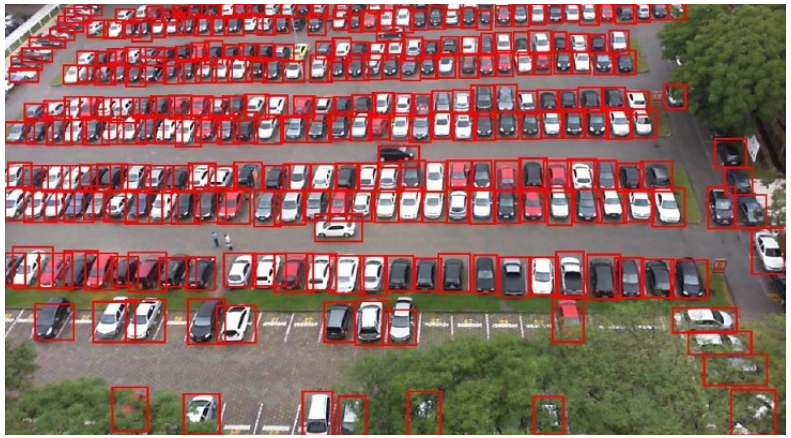
Context-Aware Crowd Counting

Ananya Sankaranarayanan - MDS202105
Pragya Jaiswal - MDS202129
Prashant Bajpai - MDS202130

Crowd Counting

Crowd counting is the process of estimating the number of *individuals* or *objects* present in a given area or scene from visual data, such as images or videos.

(a) Counting humans on a street



(b) Counting cars in a parking lot

Applications of Crowd Counting:

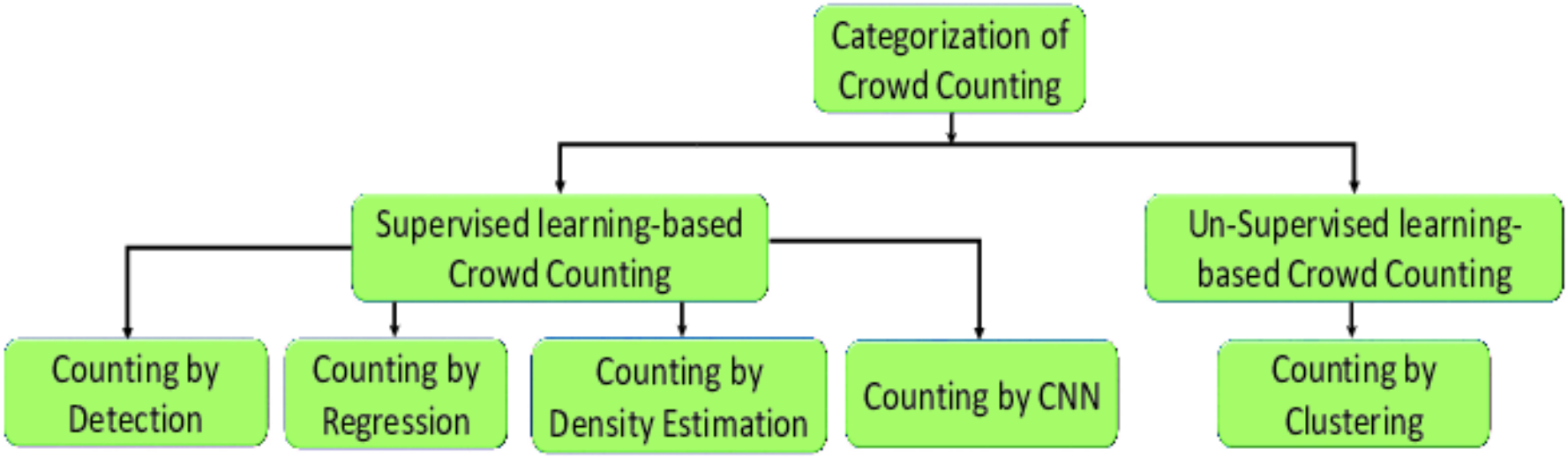
- **Public safety:** Monitor and manage crowds at events and public spaces to ensure public safety. Prevents overcrowding, reduce the risk of stampedes, enables prompt responses to emergencies.
- **Transportation:** Monitor passenger volumes at train stations, airports, and other transportation hubs.
- **Retail:** Optimize staffing levels and improve customer service.
- **Event planning:** To ensure that the event space is large enough, there are enough resources (e.g., restrooms, food and beverage vendors, security personnel., etc.

Why is it challenging?

- High degree of variation in the appearance of people;
- Lighting conditions;
- Density of the crowd;

As a result, it requires sophisticated algorithms and machine learning models that can accurately detect and count people in different scenarios.

Counting Techniques



Counting by Detection

- **Monolithic Detection:** It trains the classifier using the full-body appearance that's available in the training images using typical features such as Haar wavelets, gradient-based features such as a histogram of oriented gradient (HOG), etc
- **Part-based detection:** This technique considers a part, say head or shoulders and applies a classifier to it.
- **Shape Matching:** Ellipses are considered to draw boundaries around humans, and then a stochastic process is used to estimate the number and shape configuration.

Counting by Detection

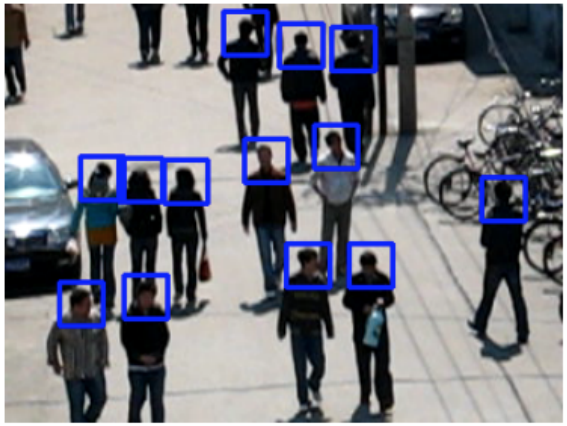




Figure: (a)Monolithic detection (b)Part-based detection (c)Shape matching detection

Counting by Regression

Counting by detection is not very accurate when the crowd is dense and the background clutter is high. Typically, this method consists of two main components.

- Extracting low-level features, such as Foreground features, texture, edge features, and gradient features;
- Mapping in a regression function, e.g., linear regression, piece-wise linear regression, ridge regression, or *Gaussian process regression(2008)*, to map the extracted features into counts.

Counting by Regression

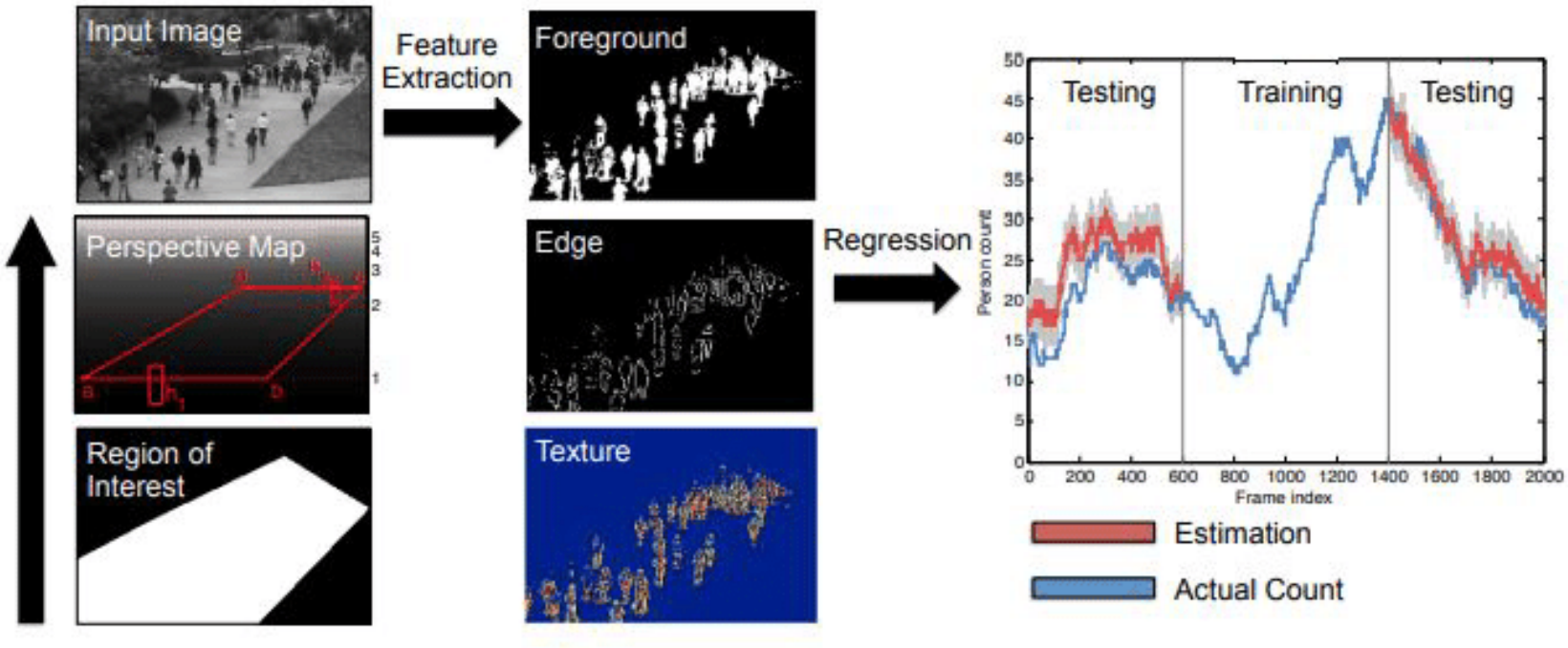


Figure: Crowd counting pipeline using regression model

Counting by Density Estimation

This approach focuses on the **density** by learning the mapping between *local features* and *object density maps*, thereby incorporating **spatial information** in the process. This avoids learning each individual separately and therefore tracks a group of individuals at a time.

Counting by Density Estimation

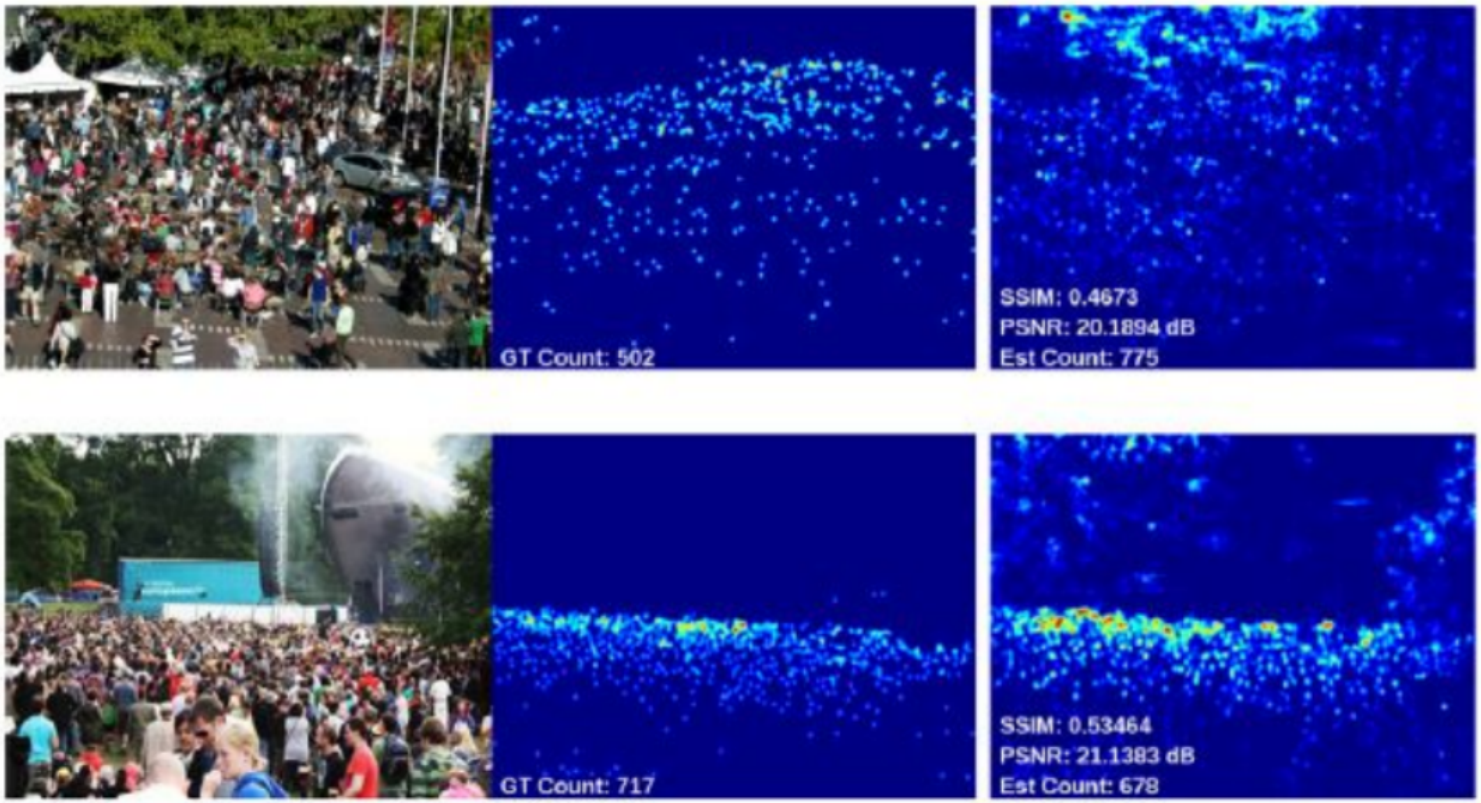


Figure: Crowd counting pipeline using regression model

Counting by CNN

Counting through CNN employs convolution, pooling, Rectified Linear Unit (RelU), and Fully Connected Layers (FCLs) to extract features that are used to obtain the density map. Counting through CNN is more efficient in terms of accuracy, but at the cost of high computational complexity.

Counting by Clustering

Counting by clustering relies on the assumption that visual features and individual motion fields are uniform, so similar features are grouped into different categories. Counting by clustering performs better in continuous image frames. In a paper titled, Counting crowded moving objects, **Kanade–Lucas–Tomasi (KLT) tracker**(2007) was used to obtain low-level features, and then Bayesian clustering was employed to find the approximate number of people in an image.

About the paper

We are going to look at a paper titled "Context-Aware Crowd Counting" by Weizhe Liu, Mathieu Salzmann, Pascal Fua.

- Submitted to the arXiv on 26 Nov 2018 (v1) and last revised on 15 Apr 2019 (v2).
- It was published in the International Journal of Computer Vision (IJCV) in May 2020.

About the paper

- In this paper, we aim to exploit **context**, that is, the large-scale consistencies that often appear in images.
- We introduce a new deep net architecture that adaptively encodes **multi-level contextual information** into the features it produces.
- We then show how to use these **scale-aware features** to regress to a final density map.

Objective

- Our aim is to get a density map from an image.
- For a set of N training images $\{I_i\}_{1 \leqslant i \leqslant N}$, we are given the corresponding ground truth density maps $\{D_i^{gt}\}$.
- The goal is to learn a non-linear mapping $\mathcal{F}$ that maps an input image I_i to an estimated density map $D_i^{est}(I_i) = \mathcal{F}(I_i, \theta_i)$ such that the L^2 norm is minimum.

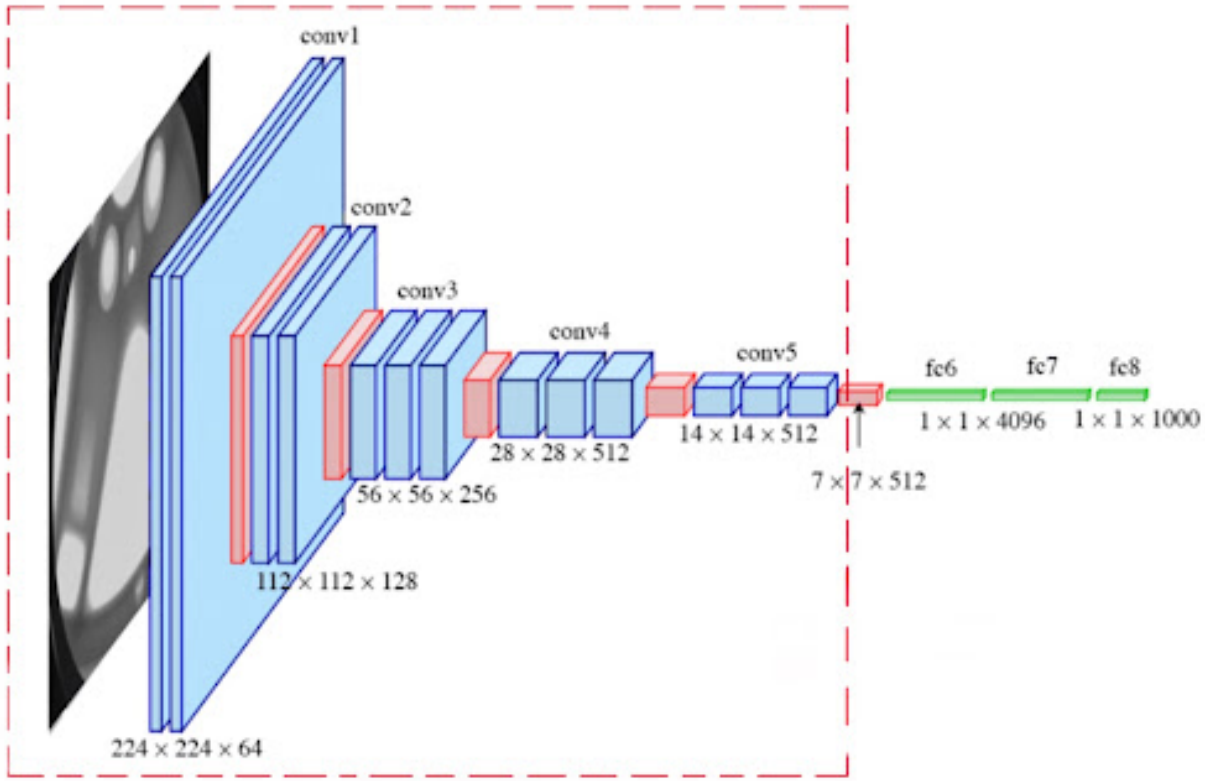
Architecture

The network architecture used in this paper can be broadly divided into three frameworks :

- Backbone Network
- Context Network
- Decoder Network

Architecture - Backbone Network

This network is based on the first 10 layers of VGG-16 network architecture and is responsible for feature extraction from the input image.



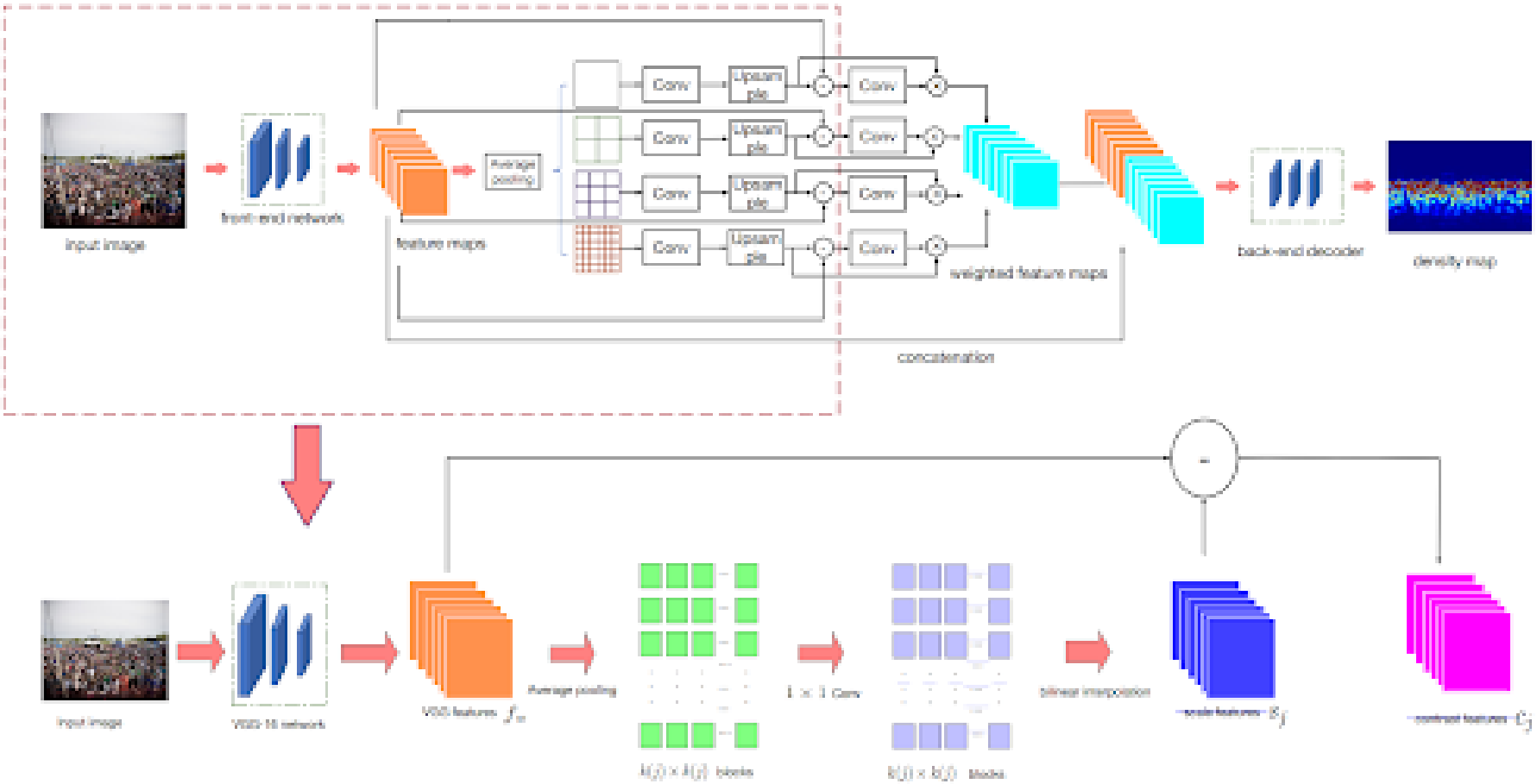
Architecture - Backbone Network

Given an image I, the first 10 layers of the VGG-16 network outputs features of the form

$$f_v = \mathcal{F}_{vgg}(I)$$

The limitation of $\mathcal{F}_{vgg}$ is that it encodes the same receptive field over the entire image.
To account for different scales, the context network is used to extract multi-scale context information from the VGG features.

Architecture - Context Network



Architecture - Context Network

Scale-aware features: $s_j = U_{bi}(\mathcal{F}_j(P_{ave}(f_v, j), \theta_j))$

- A Spatial Pyramid Pooling layer is added on top of the backbone network to divide the input feature map into a fixed number of sub-regions.
- Average pooling is applied to each sub-region.
- This process is repeated for different grid sizes or scales, creating a multi-level pyramid of output vectors. The four different scales are generally of block sizes $\{1, 2, 3, 6\}$.
- $\mathcal{F}_j$ is a convolutional network with kernel size 1 to combine the context features across channels without changing their dimensions.
- The feature outputs are upsampled using bilinear interpolation to make it the same dimension as f_v.

Architecture - Context Network

The scale-aware features are used to find the contrast features c_j as:

$$c_j = s_j - f_v$$

These are then used as input to another network with weights θ^j_{sa} that compute the weights ω_j assigned to each one of the scales used.

$$\omega_j = \mathcal{F}^j_{sa}(c_j, \theta^j_{sa})$$

$\mathcal{F}^j_{sa}$ is a 1x1 convolutional layer followed by sigmoid function. The final contextual features for an image I is defined as

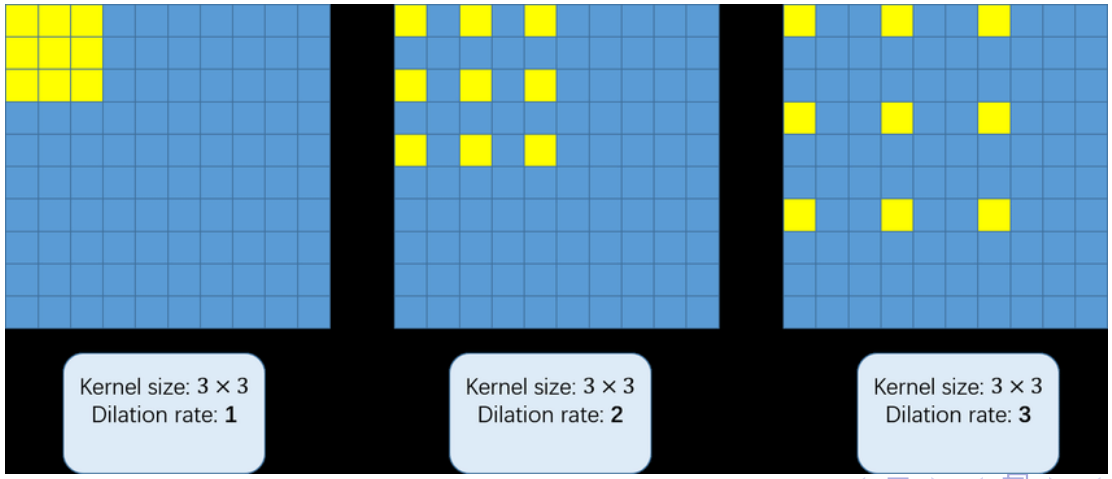
$$f_I = [f_v | \frac{\sum_{j=1}^{S} \omega_j \odot s_j}{\sum_{j=1}^{S} \omega_j}]$$

where $[\cdot|\cdot]$ denotes the channel-wise concatenation operation, and $\odot$ is the element-wise product between a weight map and a feature map.

Architecture - Decoder Network

The contextual features f_l are passed to a decoder consisting of several dilated convolutions. The output from this is the required density map.

Dilated convolutions introduce gaps or spacing between the values in the kernel used for convolution. This spacing is determined by the dilation rate. It can be thought of as "stretching" the filter by inserting empty spaces (or zeros) between the values in the filter which allows the filter to cover a larger receptive field, without increasing the size of the filter.



Training Details

- The ground truth density maps D_i^{gt} are obtained using manually annotated data. That is, each image I_i is associated to a set of points P_i^{gt} that denote the position of each human head in the scene. An image with ones in these positions and zeroes elsewhere are convolved with a Gaussian kernel $N^{gt}(p|\mu, \sigma^2)$ to get the ground truth density map.
- The network is trained using L^2 loss function.

$$L(\theta) = \frac{1}{2B}\Sigma_{i=1}^{B}||D_i^{gt} - D_i^{est}||_2^2$$

B - Batch size

- To minimize the loss, we use Adam optimizer with a batch size of 32.
- Random image patches of 1/4th the size of the original image are cropped at different locations. These patches are mirrored to double the training set.

Results

include original image with density maps

Penguin Dataset

- The Dataset consists of images of penguin colonies in Antarctica taken by fixed cameras in over 40 different locations.
- These pictures have been manually annotated by several different people for every image. We take the one of them for each image to generate the ground truth density maps.

Results - Penguin Dataset

include original image with density maps

Conclusion

References

1. Liu, W., Salzmann, M., Fua, P.V. (2018). Context-Aware Crowd Counting. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 5094-5103.
2. A Regression Based Model to Count Pedestrians in Crowds with Area, Shape and Texture Feature
3. Liu, Jiang Gao, Chenqiang Meng, Deyu Hauptmann, Alexander. (2017). DecideNet: Counting Varying Density Crowds Through Attention Guided Detection and Density Estimation.
4. Ilyas, Naveed Shahzad, Ahsan Kim, Kiseon. (2019). Convolutional-Neural Network-Based Image Crowd Counting: Review, Categorization, Analysis, and Performance Evaluation. Sensors (Basel, Switzerland). 20. 10.3390/s20010043.

References

5. Elbishlawi, Sherif Abdelpakey, Mohamed Eltantawy, Agwad Shehata, Mohamed Mohamed, Mostafa. (2020). Deep Learning-Based Crowd Scene Analysis Survey. Journal of Imaging. 6. 95. 10.3390/jimaging6090095.
6. https://encyclopedia.pub/entry/25356
7. https://www.analyticsvidhya.com/blog/2021/06/crowd-counting-using-deep-learning/