CrowdDiff: Multi-hypothesis Crowd Density Estimation using Diffusion Models

Supplementary Material

A. Pseudocodes

The pseudocode for training is given in Algorithm A.1 and testing in Algorithm A.2 for CrowdDiff.

Algorithm A.1 Training phase

```
def train(images,density_maps,gt_counts):
    """
       images: [B, H, W, 3]
       density_maps: [B, H, W]
       gt_counts: [B,]
    """

    # Density scaling
    density_maps = (2*scale*density_maps-1)

    # Corrupt density_maps
    t = randint(0,T) # time step
    eps = normal(mean=0,std=1) # noise: [B,H,W]
    crpt_density_maps =
      diffusion_process(density_maps,eps,t)

    # Estimate noise and encoder-decoder features
    eps_pred, feats =
      denoising_network(images,crpt_density_maps,t)

    # Estimate crowd count
    count_est = counting_decoder(feats)

    # Compute denoising network loss
    loss =
      l_hybrid(eps_pred, eps) +
      count_scale * l1_loss(count_est,gt_count)

    return loss
```

Algorithm A.2 Testing phase

```
def testing(images, realizations):
    """
    images: [B, H, W, 3]
    realizations: N
    """

    # Encode image features
    feats = image_encoder(images)

    # noisy density maps: [B, H, W]
    density_pred = normal(mean=0, std=1)

    # uniform sample step size
    times = reversed(
    linespace(diffusion_steps, sampling_steps))

    # Perform DDIM sampling
    for t in times:
       # Predict noise from density_pred
       eps_hat = denoising_network(images,
            noisy_density, t)
       # Compute posterior of noisy density
       density_pred = q_posterior(noisy_density, eps,
             t)

    # Detect head locations
    locations = contours(density_pred) # [B, N, *, 2]

    # Perform crowd map fusion: [B, *, 2]
    final_locations = crowd_map_fusion(locations)

  # Compute crowd count
  return count(final_locations) # [B, ]
```

B. Experimental details

1. Denoising network architecture **Denoising network** has a U-Net architecture [4], and each downsampling and upsampling layer scales the features by a factor of two along each spatial dimension. We use average pooling for downsampling with a 2×2 kernel, a stride of 2, and nearest neighbor interpolation for upsampling. The 2-dimensional convolution layers are 3×3 kernels with a stride of 1, and the 1-dimensional convolution layers have a kernel size and a stride of 1. In the multi-head self-attention module, the channel dimension of each head is kept constant at 64, and the number of heads is varied according to the channel dimension of each depth level. The denoising network and the basic modules are illustrated in Fig. B.1.

Regression branch is a lightweight network with linear layers and a Rectified Linear Unit (ReLU) [1] activation. We apply global average pooling to maintain compatibility along the spatial dimension for channel-wise concatenation.

C. Evaluation metrics

To evaluate crowd counting performance, we use the mean absolute error (MAE) and mean squared error (MSE):

$$MAE = \frac{1}{N}\sum_{n=1}^{N}|c_n - \hat{c}_n|,$$

$$MSE = \sqrt{\frac{1}{N}\sum_{n=1}^{N}\|c_n - \hat{c}_n\|_2^2}$$

as the performance metrics. Here, N is the total number of test samples, c_n is the ground truth count, and $\hat{c}_n$ is the prediction for the n^{th} sample.

D. Datasets

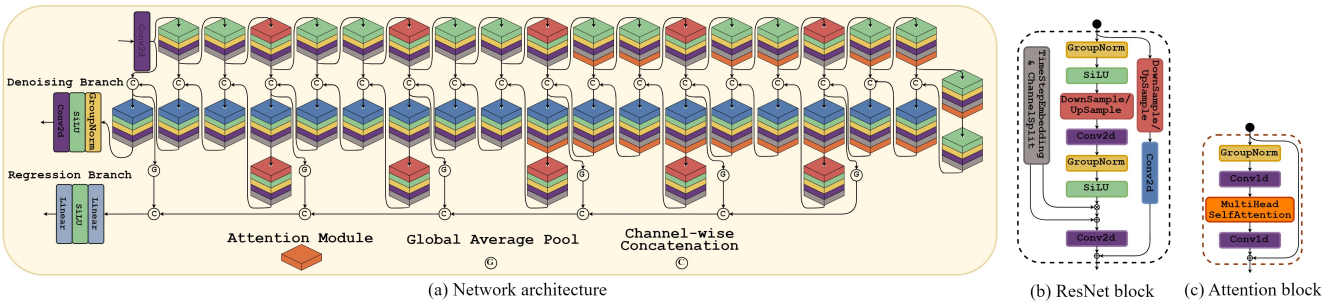
We evaluate our method on five public datasets: JHU-Crowd++[5], ShanghaiTech A[7], ShanghaiTech B[7], UCF-CC-50[2], and UCF-QNRF[3] for crowd counting.

JHU-Crowd++ [5] has 2,722 training images, 500 validation images, and 1,600 test images collected from diverse scenarios. The dataset consists of crowd images with numbers ranging up to 25,791 and images without any crowd.

ShanghaiTech A [7] contains 300 training images and 182 test images with annotations. We randomly select 30 samples from the training dataset as the validation dataset.

ShanghaiTech B [7] contains 400 training images and 316 testing images with annotations. We create a validation dataset with randomly selected 40 crowd images from the training dataset.

UCF-CC-50 [2] is a comparatively small crowd dataset



(a) Network architecture (b) ResNet block (c) Attention block

Figure B.1. (a) **Network architecture** for the denoising U-Net in conjunction with the count regression branch and the basic modules, (b) **ResNet block**, and (c) **Attention module**, used to construct the network. Each cuboid in a stack represents the functioning modules in the ResNet Block and whether the attention module is applied. **Top stacks** are in the encoder. **Bottom stacks** are in the decoder.

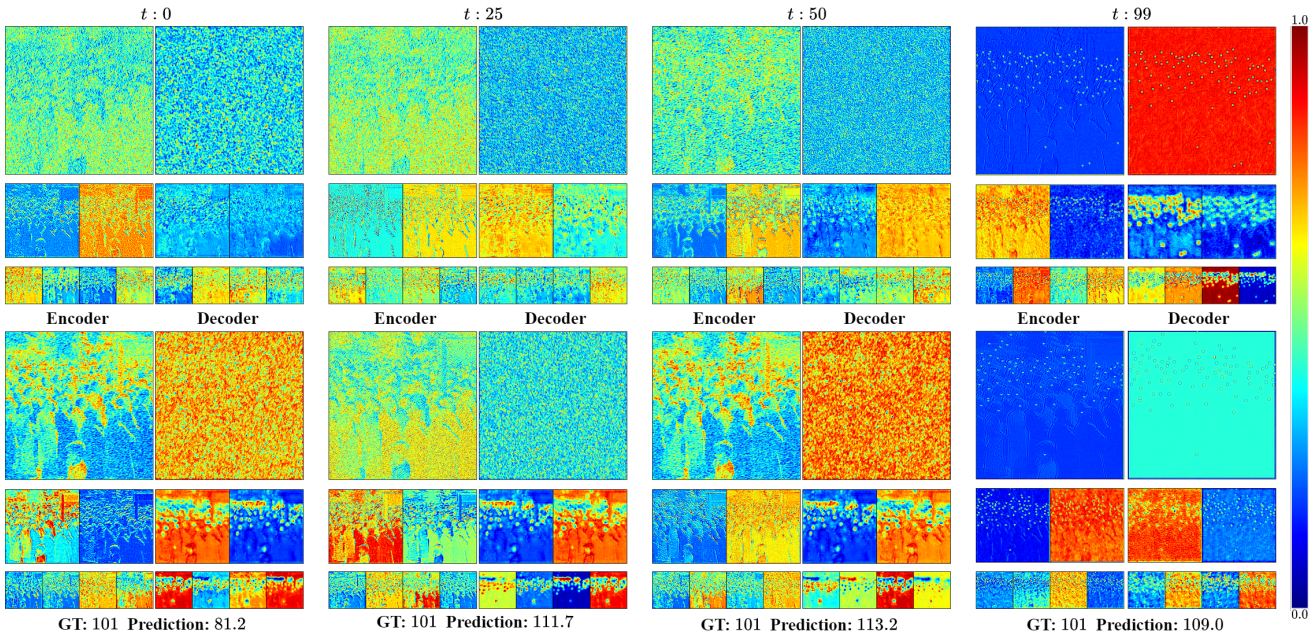


Figure C.1. Difference in feature maps without (*top row*) and with (*bottom row*) counting decoder.

for extremely dense crowd counting with just 50 samples. We perform a 5-fold cross-validation following the standard protocol in [2].

UCF-QNRF [3] dataset contains 1,535 images of unconstrained crowd scenes, with approximately one million annotations in total. The dataset is split into a training set of 1,201 images and a testing set of 334 images.

NWPU-Crowd NWPU-Crowd [6] is a large-scale dataset collected from various scenes, consisting of 5,109 images. The images are randomly split into training, validation, and test sets containing 3109, 500, and 1500 images, respectively. This dataset provides box-level annotations.

E. Additional qualitative results

We provide a qualitative comparison between the feature maps of the denoising U-Net with and without the counting branch prediction for different time steps in Fig. C.1. From Fig. C.1, we can see that the decoder features are richer in detail for the case with the counting branch than without it. With the counting branch, the decoder generates features for the crowd starting from the initial time step. The performance of the counting branch further clarifies this, as the predicted count has not varied with time and deviated from the ground truth count significantly.

References

[1] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018. 1

[2] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. Multi-source multi-scale counting in extremely dense crowd images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2547–2554, 2013. 1, 2

[3] Haroon Idrees, Muhmmad Tayyab, Kishan Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European conference on computer vision (ECCV)*, pages 532–546, 2018. 1, 2

[4] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International Conference on Machine Learning*, pages 8162–8171. PMLR, 2021. 1

[5] Vishwanath A Sindagi, Rajeev Yasarla, and Vishal M Patel. Pushing the frontiers of unconstrained crowd counting: New dataset and benchmark method. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1221–1231, 2019. 1

[6] Qi Wang, Junyu Gao, Wei Lin, and Xuelong Li. Nwpu-crowd: A large-scale benchmark for crowd counting and localization. *IEEE transactions on pattern analysis and machine intelligence*, 43(6):2141–2149, 2020. 2

[7] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 589–597, 2016. 1