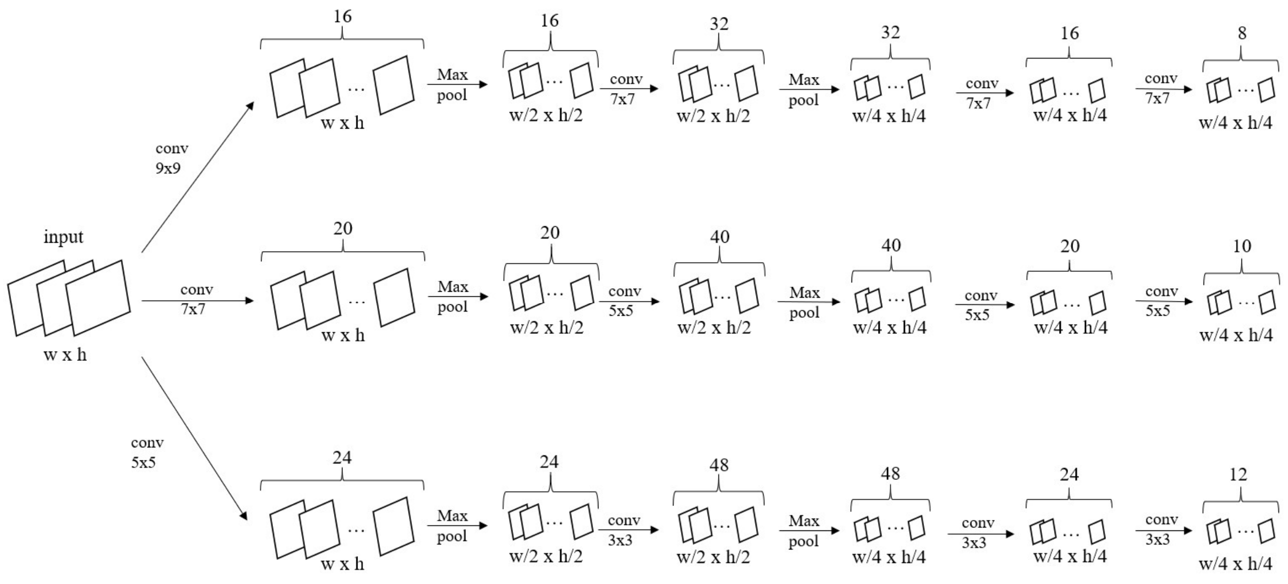


Figure. 8 Overview of convolution layers

The outputs of each convolution networks are then fused together to form a 30 channels output with size w/4 x h/4. These are then converted to a single channel through a convolution layer with a kernel size of 1 x 1. The final output is then obtained by up scaling the latter by a factor 4 so that the final output shape matches that of the input.

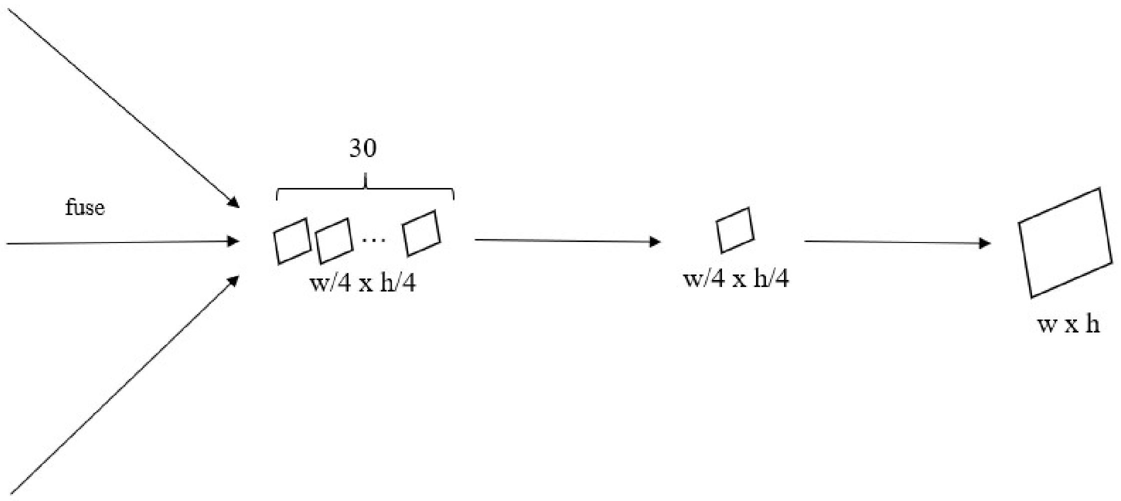


Figure 9. Final output of convolution network

The output, which will be of the same dimensions as the original image, w x h, will consist of zeros where an object is not detected and a non-zero value where an object is detected.

Network training and testing

The neural network was trained by passing the images from the dataset through it. For this project, the epoch updating method was used. As mentioned previously, during epoch updating, the whole dataset is passed through the neural network before back propagation takes place. This method is used instead of the stochastic updating method because firstly, the result obtained is more accurate and secondly, since the datasets are relatively small, the largest consisting only of about 15000 images, the slow computation power is not a major deterrence.

Throughout each epoch, the images found in the datasets are first converted into tensors and then are passed into the network. The cost function is then computed by comparing the output of the neural network and that from the dataset. Based on that, the weights and biases of the network are modified by the backpropagation algorithm.

The cost function used during the project is the mean absolute error which computes the average between the expected outputs and the actual outputs.

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |\, t_i - \, o_i|$$

Where t_i: actual output of the neural network

o_i: expected output of the neural network

The t_i was calculated by summing up all the pixels obtained from the output of the neural network. On the other hand, the o_i was obtained by summing up all the pixels from the dataset. As mentioned previously, the information about the positions of people are stored in an array of identical size as the input image. Since the non-detected regions are indicated as zero values, summing up all the values of a single output and comparing it to the sum of the values from the dataset can be used as a measure of the accuracy of the network.

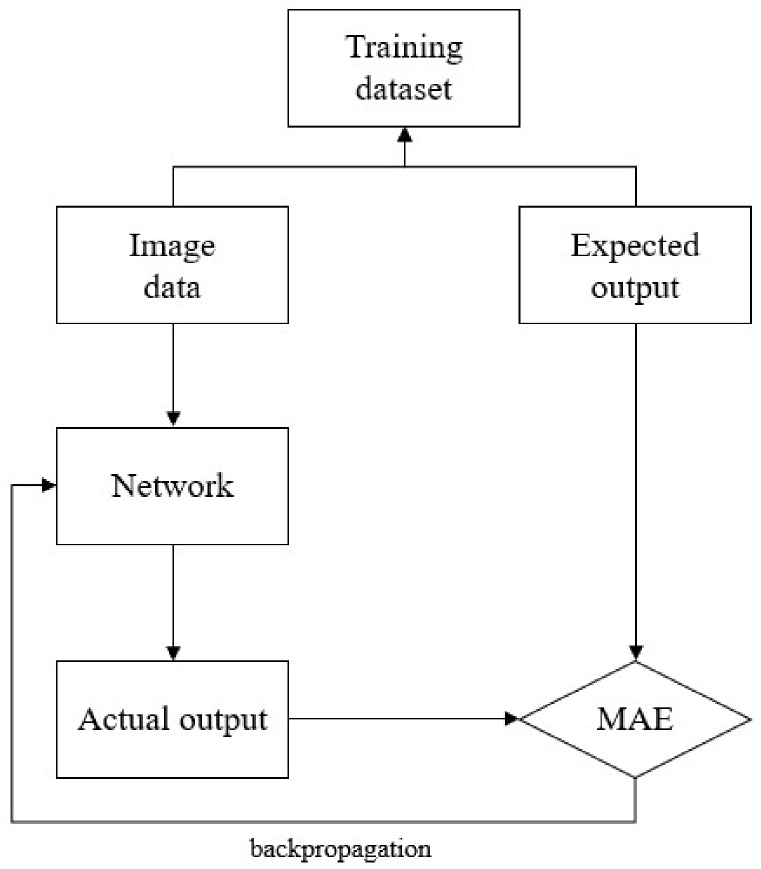


Figure. 10 Training of neural network

Each dataset used was divided into 2 parts: training and testing. The training images were used during the training phase and the testing images were used in the testing phase. The datasets were separated so that the network can be tested using images that have not been previously been run through it.

The testing process is one where the accuracy of the network is tested. The images in the testing folder are passed through the network. The testing phase is very similar to the training phase in that after the training dataset is completely passed through the network, the mean absolute error is calculated.

However, unlike during the training phase, the testing phase is only iterated once compared to the number of epochs over which the training phase is iterated. Also, instead of being used during backpropagation to adjust the weights and biases of the neural network, the mean absolute error is now used as a measure of the accuracy of the network. The lower the MAE, the more accurate is the network.

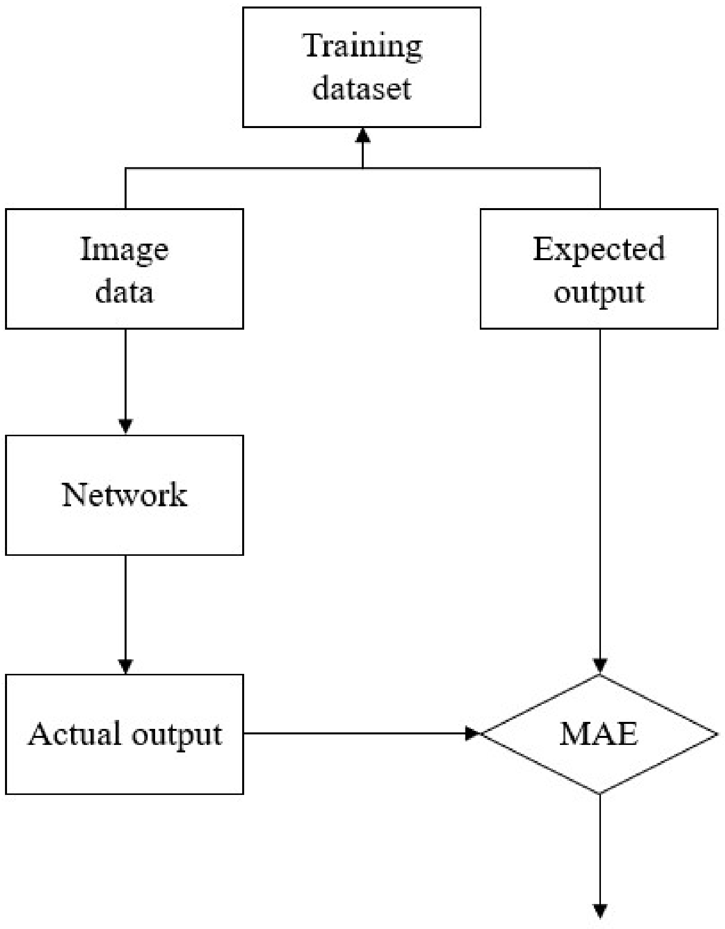


Figure. 11 Testing of neural network

Throughout the project the datasets were passed through the network across a series of epochs. At the start of training, the weights and biases e of the network were randomly generated. After going through each epoch, the mean absolute error of the network was calculated. The network with the best mean absolute error was saved. After each epoch, the mean absolute errors were compared and the best one was saved accordingly.

At the start of the project, the plan was to train the network across 1 dataset. However, the results acquired were not very convincing and thus, the network was trained across the other 2 datasets to improve on these results.

Shanghai Tech dataset

The first dataset with which the network was trained was the Shanghai tech dataset. The network was trained across 200 epochs and the results obtained were analyzed. The epoch with the best MAE was seen to be not very accurate. Across the testing dataset, the MAE was equal to 97.03. this shows that the average difference between the expected output and the actual output from the network was very large.

However, upon inspection of the error of individual images in the testing dataset, it was seen that the network was also very inconsistent. Some of the error values were very small while others twice or thrice the mean absolute error. Figure. 12 shows some of the input images with small error values.

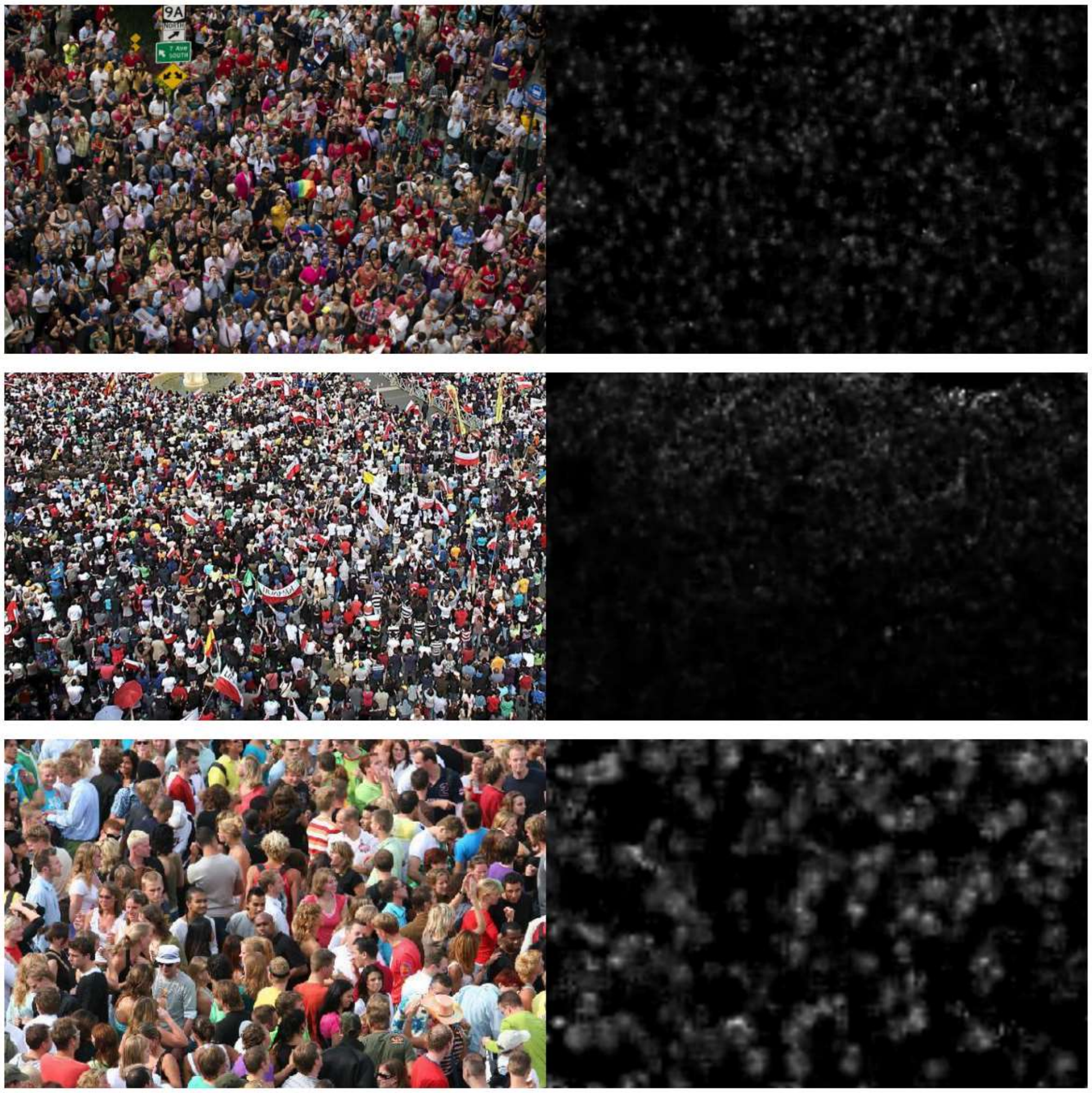


Figure. 12 Input image and network output with small error values

It can be seen that for these specific input images, the regions where the faces of the people are being shown as non-zero values. The correct positions of these people in the images could be obtained if the output is processed. However, with other input images, the output obtained is completely unusable. Figure. 13 shows some of these outputs which have a very large error value.

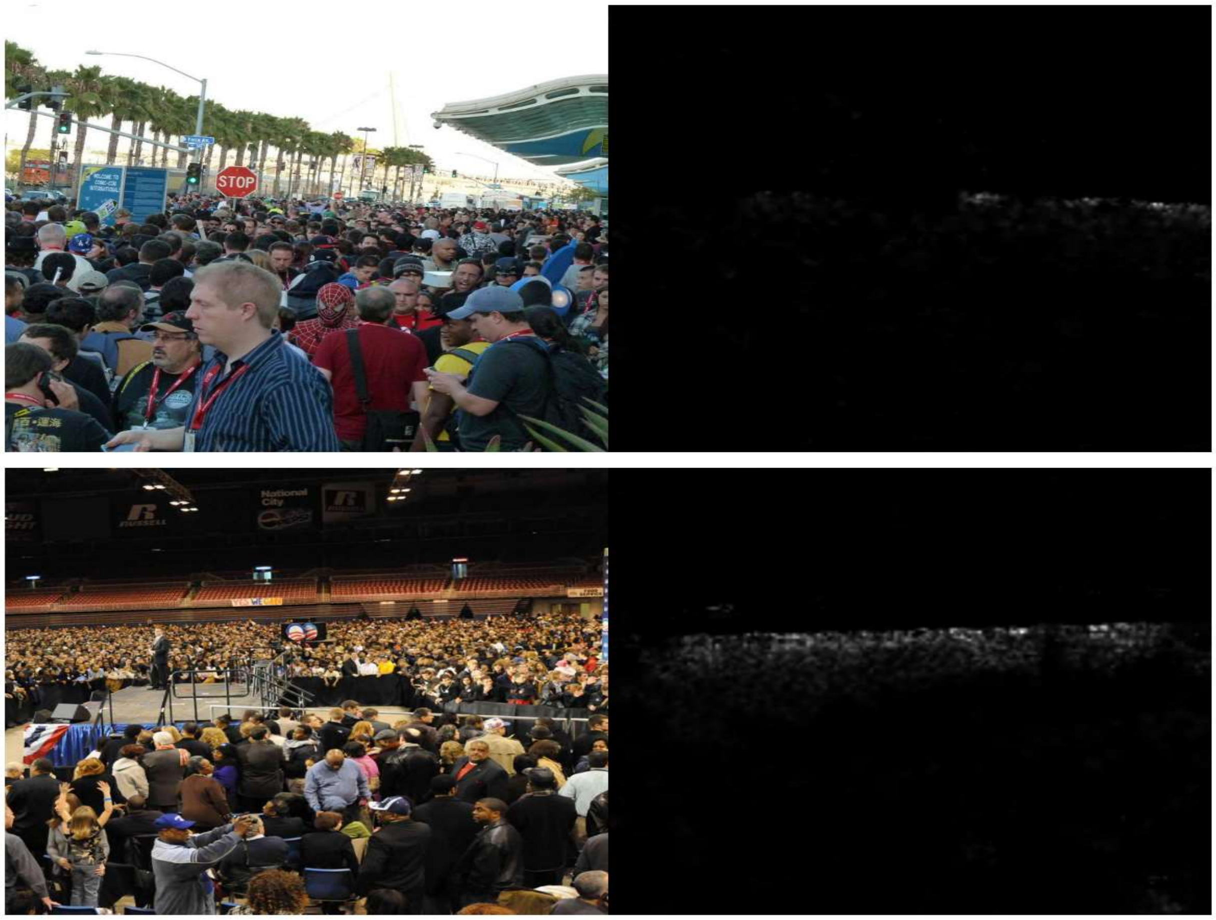


Figure. 13 Input images and network output with large error values

Compared to the output in Figure. 12, the outputs in Figure. 13 have very little similarities to the input images. The difference between the inputs in Figure. 12 and Figure. 13 is that the figures in Figure. 13 consist of zoomed in people in the foreground and zoomed out people in the background. Comparing that to the input images in Figure. 12, the figures are more uniformly distributed across the whole image. This is a factor that affects the network and must be taken into consideration when applying it in real life.

Even though some inputs had promising results, the first iteration of the neural network was ineffective in getting the desired outputs. Some possible explanation for this lack of accuracy may be:

- The neural network was not trained enough

  Even though the network was trained through 200 epochs, this may not have been enough to obtain one which is very accurate. Although not always true, a general rule of thumb is that the more a network is trained, the more accurate it becomes.

- The dataset and labels were not adequate

  The dataset may not be the most adequate one for this network. The main application of this neural network is to be able to detect individual people in an image and perform counting. This dataset contains images with very dense crowds, meaning that the people are all very close to each other. Creating an accurate network which distinguishes people who are only some couple of pixels away from each other could not be obtained.

  The dataset labels were in the form of an array of identical size as the input image and consisted of zero and non-zero values. The zero values indicated the absence of people and the non-zero values indicated their presence. For the purpose of crowd detection and crowd counting, these labels are not adequate as neither of them indicate the actual individual position of people in a crowd.

Mall dataset

The Shanghai tech dataset contained images of dense crowds and the network trained was not accurate and was inconsistent. Therefore, the next iteration of the neural network was trained with the mall dataset which contained less crowded images. Since the distance between 2 objects that needed to be detected was larger, it enabled a larger error margin. The noise produced could be removed in post processing to get an accurate representation of the crowd.

The next iteration of the network was trained using the mall dataset. Similar to the Shanghai tech dataset, it was trained for 200 epochs. The network with the best mean absolute error was obtained.

Compared to the previous iteration, the accuracy and consistency of the new network was much better.

Taking a look at individual images from the testing dataset, it can be seen that blobs of white spaces can be seen at the positions of the people in the crowd. Some of these input images along with the output of the neural network is shown in Figure. 14.

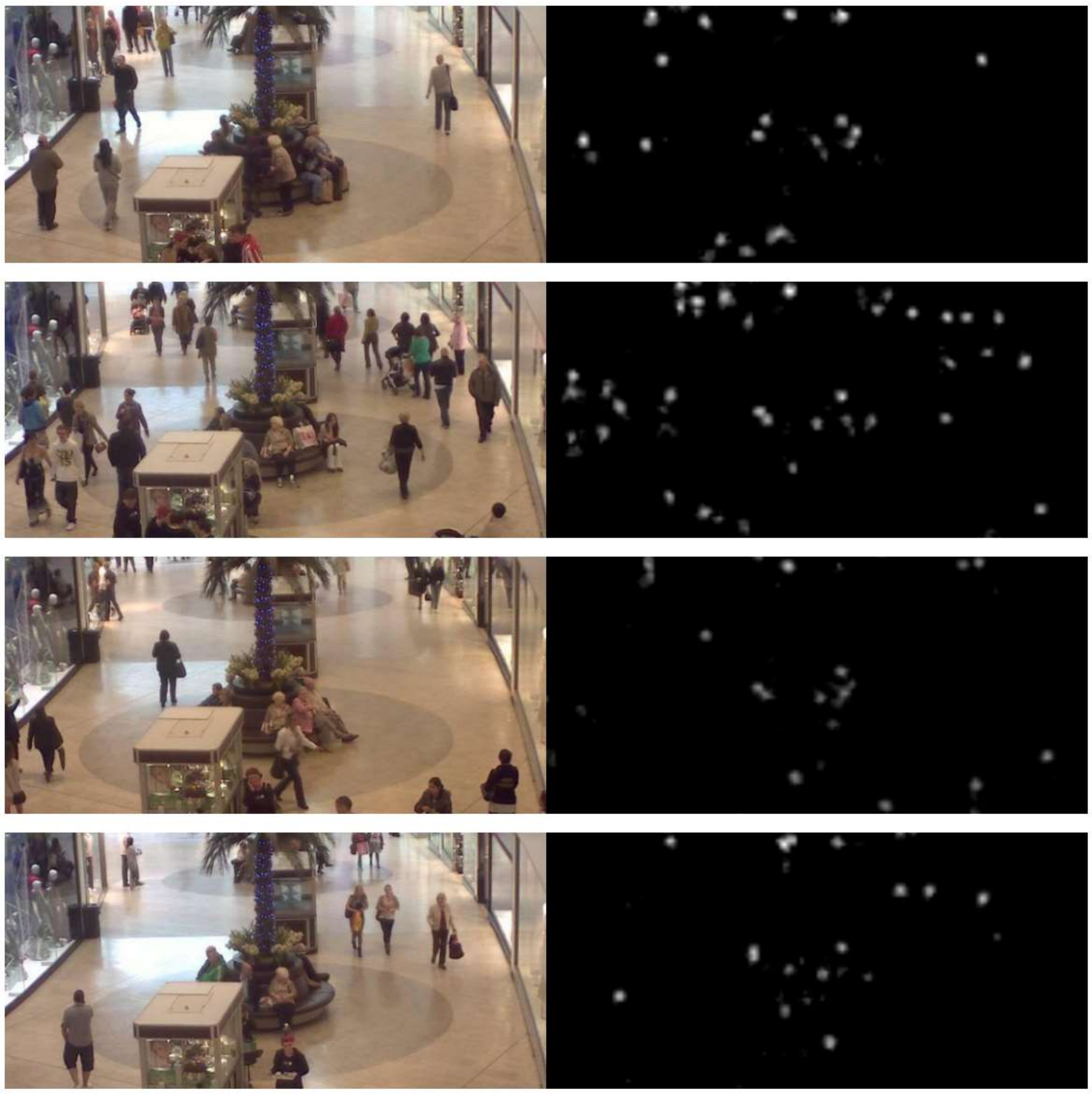


Figure. 14 Input and output of second iteration of neural network

From these observations, it can be deduced that:

- The mall dataset could be better trained across 200 epochs than the Shanghai tech dataset.
- The dataset label is not an issue for the mall dataset. This may be because of the less dense crowd in the latter.

However, some of the problems from the previous iteration of the neural network still lingers:

- If 2 objects are too close together, they may appear as a single object. This problem may be resolved by training the neural network through more epoch.
- There is also some noise present which may affect the readings in the application process later on. However, this problem may be resolved by applying a larger threshold when filtering out the noise.

UCSD dataset

To confirm these observations, the training process was repeated a third time. The dataset used was the UCSD dataset. This time, the process was run across 100 epochs. The epoch with the best mean absolute error was saved and was then tested using the testing dataset.

The results obtained was very similar to that of the second iteration of the neural network in that the accuracy and consistency of the network was an improvement on the first one. Some input images and their corresponding output from the neural network are shown in Figure. 15.

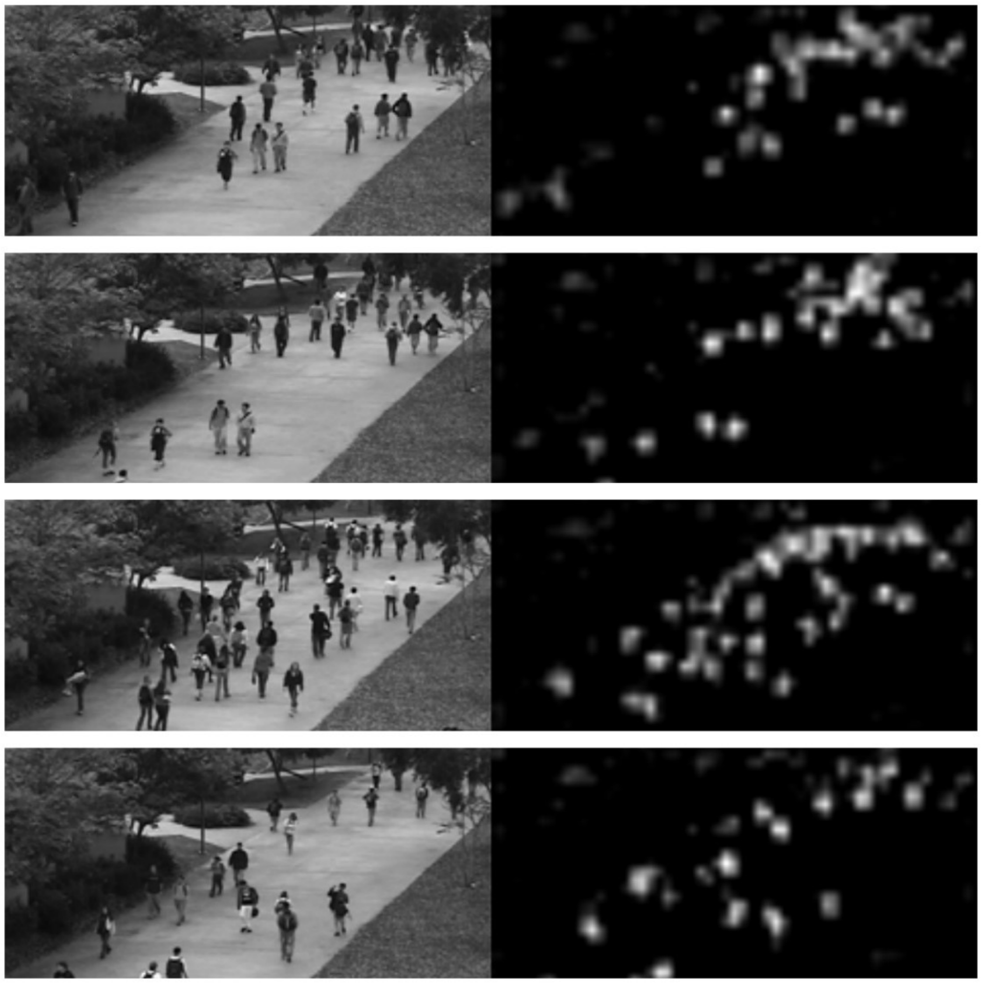


Figure. 15 Input images and output of third iteration of the neural network

Even though it was not as accurate as the second iteration, the third network performed relatively well compared to the first one. Most of the people in the input images had a corresponding blob on the output of the network. However, like the previous ones, it could not separate 2 different objects which were close to each other. Since this error is recurrent across all the datasets, it can be concluded that the problem comes from the network and not from the datasets.

The choice of dataset is quite important when working with machine learning. As we saw in the first iteration of the neural network, an inadequate dataset may lead to inaccurate and inconsistent results.

As a result, the dataset that will be used for further training will be the mall dataset. Using the mall dataset had a better accuracy that the UCSD dataset. Moreover, because the UCSD dataset is in grayscale, it may prove to be inadequate in application where the input image is in color.

Application of neural network

In this section, the final neural network developed from the mall dataset will be used for some real-life application.

Real time crowd counting

The first application and the main objective of this project is to perform real time crowd counting on a live video. Using the imutils library, the live video is separated into frames. Each frame is passed into the final neural network.

As seen previously, the output of the neural network consists of white blobs on a black background. These white blobs indicate the position of a detected person. Figure. 16 shows an example of an input frame and the output of the neural network.

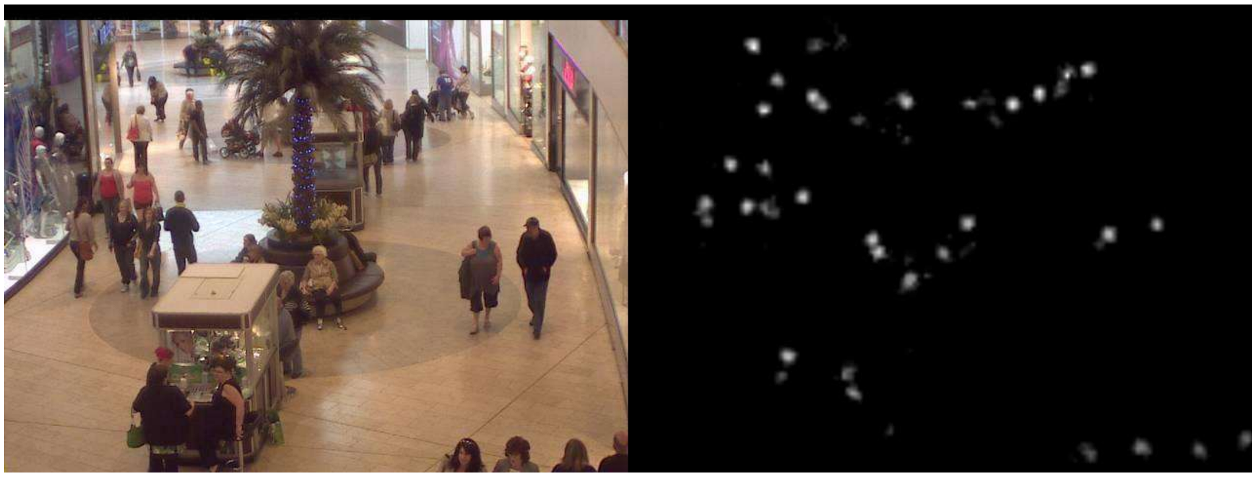


Figure. 16 Input frame and output of neural network

The output of the neural network still contains some noise and must be processed further to obtain a cleaner image. To get rid of the noise, a simple threshold algorithm is applied to the image. The pixels above the threshold value will remain the same while those below it will be reduced to zero. The output to the thresholding algorithm compared to the original frame is shown in Figure. 17.

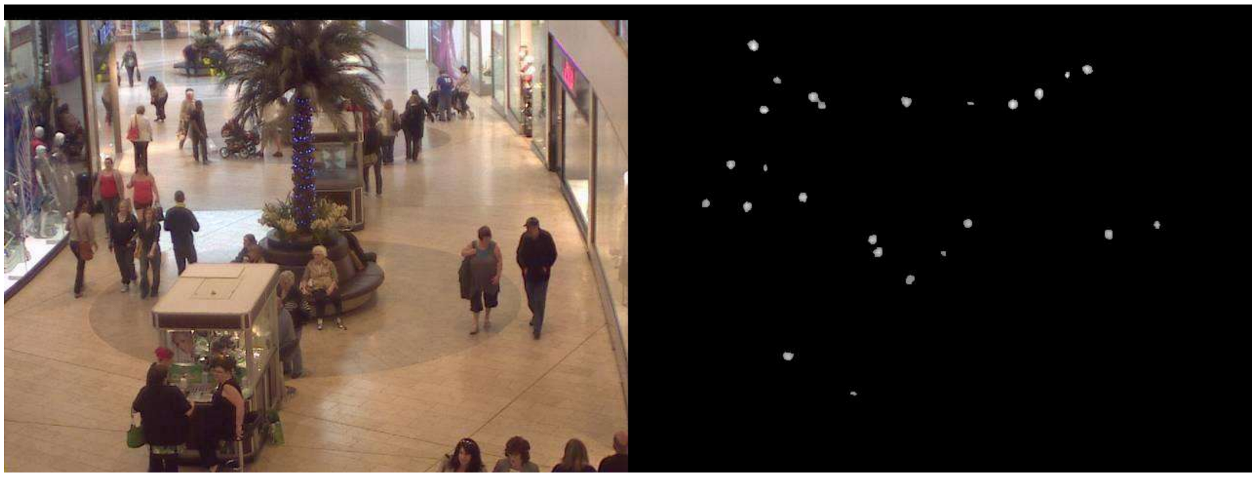


Figure. 17 Input frame and output of threshold algorithm

The choice of the threshold value must be chosen carefully. Should it be too small, some noise will be present in the output. However, if it is too large, some of the required blobs will be filtered out of the output. The information on the number of people in the crowd will then be computed incorrectly.

After that, the remaining blobs are further processed to make them easier to locate. To do that the OpenCV library is used. The output after the processing is shown in Figure. 18.

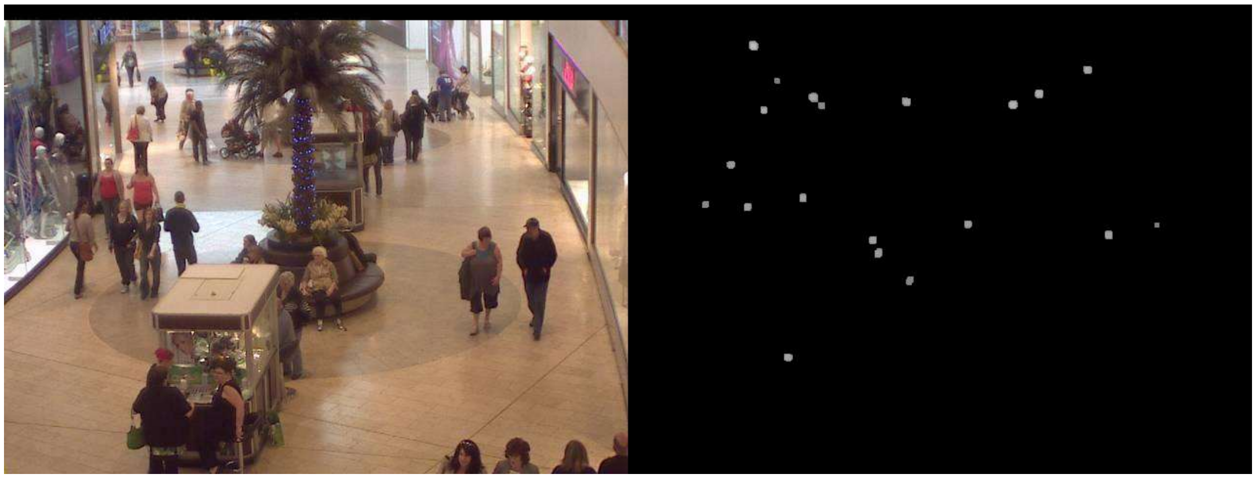


Figure. 18 Input frame and processed output

The processed image is used to find the coordinates of the people in the input frame. Each blob found on the processed output denotes a person inside the input frame. For each blob, a circle is drawn on the input frame at the corresponding coordinate to show a person. The output frame is shown in Figure. 19.

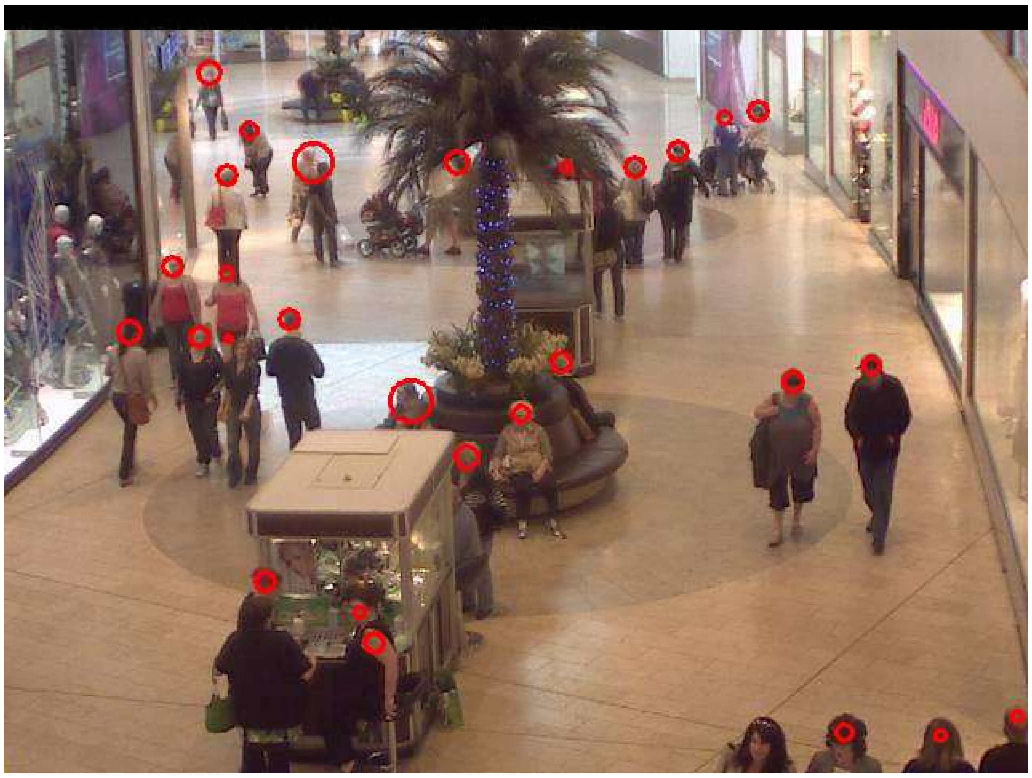


Figure. 19 Output frames

The number of blobs found is then computed and is displayed on the output frame as the total number of people in the image. The output frame with the crowd counting is shown in Figure. 20.

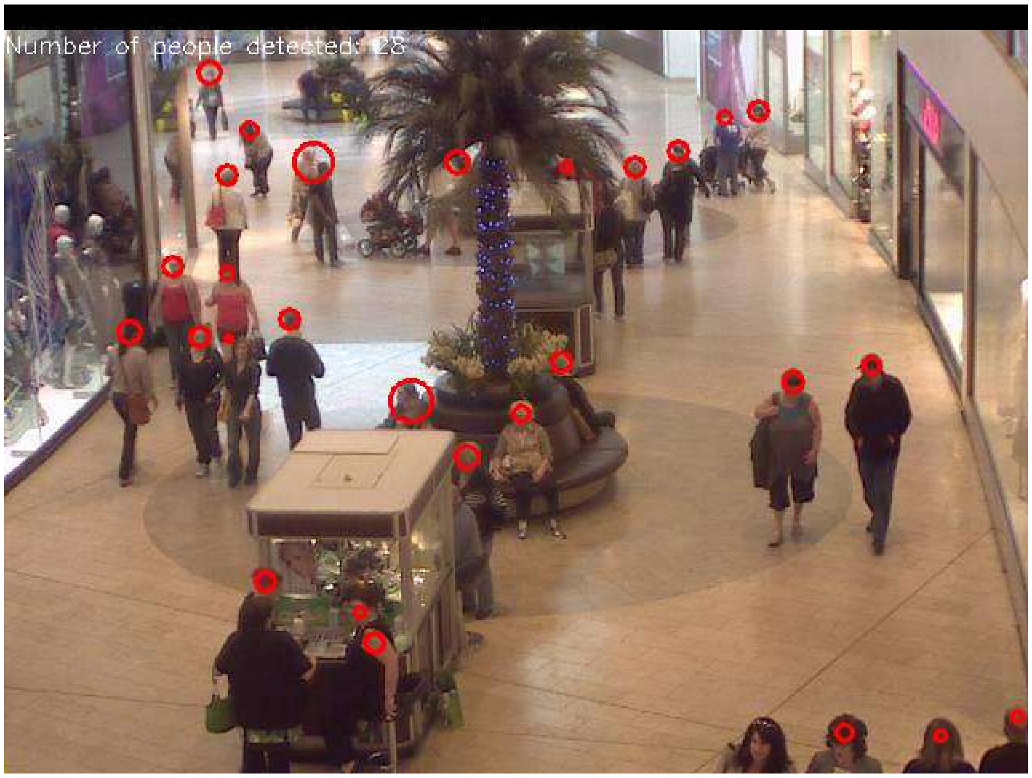


Figure. 20 Output frames with crowd counting

The accuracy of this application depends on the accuracy of the neural network. If the neural network can detect the positions of the people correctly, the accuracy of getting the correct number of people will be better.

However, this system shows some limitations. These are:

- Two people too close to one another are treated a single unit
- Some people remain undetected.
- Some extra people are detected.
- The performance of the system is dependent on hardware.

The first 3 limitations are mainly due to the limitations of the neural network. It shows that the accuracy of the neural network is not sufficiently high enough. The problem of combining 2 people into one was detected during training and testing of the neural network. It could be rectified by training the neural network more or by using a different neural network. The same applies to the problem of not detecting people.

On the other hand, the problem of detecting extra people is due to the noise produced by the neural network. Even though a threshold algorithm is applied to remove such noise, some still pass through and affect the output of the system. The total number of people in the frame is therefore overstated. A higher threshold value can be applied to remove the existing noise. However, this may lead to further inaccuracy as some actual outputs may be filtered out.

Since the system makes use of a convolution neural network, the performance of the system is heavily dependent on hardware, namely CPUs and GPUs. Having better hardware will lead to a better frame rate in the output feed. During the testing of the system on the webcam footage of a laptop, a frame rate of about 7 fps was obtained. The CPU of the laptop was an intel core i7-8550 and the GPU used was an NVIDIA GeForce MX 150. These hardware can be considered as mid to low tier hardware. Therefore, the performance of the system is acceptable. However, better hardware is needed if the neural network becomes more complex or if better performance is desired.

Real time path tracking

The next application performed was to obtain the paths of people in a real time video. Similar to the crowd counting application, the trained neural network will be used to track the positions of people in an image.

The input video frame is processed frame by frame. Each frame is passed through the neural network to get the positions of people in the frame. The positions of the obtained points are compared to that of the previous frame. a point from 1 frame is compared to each point in the other frame. The 2 points from each frame having the smallest distance between them are joined together to form a path.

A constant minimum distance, d_{min}, and a maximum distance, d_{max}, is inputted to indicate the limits of detection.

The paths of the individuals are computed as follows:

- o If the minimum distance calculated between 2 points is less than the constant distance, d_{min}, it is assumed that the person has not moved at all or has moved very little. A path is not drawn between those 2 points.
- o If the minimum distance calculated between 2 points is greater than the other constant distance, d_{max}, the 2 points are considered further apart enough to not be associated with each other. A path is also no drawn between these 2 points.
- o If the minimum distance calculated between 2 points is within these limits, a path is drawn on the image.

Figure. 21 and Figure. 22 show the first and second frame of a video feed. The first frame consists only of the positions of the people that were detected in the frame, indicated by a circle. The second frame contains the detected positions and the paths between the closest points from the previous frame.

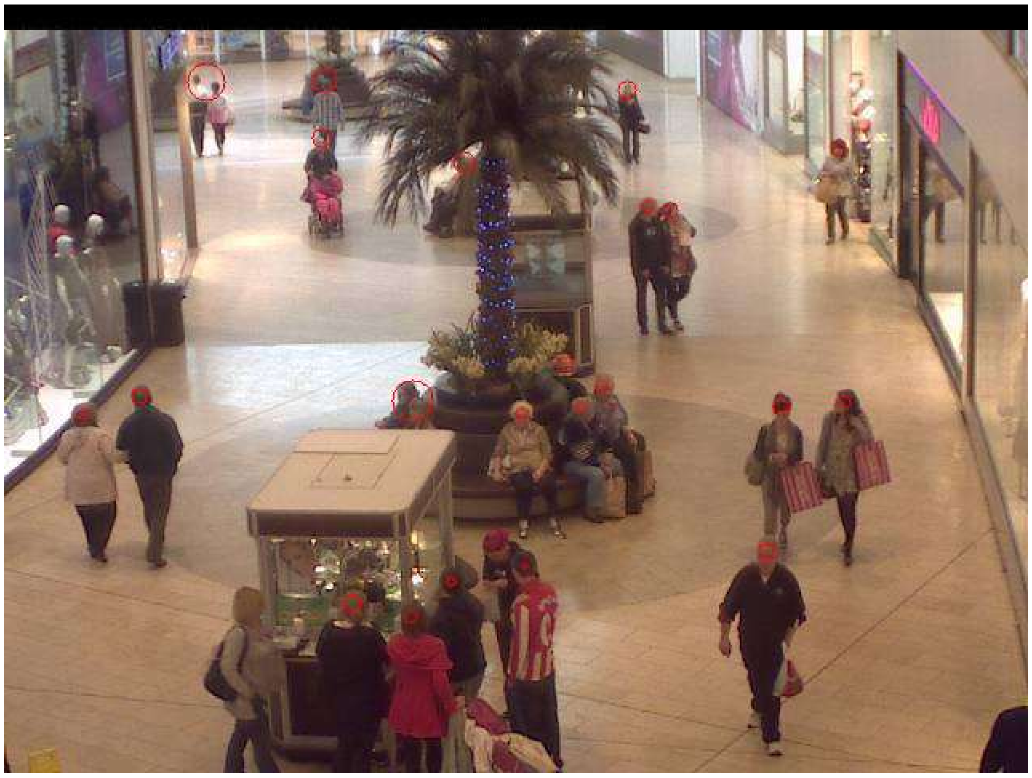


Figure. 21 First frame

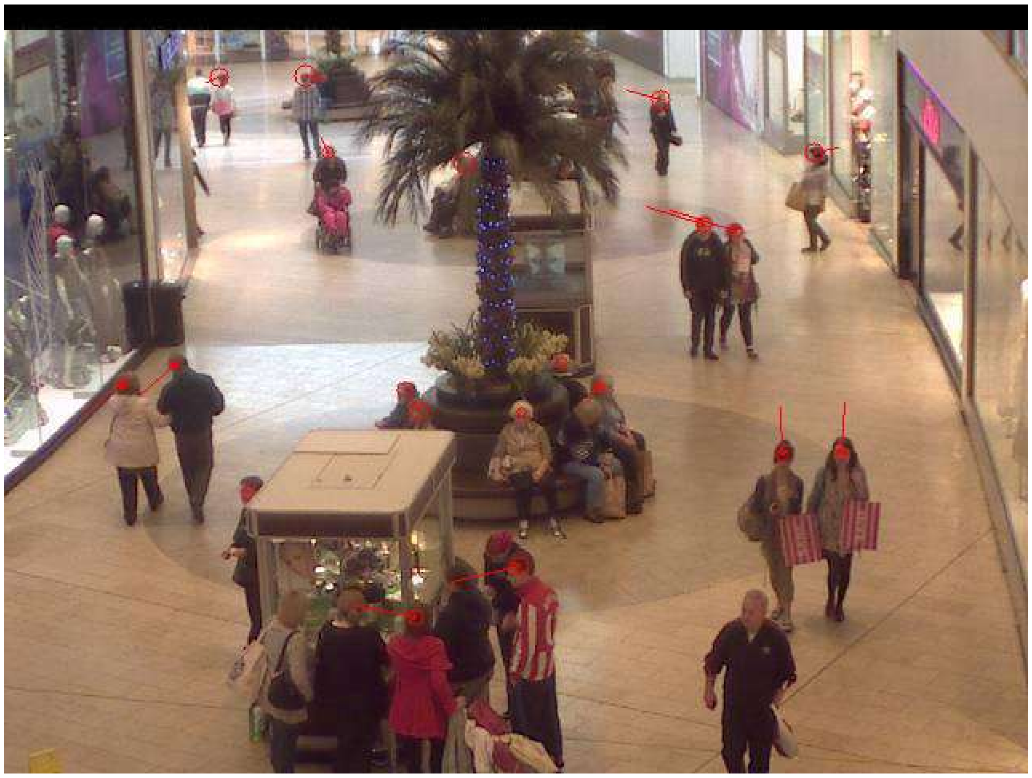


Figure. 22 Second frame with path lines

Since the same trained neural network is being used for the path tracking, the same problem as before arises; detection is not always accurate. However, another problem, this time during the path tracking, can be seen.

The path tracking algorithm works in such a way that a point in the initial frame is compared to each point in the next frame. the problem that arises is that sometimes, 2 paths lead to the same final point, which should not be possible. To prevent this, the algorithm is modified in such a way that, once a path is set up between 2 points, the point in the final frame corresponding to that path is no longer taken into consideration when calculating the distance between subsequent points in the initial and final frames. As a result, paths with intersecting points are prevented.

During testing of the above-mentioned solution, another problem occurred. Since the point in the final frame is no longer taken into consideration after finding an initial distance which was within the limits, d_{min} and d_{max}, the smallest distance between 2 points in different frames is sometimes not obtained. This led to the wrong path being selected.

A possible solution is to first calculate the distance between every point in the 2 different frames. These distances can then be sorted such that the smallest is at the start. The closest points are joint by a path and are subsequently not taken into consideration. The next closest points are joint, and the process is repeated until either all points are joint together or the distances left are all greater than d_{max}. Due to time constraints, this solution could not be tested.

Conclusions and future works

Conclusions

Using deep learning for crowd counting on a live video feed is possible. The neural network that was used could detect and count most of the people in each frame. However, there are still some limitations when using the neural network trained from the mall dataset.

The main limitation of the system is that it is sometimes unable to distinguish between 2 objects. When the distance between 2 objects becomes less than a certain value, the neural network shows these 2 objects as 1 and the crowd counting is wrongly computed. As a result, this system may be unable to perform the crowd counting in a densely crowded place because the distance between people in the frames of the live video will be too small. This was partly shown during training of the network with the Shanghai tech dataset. The accuracy of the network was very bad, and it was inconsistent.

Another shortcoming of the overall system is that it allows some noise to pass through. They are then considered as an object by the neural network and therefore, the number of people in the crowd is overstated.

Most of these shortcomings can be overcome by either:

- Training the network through more epochs.
- Using more appropriate datasets.

As for the detection and tracking of paths, it is possible provided a neural network capable of detecting every people in each frame is obtained. Similar problems occurring in the crowd counting application occurred during the path tracking application. This is because the same neural network was used for both applications.

Therefore, it is imperative to obtain a solid neural network which is capable of obtaining the positions of people in any individual frame. Further research must be done to improve on the accuracy of such neural network.

Recommendation for future works

As mentioned above, the advancement of neural networks is essential to obtain better result in the respective application. This can be done in 2 ways:

- Improving the dataset used during training of the neural network.
- Using a different type of neural network.

The datasets used in this project are very small ones containing at most a few thousand images. Deep learning depends hugely on the dataset used. Therefore, there is a need for more datasets. Obtaining data, whether in the form of images or video, is very easy. However, a dataset consists of the data as well as the data label. Creating these data labels are very time consuming. For the advancement of deep learning networks, larger datasets are required.

Another possible improvement which may be done to improve accuracy is to use a more complex neural network. In this project, the neural network used is a very simple one consisting of only some convolution layers and therefore, it is not computationally demanding. A more complex network may be used to increase its accuracy. However, the trade off is that, the more complex a network is, the more computationally demanding it becomes. Better CPUs and GPUs will be needed to attain acceptable speed.

References

[1] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[2] W. Li, R. Zhao, T. Xiao, and X. Wang, "DeepReID: Deep Filter Pairing Neural Network for Person Re-identification," *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[3] W. Ouyang and X. Wang, "Joint Deep Learning for Pedestrian Detection," *2013 IEEE International Conference on Computer Vision*, 2013.

[4] X. Zeng, W. Ouyang, M. Wang, and X. Wang, "Deep Learning of Scene-Specific Classifier for Pedestrian Detection," *Computer Vision – ECCV 2014 Lecture Notes in Computer Science*, pp. 472–487, 2014.

[5] N. Wang and D.-Y. Yeung, "Learning a Deep Compact Image Representation for Visual Tracking," *Advances in neural information processing systems*, pp. 809–817, 2013.

[6] M. Müller, A. Bibi, S. Giancola, S. Alsubaihi, and B. Ghanem, "TrackingNet: A Large-Scale Dataset and Benchmark for Object Tracking in the Wild," *Computer Vision – ECCV 2018 Lecture Notes in Computer Science*, pp. 310–327, 2018.

[7] C. Zhang, H. Li, X. Wang, and X. Yang, "Cross-scene crowd counting via deep convolutional neural networks," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[8] H. Nam and B. Han, "Learning Multi-domain Convolutional Neural Networks for Visual Tracking," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[9] D. J. Montana and L. Davis, "Training Feedforward Neural Networks Using Genetic Algorithms," *IJCAI*, vol. 89, pp. 762–767, 1989.

[10] C. M. Bishop, *Neural networks for pattern recognition*. Oxford: Oxford Univ. Press, 1999.

[11] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," *Proceedings of the fourteenth international conference on artificial intelligence and statistics.*, 2011.

[12] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.