# INTERNSHIP-REPORT

**Prepared by : Tikam Alma**
**Organization : Amnex Infotechnologies Pvt. Ltd.**
**Institute : Dhirubhai Ambani Institute of Information and Communication Technology**
**Topic : Crowd-Density[Number Estimation] Through Deep Learning.**
**Mentor : Mr. Arjun Bhasin**
**Duration : 4-June-2018 - 25-July-2018**

# Contents

# 1 Introduction.

In this internship my main aim was to learn and implement machine learning and deep learning in real life.I started from scratch with only knowledge of python, then i learnt Machine Learning made some programs and then move forward to deep learning with Crowd-Density Estimation Project.
The project assigned to me was  - That is counting the number of peoples in the crowd and make a **Crowd Density Estimation** deep learning model to predict in the real life crowd so that when a threshold of crowd will reach the application will give alert to that specific area of crowd.

In the new year eve of 2015, 35 people were killed in a massive stampede in Shanghai, China.Unfortunately, since then, many more massive stampedes have taken place around the world which have claimed many more victims. Accurately estimating crowds from images or videos has become an increasingly important application of computer vision technology for purposes of crowd control and public safety. In some scenarios, such as public rallies and sports events, the number or density of participating people is an essential piece of information for future event planning and space design. Good methods of crowd counting can also be extended to other domains, for instance, counting cells or bacteria from microscopic images, animal crowd estimates in wildlife sanctuaries, or estimating the number of vehicles at transportation hubs or traffic jams, etc.
Crowd counting is the problem of estimating the number of people in a still image or a video. It has drawn a lot of attention due to the need for solving this problem in many real-world applications such as video surveillance, traffic control, and emergency management.

I implemented this project by using 2-3 research paper published in CVPR and some other conferences.Using CNN and other architectures and concepts of Deep Learning like MCNN, Cascaded CNN, Multi-Learning Task etc.There are different approaches and methods to estimate the crowd density and implemented in different architectures.

# 2 Approaches to Estimate Crowd Density

the potential solutions for crowd scenes analysis can be classified into three categories: detection-based methods, regression based methods, and density estimation-based methods. By combining the deep learning, the CNN-based solutions show even stronger ability in this task and outperform the traditional methods.

### 2.1. Detection-based approaches

Most of the early researches focus on detection-based approaches using a moving-window-like detector to detect people and count their number . These methods require well-trained classifiers to extract low-level features from the whole human body . However, they perform poorly on highly congested scenes since most of the targeted objects are obscured. To tackle this problem, researchers detect particular body parts instead of the whole body to complete crowd scenes analysis.

### 2.2. Regression-based approaches

Since detection-based approaches can not be adapted to highly congested scenes, researchers try to deploy regression-based approaches to learn the relations among extracted features from cropped image patches, and then calculate the number of particular objects. More features,such as foreground and texture features, have been used for generating low-level information.

### 2.3. Density estimation-based approaches

When executing the regression-based solution, one critical feature, called saliency, is overlooked which causes inaccurate results in local regions. Lempitsky  propose a method to solve this problem by learning a linear mapping between features in the local region and its object density maps. It integrates the information of saliency during the learning process. Since the ideal linear mapping is hard to obtain, using random forest regression to learn a non-linear mapping instead of the linear one.

To estimate the number of people in a given image via the Convolutional Neural Networks (CNNs), there are two natural configurations. One is a network whose input is the image and the output is the estimated headcount. The other one is to output a density map of the crowd (say how many people per square meter), and then obtain the head count by integration.

- Density map preserves more information. Compared to the total number of the crowd, density map gives the spatial distribution of the crowd in the given image, and such distribution information is useful in many applications. For example, if the density in a small region is much higher than that in other regions, it may indicate something abnormal happens there.
- In learning the density map via a CNN, the learned filters are more adapted to 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.

### 2.4. CNN-based approaches

Literature also focuses on the CNN-based approaches
to predict the density map because of its success in classification and recognition . In the work presented by Walach and Wolf , a method is demonstrated with selective sampling and layered boosting. Instead of using patch-based training,  an end-to-end regression method using CNNs which takes the entire image as input and directly outputs the final crowd count. The first work purely using convolutional networks and dual-column architecture for generating density map. In 2018 some researchers used CNN which uses the high-level prior information to boost the density prediction performance. An improved structure is proposed a multi-column based architecture (MCNN) for crowd counting. It is clear that the CNN-based solutions outperform the previous research works.

# 3 Mathematical Part-Gaussian Function,Gaussian Blur.

Gaussian Function and Gaussian Blur used for generation of Density Maps of images.

Gaussian Function - In mathematics, a **Gaussian function**, often simply referred to as a **Gaussian**, is a function of the form:

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

for arbitrary real constants a, b and c. It is named after the mathematician Carl Friedrich Gauss. The graph of a Gaussian is a characteristic symmetric "bell curve" shape. The parameter a is the height of the curve's peak, b is the position of the center of the peak and c (the standard deviation, sometimes called the Gaussian RMS width) controls the width of the "bell".
Gaussian functions are often used to represent the probability density function of a normally distributed random variable with expected value $\mu = b$ and variance $\sigma^2 = c^2$. In this case, the Gaussian is of the form:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}.$$

Gaussian Blur: In image processing, a **Gaussian blur** (also known as **Gaussian smoothing**) is the result of blurring an image by a Gaussian function (named after mathematician and scientist Carl Friedrich Gauss). It is a widely used effect in graphics software, typically to reduce image noise and reduce detail. The visual effect of this blurring technique is a smooth blur resembling that of viewing the image through a translucent screen, distinctly different from the bokeh effect produced by an out-of-focus lens or the shadow of an object under usual illumination. Gaussian smoothing is also used as a pre-processing stage in computer vision algorithms in order to enhance image structures at

different scales.Mathematically, applying a Gaussian blur to an image is the same as convolving the image with a Gaussian function. This is also known as a two-dimensional Weierstrass transform. By contrast, convolving by a circle (i.e., a circular box blur) would more accurately reproduce the bokeh effect. Since the Fourier transform of a Gaussian is another Gaussian, applying a Gaussian blur has the effect of reducing the image's high-frequency components; a Gaussian blur is thus a low pass filter.
The Gaussian blur is a type of image-blurring filter that uses a Gaussian function (which also expresses the normal distribution in statistics) for calculating the transformation to apply to each pixel in the image. The equation of a Gaussian function in one dimension is

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

in two dimensions, it is the product of two such Gaussians, one in each dimension:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

## Two-dimensional Gaussian function - 2D-Gaussian Kernel

In two dimensions, the power to which *e* is raised in the Gaussian function is any negative-definite quadratic form. Consequently, the level sets of the Gaussian will always be ellipses.
A particular example of a two-dimensional Gaussian function is

$$f(x, y) = A \exp\left(-\left(\frac{(x - x_o)^2}{2\sigma_x^2} + \frac{(y - y_o)^2}{2\sigma_y^2}\right)\right).$$

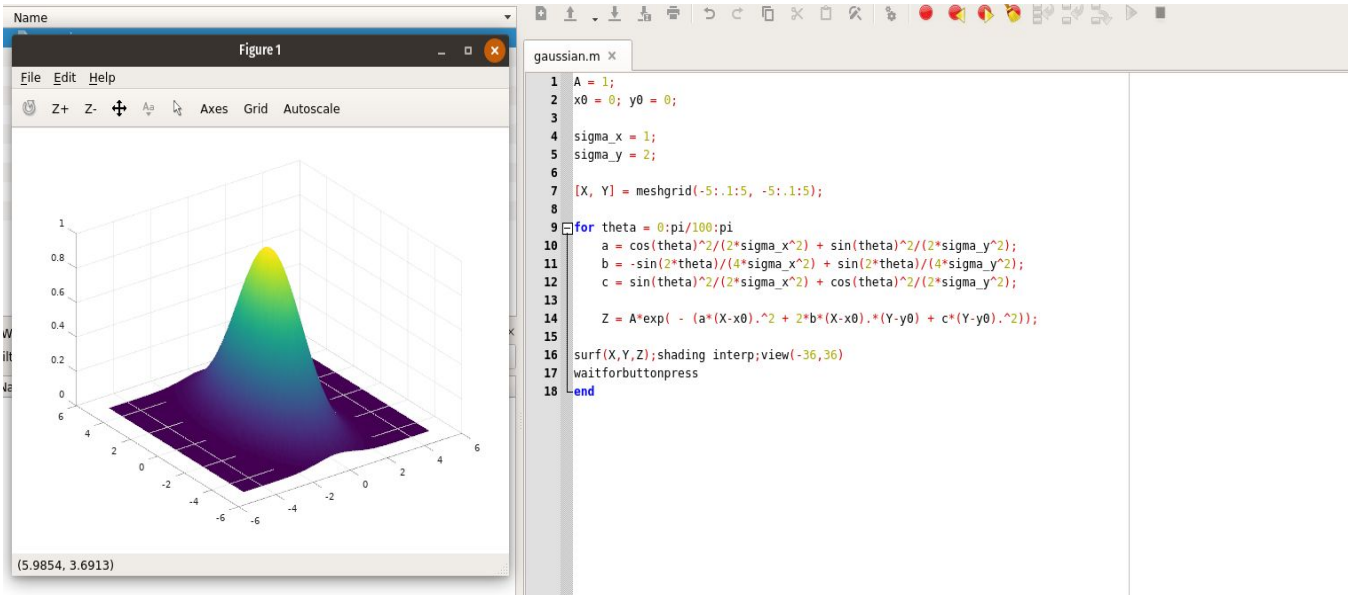
Here the coefficient $A$ is the amplitude, $x_o$,$y_o$ is the center and $\sigma_x$, $\sigma_y$ are the $x$ and $y$ spreads of the blob. The figure on the right was created using $A = 1$, $x_o = 0$, $y_o = 0$, $\sigma_x = \sigma_y = 1$.
The volume under the Gaussian function is given by

$$V = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y)\, dx\, dy = 2\pi A \sigma_x \sigma_y.$$

In general, a two-dimensional elliptical Gaussian function is expressed as

$$f(x, y) = A \exp\left(-\left(a(x - x_o)^2 + 2b(x - x_o)(y - y_o) + c(y - y_o)^2\right)\right)$$

Implementation In Matlab which will be used for Density Mapping.




For the general form of the equation the coefficient $A$ is the height of the peak and $(x_o, y_o)$ is the center of the blob.
If we set

$$a = \frac{\cos^2\theta}{2\sigma_x^2} + \frac{\sin^2\theta}{2\sigma_y^2}$$

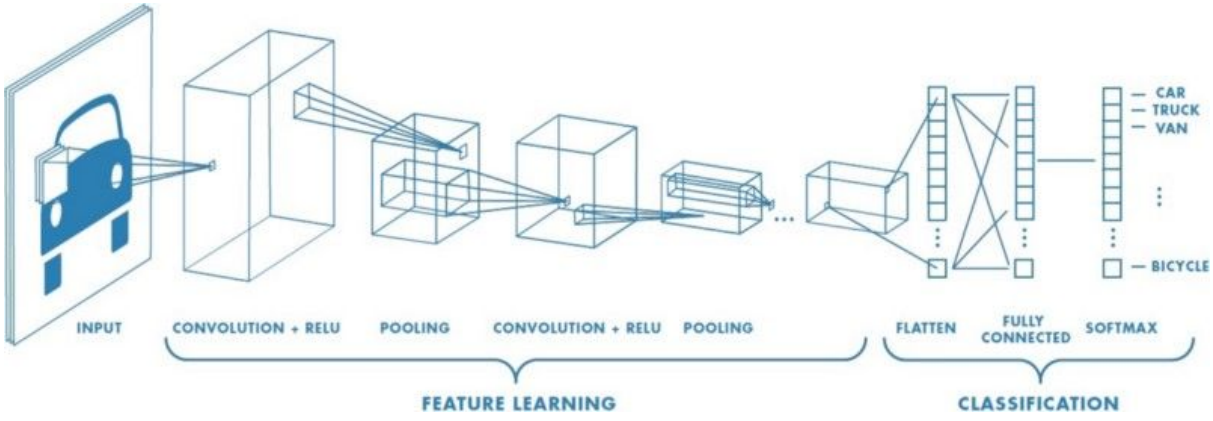
$$b = -\frac{\sin 2\theta}{4\sigma_x^2} + \frac{\sin 2\theta}{4\sigma_y^2}$$

$$c = \frac{\sin^2\theta}{2\sigma_x^2} + \frac{\cos^2\theta}{2\sigma_y^2}$$

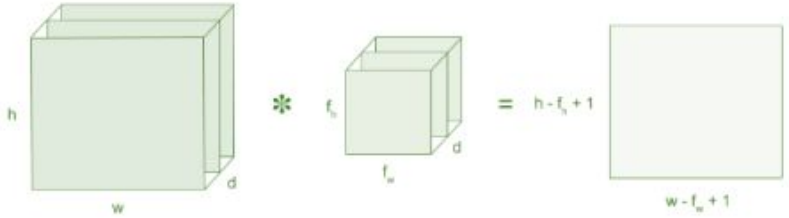
# 4 Convolution Neural Network (CNN) concept.

### 4.1 Basic Convolution Network Structure

- Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d( h = Height, w = Width, d = Dimension ). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.
- Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1.



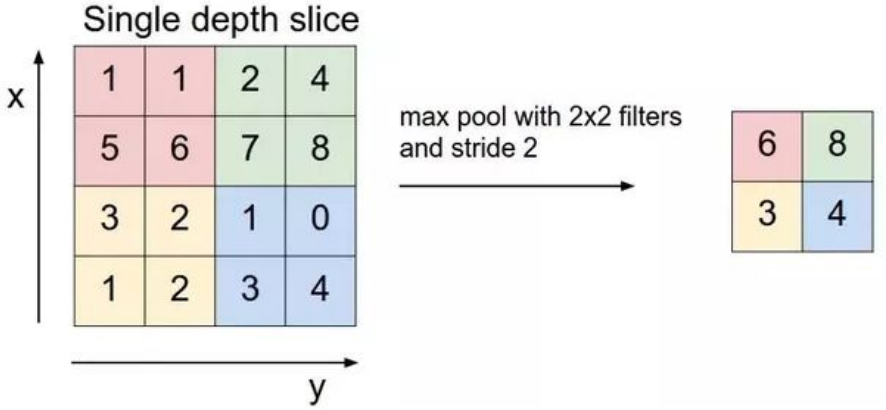

- Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernal
  - An image matrix (volume) of dimension **(h x w x d)**
  - A filter **($f_h$ x $f_w$ x d)**
  - Outputs a volume dimension **(h - $f_h$ + 1) x (w - $f_w$ + 1) x 1**



- **Pooling Layer**

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or downsampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types: Max Pooling,Average Pooling,Sum Pooling

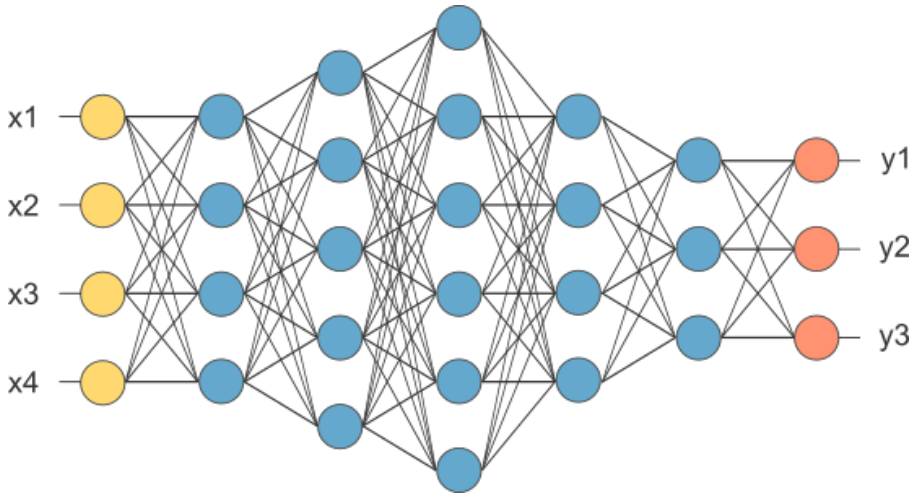
Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.




- **Fully Connected Layer**

The layer we call as FC layer, we flattened our matrix into vector and feed it into a fully connected layer like neural network.

Feature map matrix will be converted as vector (x1, x2, x3, ...). With the fully connected layers, we combined these features together to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs



# 5 Density Map Generation.

If there is a head at pixel x i , we represent it as a delta function δ(x − x i ). Hence an image with N heads labeled can be represented as a function

H(x) = $\sum_{i=1}^{N}$ δ(x − x i )

- To convert this to a continuous density function, we may convolve this function with a Gaussian kernel G σ so that the density is F (x) = H(x) ∗ G σ (x).
- However, such a density function assumes that these x i are independent samples in the image plane which is not the case here: In fact, each x i is a sample of the crowd density on the ground in the 3D scene and due to the perspective distortion, and the pixels associated with different samples x i correspond to areas of different sizes in the scene.
- For each head x i in a given image, we denote the distances to its k nearest neighbors as $\{d_1^i, d_2^i, \ldots, d_m^i\}$. The average distance is therefore $d^i = \frac{1}{m}\sum_{j=1}^{m} = d_1^i$

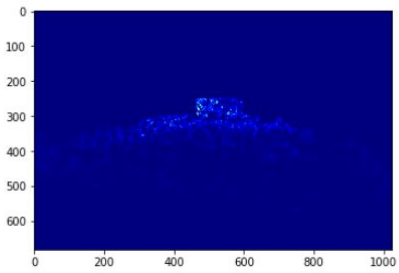
  Thus,the pixel associated with $x_i$ corresponds to an area on the ground in the scene oughly of a radius proportional to $d_i$ therefore, to estimate the crowd density around the pixel $x_i$ , we need to convolve δ(x − x i ) with a Gaussian kernel with variance σ i proportional to $d_i$ , More precisely, the density F will be

  F (x) = $\sum^{N} \delta(x - x\,i\,) * G\,\sigma\,i\,(x),$ with σ $_i$ = β d $_i$

```
In [5]: img='/home/tikam/ML/Crowd-count/final/CSRNet/data/part_A_final/train_data/images/IMG_5.jpg'
```

```
In [6]: plt.imshow(Image.open(img))
Out[6]: <matplotlib.image.AxesImage at 0x7efe1da1b890>
```


```
In [8]: gt_file = h5py.File(img.replace('.jpg','.h5').replace('images','ground_truth'),'r')
        groundtruth = np.asarray(gt_file['density'])
        plt.imshow(groundtruth,cmap=CM.jet)
        plt.show()
```



# 6 Crowd Density Estimation Using Cascaded CNN.

## 6.1 What is Cascaded CNN?

- Depth images uses different grey levels to represent the distance between captured objects and the cameras.
- Texture images associated with depth images can provide a depth perception of real scenes, and support free navigation into other viewpoints by some view synthesis techniques(e.g,DIBR).
- **Problem -** 3d Image - Lots of size and Space,compression needed.
  - Lossy compression inevitable degraded the images quality.
  - Depth map reduces the images quality,pixels are blurred.
  - Enhancement Required.

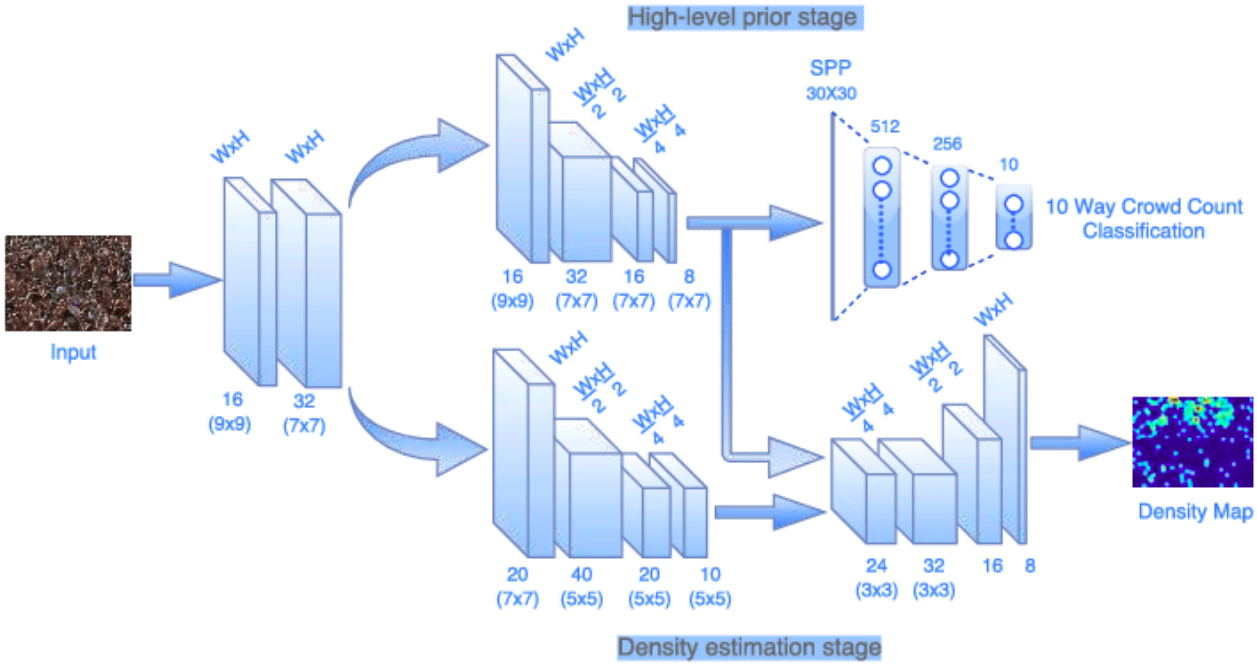
- **Solution -** Cascaded Single input single output end-to-end CNN for quality enhancement of the compressed depth images.
  - Each cascaded unit of the network is 4 Layer CNN
  - Stage by Stage Enhancement quality of the Depth map.

**Network Design**

- The well trained CNNs are used to assemble the deep one by cascade.
- Network setting of each shallow CNN:
  Layer1 : 9X9X64; Layer2: 7X7X32; Layer3: 1X1X16; Layer4: 5X5X1
- Each layer with corresponding Zero padding.
- Parametric Rectified Linear Unit(PReLU) is applied on the first three convolution layers.
- The whole network is re-trained.

## 6.2 Explanation Of Code-Cascaded CNN Crowd Estimation.

- Classifying images into groups - efficient and more accurate.
- The high-level prior learns to classify the count into various groups whose class labels are based on the number of people present in the image. By exploiting count labels, the high-level prior is able to estimate coarse count of people in the entire image irrespective of scale variations thereby enabling the network to learn more discriminative global features.
- The high-level prior is jointly learned along with density map estimation using a cascade of CNN networks. The two tasks (crowd count classification and density estimation) share an initial set of convolutional layers which is followed by two parallel set of networks that learn high-dimensional feature maps relevant to high-level prior and density estimation, respectively. The global features learned by the high-level prior are concatenated with the feature maps obtained from the second set of convolutional layers and further processed by a set of fractionally strided convolutional layers to produce high resolution density maps.



- The initial shared network consist of 2  convolutional layers with a parametric Rectified Linear Unit(PReLU) activation function after every layer.
- The first convolutional network layer has 16 feature maps with a filter of 9X9 and the second convolutional layer has 32 features with a filter size of 7X7.
- The feature maps generated by this shallow networks are shared by the two stages: high level prior stage that is crowd classification and density estimation.
- **Cascaded CNN - PReLu with 4 CNN layers:** The high level prior stage takes feature maps from previous shared convolutional layers.This stage consist of 4 convolutional layers with a PReLU after every layer.
- The first two layers are followed by max pooling layers with a stride of 2.
- At the end, the high-level stage of three fully functional connected layers with PReLU activation function after every layer.
- The first FC layer consist of 512 neurons.
- The Second FC layer consist of 256 neurons.
- The final layer consist of a set 10 neurons followed by a sigmoid layer indicating the count class of the input image.
- To enable the use of arbitrary sized images for training, spatial pyramid pooling is employed as it eliminates the fixed size constraints of the deep neural network.
- **Density Estimation stage :** The feature maps obtained from the shared layers are processed by another CNN network that is cascaded CNN.
- The first two layers are followed by max pooling layers with a structure of 2 due to which the output of CNN layer is downsampled by a factor of 4.
- The first filter convolutional layer has 20 feature maps with a filter size of 7X7.
- The second filter convolutional layer has 40 feature maps with a filter size of 5X5.
- The third  filter convolutional layer has 20 feature maps with a filter size of 5X5.

- The last filter convolutional layer has 10 feature maps with a filter size of 5X5.
- The output of this network is combined with that id the last convolutional layer of high-level prior stage using a set of 2 convolutional and 2 fractionally convolutional layers.
- The first two convolutional layers have a filter size of 3X3 with 24 and 32 feature maps respectively.
- These layers are followed by 2 sets fractionally strided convolutional layer with 16 and 18 feature maps.
- In addition to integrating high-level prior from an earlier stage,the fractionally strided convolutions learns to upsample the feature maps to the original image size thereby restoring the details lost due to the earlier max pooling layers.
- The use of these layers results in upsampling of CNN output by a factor of 4, thus enabling us to regress in Full resolution density maps.

**6.3 Training and Implementation.**

To create the training dataset, patches of size 1/4 th the size of original image are cropped from 100 random locations. Other augmentation techniques like horizontal flipping and noise addition are used to create another 200 patches. The random cropping and augmentation resulted in a total of 300 patches per image in the training dataset.

Ground truth density map $D_i$
corresponding to the i th training patch is calculated by summing a 2D Gaussian kernel centered at every person's location x g as defined below:

$$D_i(x) = \sum_{Xg \in S}^{N} N(X - Xg, \sigma) \ ,$$

where σ is the scale parameter of the 2D Gaussian kernel
and S is the set of all points at which people are located.

**Following are the results on Shanghaitech A and B dataset:**

Shanghaitech A MAE: 101 MSE: 148
Shanghaitech B MAE: 17 MSE: 29

# 7 Crowd Density Estimation Using CSRNet.

**CSRNet architecture**

we choose VGG as the front-end of CSRNet because of its strong transfer learning ability and its flexible architecture for easily concatenating the back-end for density map generation.In CrowdNet the authors directly carve the first 13 layers from VGG-16 and add a 1 × 1 convolutional layer as output layer. The absence of modifications results in very weak performance. Other architectures, uses VGG-16 as the density level classifier for labeling input images before sending them to the most suitable column of the MCNN, the VGG-16 performs as an ancillary without significantly boosting the final accuracy. In this architecture implementation we first remove the classification part of VGG-16 (fully-connected layers) and build the proposed CSRNet with convolutional layers in VGG-16. The output size of

this front-end network is 1/8 of the original input size. If we continue to stack more convolutional layers and pooling layers (basic components in VGG-16), output size would be further shrunken, and it is hard to generate high-quality density maps.we try to deploy dilated convolutional layers as the back-end for extracting deeper information of saliency as well as maintaining the output resolution.

**7.1 Dilated convolution**

One of the critical components of our design is the dilated convolutional layer. A 2-D dilated convolution can be defined as follow:

$$y(m, n) = \sum_{i=1}^{M} \sum_{j=1}^{N} x(m + r \times i, n + r \times j)w(i, j)$$

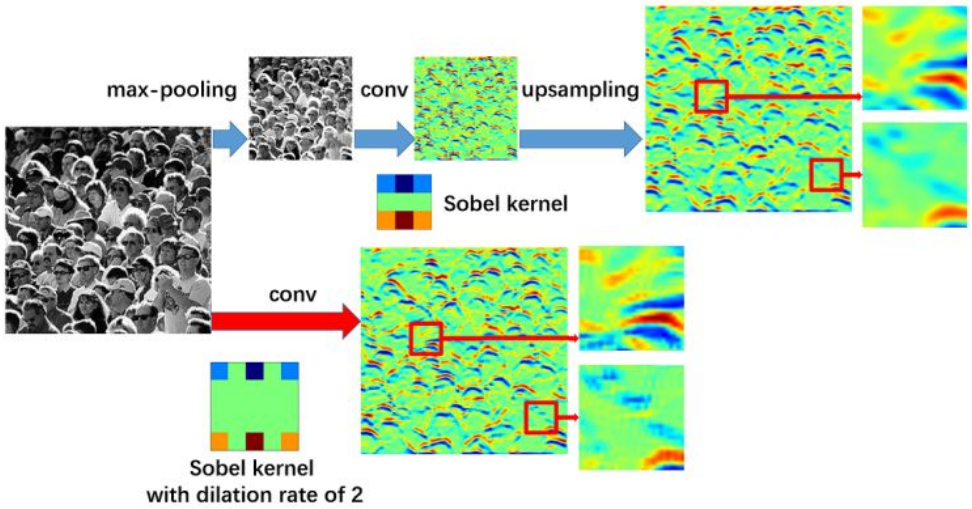
y(m, n) is the output of dilated convolution from input x(m, n) and a filter w(i, j) with the length and the width of M and N respectively. The parameter r is the dilation rate. If r = 1, a dilated convolution turns into a normal Convolution. Dilated convolutional layers have been demonstrated in segmentation tasks with significant improvement of accuracy and it is a good alternative of pooling layer.

- Dilated convolution is a better choice, which uses sparse kernels to alternate the pooling and convolutional layer. This character enlarges the receptive field without increasing the number of parameters or the

amount of computation (e.g., adding more convolutional layers can make larger receptive fields but introduce more operations).

- In dilated convolution, a small-size kernel with k × k filter is enlarged to k + (k −1)(r − 1) with dilated stride r. Thus it allows flexible aggregation of the multi-scale contextual information while keeping the same resolution.

- For maintaining the resolution of feature map, the dilated convolution shows distinct advantages compared to the scheme of using convolution + pooling + deconvolution. The input is an image of crowds, and it is processed by two approaches separately for generating output with the same size. In the first approach, input is downsampled by a max pooling layer with factor 2, and then it is passed to a convolutional layer with a 3 × 3 Sobel kernel. Since the generated feature map is only 1/2 of the original input, it needs to be upsampled by the deconvolutional layer (bilinear interpolation).
- In the other approach, we try dilated convolution and adapt the same 3 × 3 Sobel kernel to a dilated kernel with a Figure 4. Comparison between dilated convolution and maxpooling, convolution, upsampling. The 3 × 3 Sobel kernel is used in both operations while the dilation rate is 2. factor = 2 stride. The output is shared the same dimension as the input (meaning pooling and deconvolutional layers are not required). Most importantly, the output from dilated convolution contains more detailed information (referring to the portions we zoom in).



### 7.2 Network Configuration

- In the front-end,we adapt a VGG-16 network  (except fully-connected - layers) and only use 3 × 3 kernels.
- Using more convolutional layers with small kernels is more efficient than using fewer layers with larger kernels when targeting the same size of receptive field .By removing the fully-connected layers, we try to determine the number of layers we need to use from VGG-16.
- The most critical part relays on the tradeoff between accuracy and the resource overhead (including training time,memory consumption, and the number of parameters).
- Since the output (density maps) of CSRNet is smaller (1/8 of input size), we choose bilinear interpolation with the factor of 8 for scaling and make sure the output shares the same resolution as the input image. With the same size, CSRNet generated results are comparable with the ground truth results using the PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity in Image).

**Following are the results on Shanghaitech A and B dataset:**

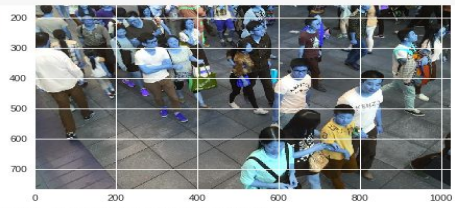
Shanghaitech A  MAE: 66.4  MSE: 115.0

Shanghaitech B MAE: 10.6 MSE: 16.0

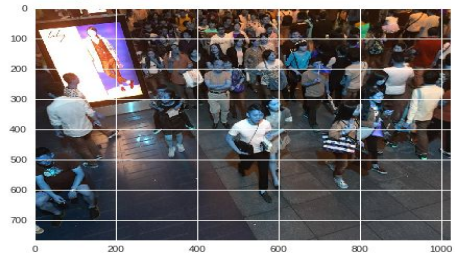
```
import os
def is_image_file(fname):
    ext = os.path.splitext(fname)[-1].lower()
    return ext == '.jpg'
data_path = '/content/drive/ml/CSRNet-mx/data/shanghaitech/part_B_final/test_data/'
image_names = list(filter(is_image_file, os.listdir(os.path.join(data_path, 'images'))))
image_paths = [os.path.join(data_path, 'images', e) for e in image_names]

fout = open('predict.txt', 'w')
num_images = len(image_names)
for i, fname in enumerate(image_paths):
    img_name = image_names[i]
    im = cv2.imread(fname)
    rows, cols, ts = im.shape
    imgs = im.transpose((2,0,1)).reshape((1, 3, rows, cols)) - MEAN_COLOR
    plt.imshow(im)
    plt.savefig(fname,format='png',bbox_inches=None, pad_inches=0.1,frameon=None)
    plt.show()
    ps = predict(imgs)
    p = ps[0]
    print ('{}/{}: {} {}'.format(i + 1, num_images, img_name, p))
    fout.write('{} {}\n'.format(img_name, p))

    i += 1
```



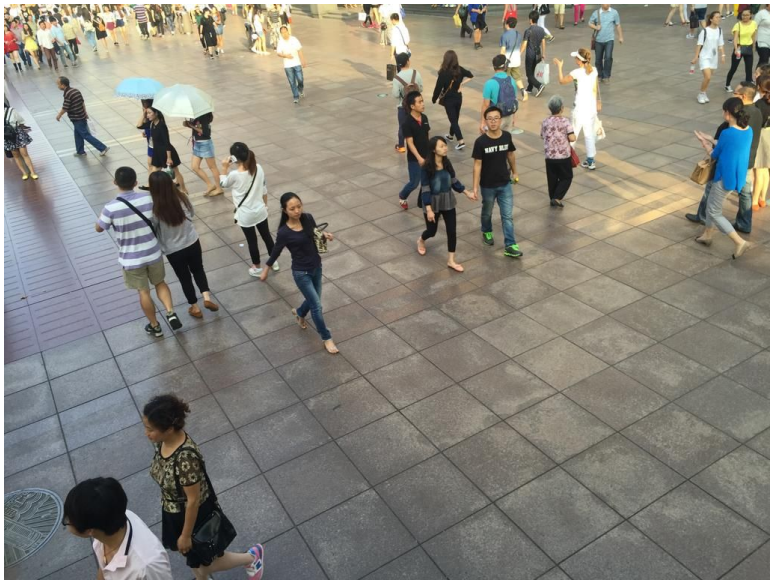
```
2/18: IMG_4.jpg 88.70083618164062
```



```
3/18: IMG_5.jpg 68.04550170898438
```

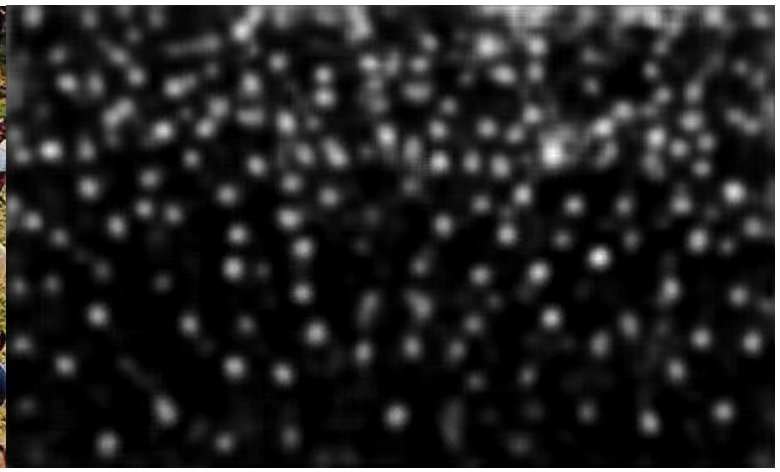
# 8 Results of both Cascaded CNN and CSRNet Method.

Original-Image Estimated with ground-truth 22.90126



Original-Image Estimated With ground-truth 57.299644

Result of Cascaded CNN - ShanghaiTech Dataset Test Images - Part-B



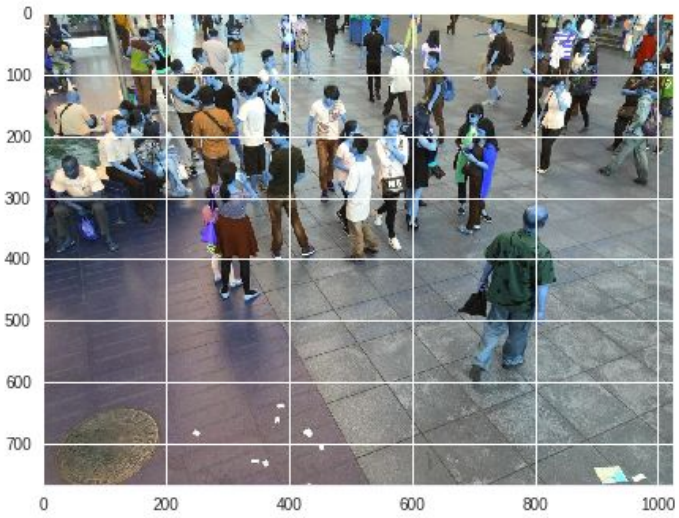
Ground-truth 222.89313

Estimated count 199.25372

Ground-truth 430.13126

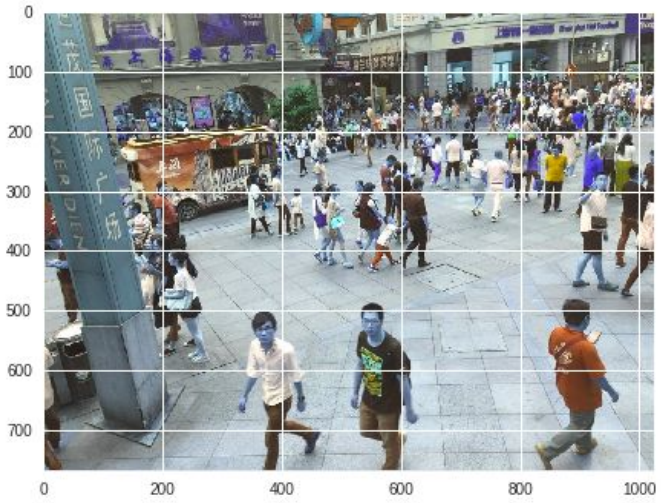
Estimated count 441.87888

Result of Cascaded CNN - ShanghaiTech Dataset Test Images - Part-A




Original Image

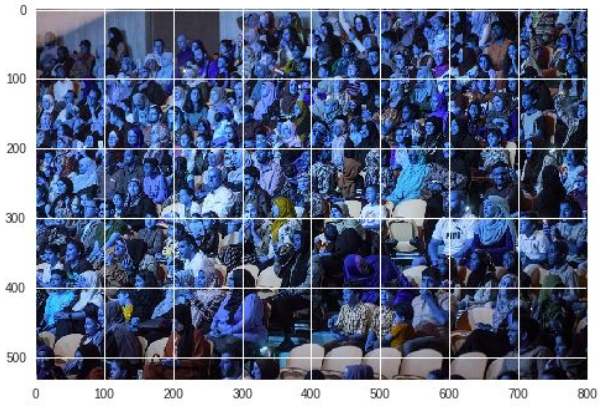
Predicted : 50.02176284790039




Original Image

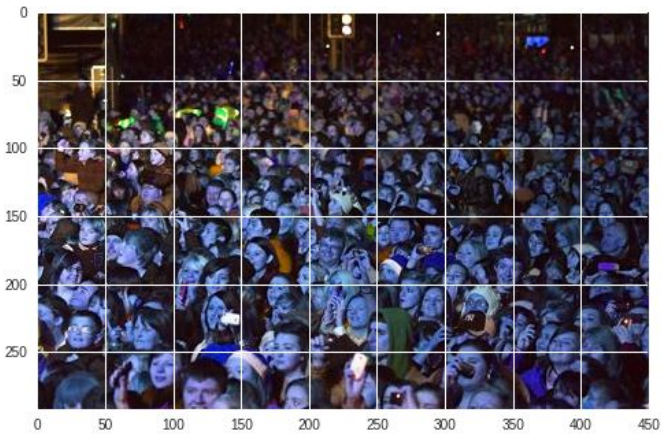
Predicted :256.233154296875

Test Result of CSRNet for Custom-Test Data-Images -Part_B




Original Image

Predicted: 242.94796752929688




Original Image

Predicted: 388.4822998046875

Test Result of CSRNet  for Custom-Test Data-Images -Part_A

**Training**

- ShanghaiTech crowd counting dataset contains 1198 annotated images with a total amount of 330,165 persons.This dataset consists of two parts as Part A containing 482 images with highly congested scenes randomly downloaded from the Internet while Part B includes 716 images with relatively sparse crowd scenes taken from streets in Shanghai.
- The loss function used are Mean Squared Error and Mean Absolute Error and  have used Adam Optimizer to minimize our loss functions and find appropriate values of trainable parameters.
- In Cascaded CNN crowd estimation the errors are:
- Shanghaitech A  MAE: 101  MSE: 148
- Shanghaitech B MAE: 17 MSE: 29
- In CSRNet Architecture crowd estimation Errors are:
- Shanghaitech A  MAE: 66.4  MSE: 115.0
- Shanghaitech B MAE: 10.6 MSE: 16.0
- Trained 110 Model of Cascaded CNN in Nvidia-950m in Pytorch.
- And Also CSRNets models prediction test on custom dataset is almost accurate and less lossy than previous methods.The model was saved in caffe framework.

# 10 References:

## Research Paper Used and Implemented

**CSRNet: Dilated Convolutional Neural Networks for Understanding the Highly Congested Scenes - Yuhong Li ，Xiaofan Zhang ，Deming Chen - 11-apr-2018**
https://arxiv.org/pdf/1802.10062.pdf

**CNN-based Cascaded Multi-task Learning of High-level Prior and Density Estimation for Crowd Counting - Vishwanath A. Sindagi Vishal M. Patel - 16-aug-2017**
https://arxiv.org/pdf/1707.09605.pdf

**MULTI-SCALE CONVOLUTIONAL NEURAL NETWORKS FOR CROWD COUNTING - Lingke Zeng, Xiangmin Xu, Bolun Cai, Suo Qiu, Tong Zhang - 8-Feb-2017**
https://arxiv.org/pdf/1702.02359.pdf

**Concepts and Frameworks Used for this Project:**

**Deep-learning Frameworks**
Tensorflow
Keras
Pytorch
Caffe

**Coding Environment and Editor**
Jupyter Notebook
Python 3m
Python 2m - CSRNet

**GPU**
GPU-NVIDIA-950M

**Dataset**
Shanghaitech Crowd-Dataset