CrowdDiffKDE: Multi-hypothesis Crowd density Estimation using Diffusion Models and Kernel Density Estimation

Aravind Seshadri* Hidangmayum Bebina Prof.Tushar Sandhan
Electrical Engineering, Indian Institute of technology Kanpur

Background

- Crowddiff uses a diffusion model to learn the crowd map generation by conditioning on an image

- Due to the stochastic nature of diffusion models, multiple realizations can be obtained. These are then fused to improve the counting performance.

.

- The generation of multiple realizations is a time-intensive process. Each new realization requires additional processing due to the inherent computational complexity of diffusion models. Can the speed of this process be improved without compromising accuracy?

Introduction

Realizations from model

- From sample realizations, we observe a degree of similarity between any two given realizations.
- This similarity can be quantified by estimating the probability density of the points.
- We make use of Kernel Density Estimation for this purpose.

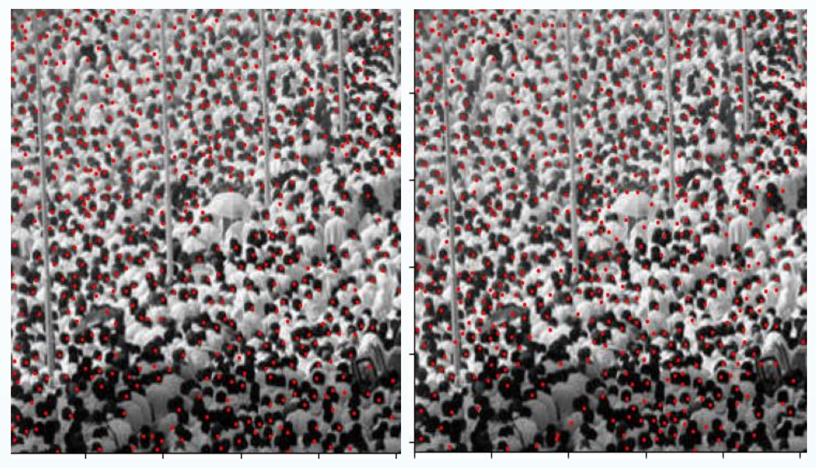
$$\hat{f}(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right)$$

- In KDE, we use a Gaussian kernel to model the density function.
- After learning the probabilistic distribution of points in a map, new points can be sampled and fused to obtain the final map.

Results

Original	JHU_Crowd	Shtech-A	Shtech-B	UCF-CC	UCF-QNRF
MAE	51.82	46.35	6.85	128.202	77.4
MSE	203.194	81.25	12.2	261.28	117.103

Count using multiple Realizations: 616. Ground Truth: 783



Predicted Count using KDE: 653 Ground Truth: 783

Proposed	JHU_Crowd	Shtech-A	Shtech-B	UCF-CC	UCF-QNRF
MAE	52.96	44.29	7.02	87.22	70.19
MSE	201.32	79.24	13.19	196.98	99.32

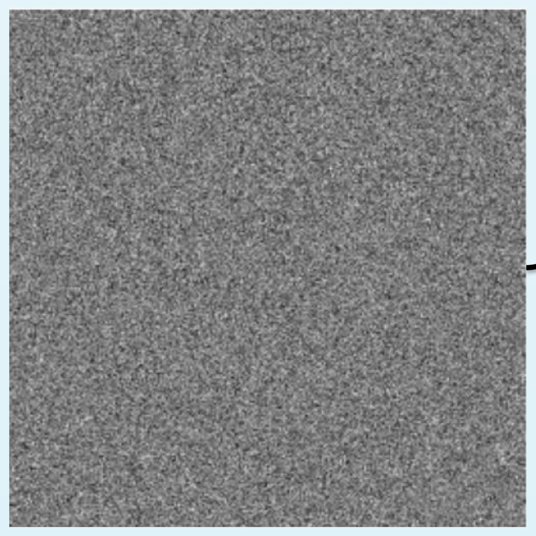
Conclusion

- Fusing with KDE improves accuracy in high-density images, such as in UCF-CC and reduces computation.
- For low-density images, such as Shtech-B KDE may reduce accuracy by predicting incorrect locations.

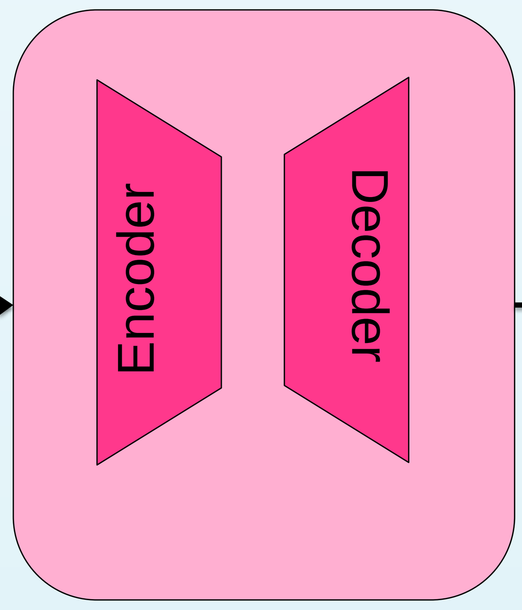
Methodology

Condition Model

Noise Realization

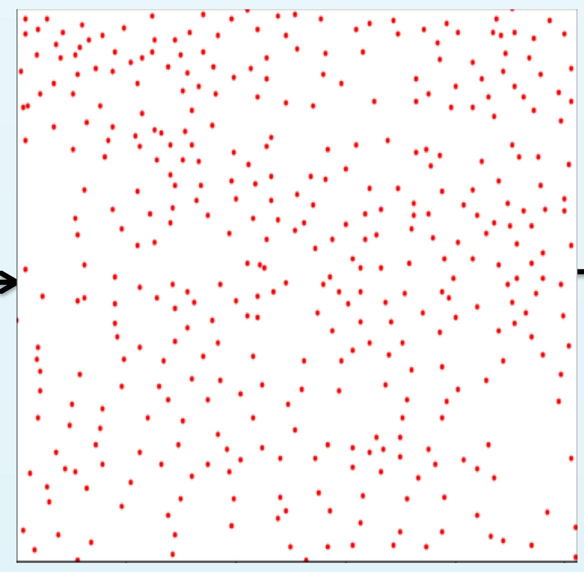


Diffusion Process



Unet x T time steps

Crowd Map Generated for One Realization

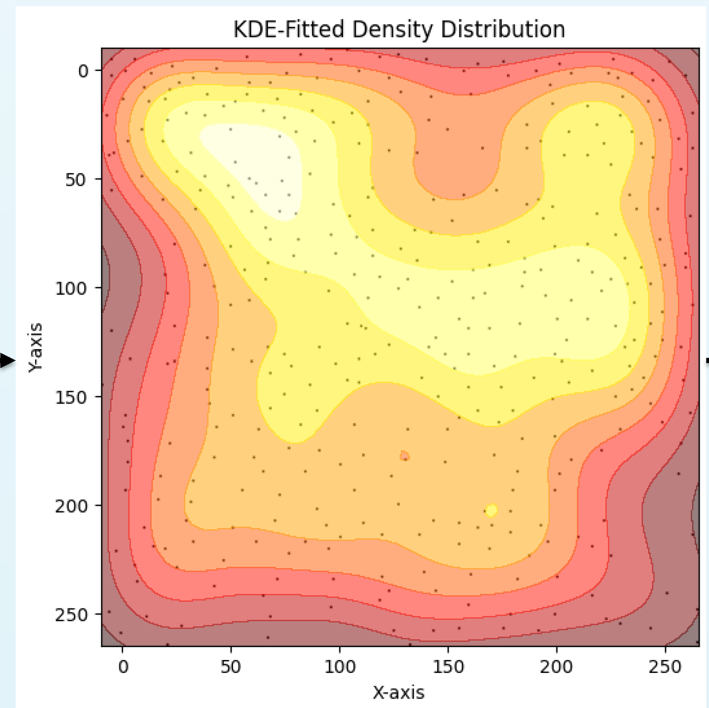


Generating more realizations is computationally expensive

Kernel Density Estimator

Gaussian Kernel and appropriate bandwidth is chosen

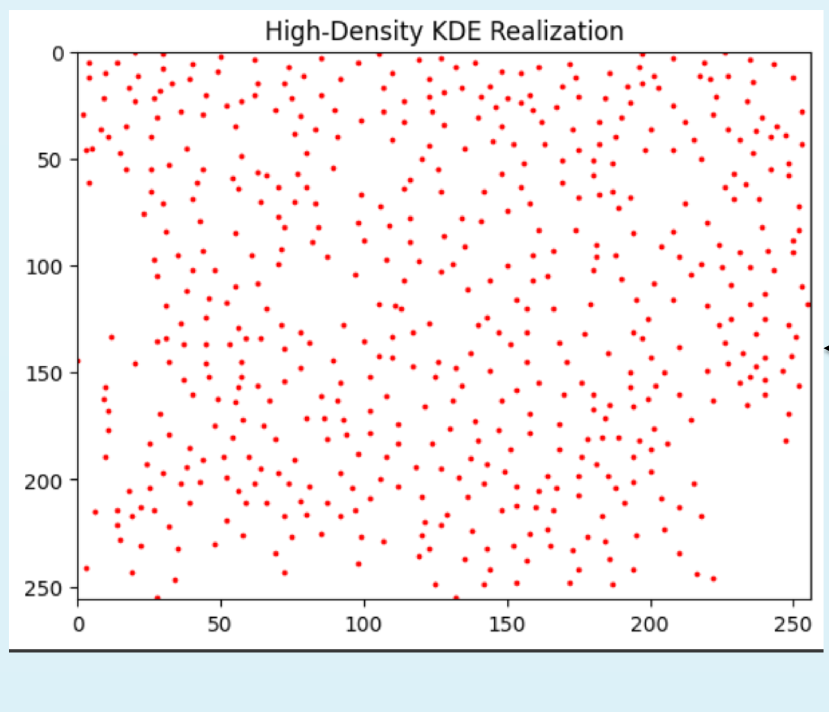
Density Distribution Map



Sampling is done 4* Initial Count of points

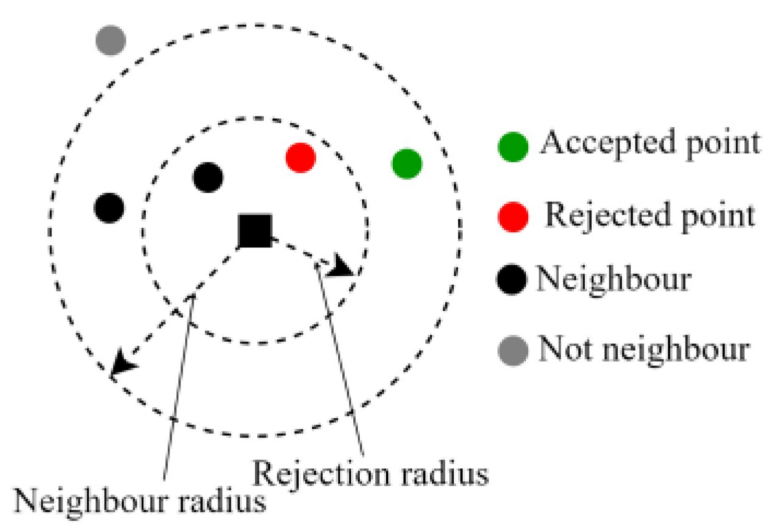
Sample and Fuse

```
Function Sample_and_Fuse(crowd_map):
    head_points = denoise_and_threshold(crowd_map)
    KDE = kde.fit(head_points)
    Set initial values for k and β and remove outliers
    Neighbours = NearestNeighbours(k + 1)
    Neighbours.fit(head_points)
    Set initial radius r by the definition of rejection radius

    For each point in 4 * head_points_count:
        dist = euclidean(point - head_points)
        If dist > r for all head_points:
            Add point to head_points
            Neighbours.fit(head_points)
            Update r, k, and β as necessary

    Return head_points
```



Final Fused Samples

Fusion criteria



Rejection Radius r

$$r_n = \frac{\beta}{2k}\sum_{i=1}^{k} r_{in}$$