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

Github Repo Link

Introduction

The Crowddiff [1] authors present a novel, state-of-the-art approach for generating crowd maps using a diffusion model. First, a narrow kernel convolves with point data, and a threshold is applied to the resulting map, which is then used as ground truth for the model-generated map.
Additionally, they perform stochastic crowd map fusion, combining multiple realizations from the diffusion model using SSIM.
Due to the high computational complexity of diffusion models, generating multiple realizations leads to high inference times. To address this issue, I propose using kernel density estimation [2] to produce new crowd map realizations. This approach significantly reduces computation time while generating new realizations quickly.

Implementation

Kernel density fit: Using the Crowddiff [1] model, we generate a single crowd map, or realization, for an image, and the resulting points are then thresholded. A Gaussian kernel is applied to fit a kernel density function, modeling the probability distribution across each point in the crowd map. The bandwidth is carefully chosen to achieve an optimal spread, ensuring a smooth distribution without restricting it to specific points.
Removing outliers: The generated points from the model may include outliers or noisy points with extremely low-density values in the kernel density estimate. We remove those in the bottom 5th percentile to filter these points, effectively discarding the least probable points and improving the overall quality of the generated crowd map.
Sample and fusion: Using the learned density distribution, we can sample new points and realizations that could be included in the density map.

1. First, we randomly sample four times the initial number of points from the crowd map. These sampled points are then filtered to ensure that their density distribution falls within the mid-range.
2. We apply the k-nearest neighbors algorithm on the initial points to calculate a radius of rejection for each point.
3. For each newly generated sample, we compute its distance to each point in the map.
4. If a sample lies outside all radii of rejection for the existing head locations in the map, it is included; otherwise, it is discarded.
5. If the point is included, we repeat this process for each sample on the updated crowd map, now containing the initial points and the newly added samples. Additionally, beta and k can be updated to avoid overcounting.

Pseudocode

Algorithm 1 Fuse_With_KDE

```
function FUSE_WITH_KDE(crowd_map)
    head_points = DENOISE_AND_THRESHOLD(crowd_map)
    KDE = KDE.FIT(head_points)
    define k and β value and their respective increments
    Neighbours = NEARESTNEIGHBOURS(k + 1)
    NEIGHBOURS.FIT(head_points)
    r = calculate initial radius
    for each point in 4 x head_points_count do
        dist = EUCLIDEAN(point - head_points)
        if dist > r for all head_points then
            APPEND(head_points, point)
            NEIGHBOURS.FIT(head_points)
            UPDATE r, k and β as required
        end if
    end for
    return head_points
end function
```

Results

Metrics	JHU-Crowd	Shtech-A	Shtech-B	UCF_CC	UCF-QNRF
MAE	51.82	46.352	6.85	128.202	77.4
MSE	203.194	81.25	12.2	261.28	117.103

Table 1: Initial values for 5 benchmarks by using realizations from diffusion model

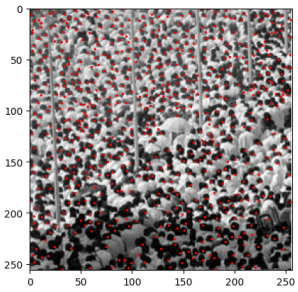
Metrics	JHU-Crowd	Shtech-A	Shtech-B	UCF-CC	UCF-QNRF
MAE	52.96	44.29	7.02	87.22	70.19
MSE	201.325	79.24	13.19	196.98	99.32

Table 2: Implemented values for 5 benchmarks using KDE

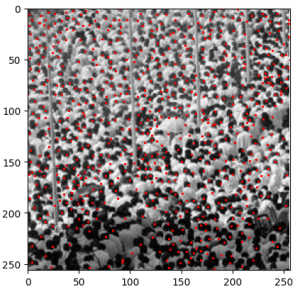
Analysis

- The computational complexity of the overall pipeline was reduced. Kernel density estimation, sampling, and fusion come with nearly no extra cost in time. In the original methodology to sample each new realization, it takes approximately 15 seconds with a GPU, however using KDE new realizations are obtained within a second.
- Fusing with KDE enhances accuracy for images with extremely high densities, such as those in UCF-CC, where crowd densities are high. This is expected due to the ability of KDE to sample as many points as required. A diffusion model generative capabilities will be limited to certain regions.

- However, for low-density images, such as shtech B, KDE does not improve accuracy and can result in incorrect location predictions. This is because KDE can sample random points that are not representative of the crowd in low-density regions.
- Further for datasets such as JHU-Crowd and Shtech-A, there is no significant improvement in the crowd count.
- An optimal bandwidth can improve the performance in low-density regions, however, it might degrade the performance in high density images.



Original Realizations Count with higher computation: 616



Modified with KDE Count has improved with Lower Computation: 653

Conclusion

The KDE method serves as a valuable alternative specifically in high-density regions. It provides an alternate solution to the issue of computational complexity while generating multiple instances in a diffusion model. It can further be tuned by varying β, k and the bandwidth.

References

[1] W. G. C. B. V. M. P. Yasiru Ranasinghe, Nithin Gopalakrishnan Nair, "Crowddiff: Multi-hypothesis crowd density estimation using diffusion models," *Computer Vision and Pattern Recognition*, 2024.

[2] J. Drapala. (2024) Kernel density estimation explained, step by step. *Towards Data Science*. Accessed: Oct. 28, 2024. [Online]. Available: https://towardsdatascience.com/kernel-density-estimation-explained-step-by-step-7cc5b5bc4517