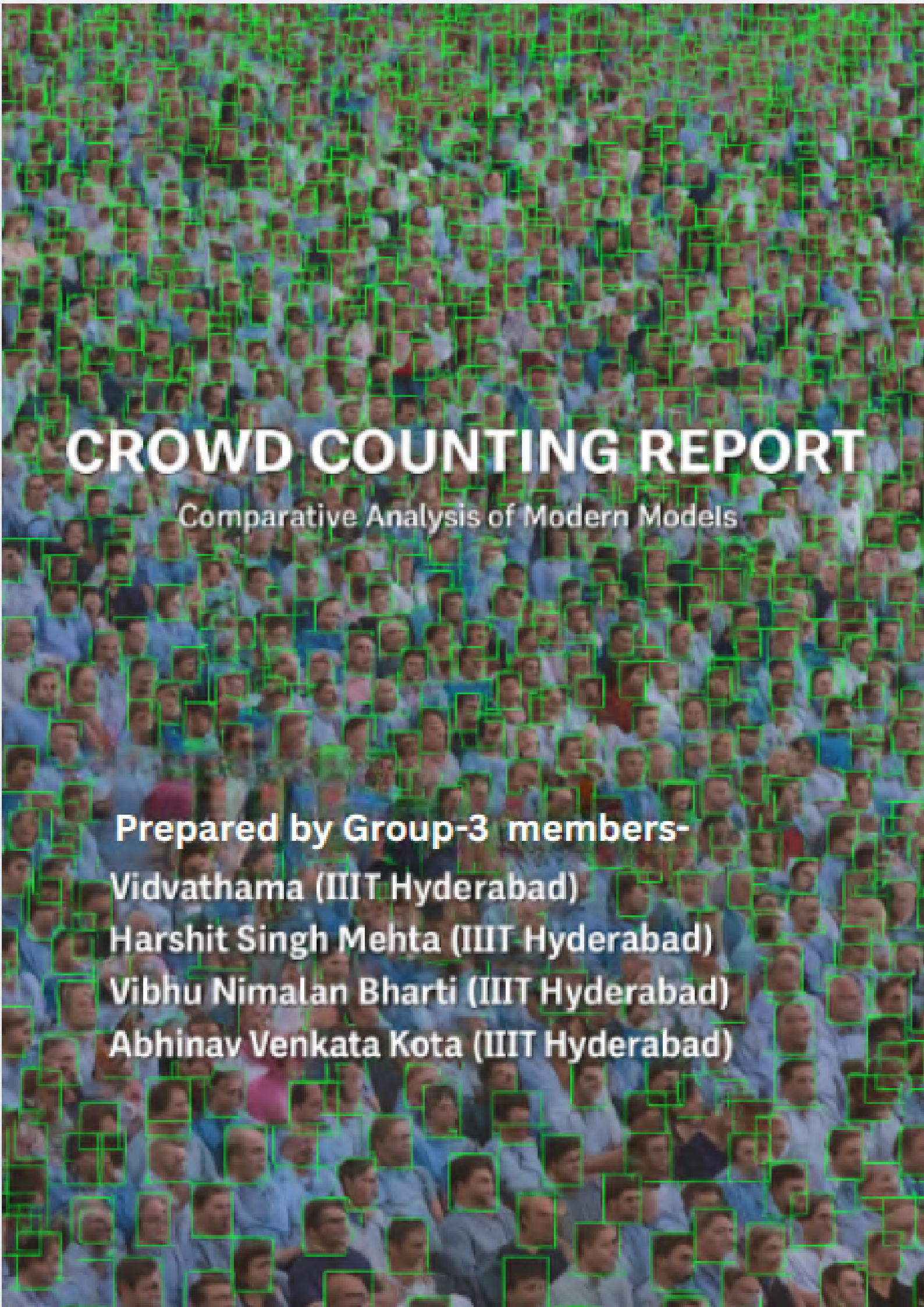


CROWD COUNTING REPORT

Comparative Analysis of Modern Models

Prepared by Group-3 members-

Vidvathama (IIIT Hyderabad)

Harshit Singh Mehta (IIIT Hyderabad)

Vibhu Nimalan Bharti (IIIT Hyderabad)

Abhinav Venkata Kota (IIIT Hyderabad)

Abstract

This report presents an overview of crowd counting, a computer vision task focused on estimating the number of people in images or video frames. It highlights the evolution of models from early traditional methods to modern deep learning-based approaches, including attention mechanisms and transformers. The report includes performance analysis on benchmark datasets and provides focused reviews of six key models: DM-Count, CCTrans, MPS, CHSNet, CrowdCLIP, and Crowd-Diff — each studied and implemented by our team.

Contents

1 What is Crowd Counting? 6

2 History of Crowd Counting Models 7
- 2.1 1. Pre-Deep Learning Era (2000–2015) . . . 7
- 2.2 2. Early Deep Learning (2015–2018) . . . 7
- 2.3 3. Modern Approaches (2019–Present) . . . 7

3 Focus of Our Reports 8

4 DM-Count: Distribution Matching for Crowd-Count 1
- 4.1 Abstract . . . 1
- 4.2 Introduction . . . 1
  - 4.2.1 Motivation . . . 1
  - 4.2.2 Why Fine-tuning vs. Training from Scratch? . . . 1
- 4.3 What DM-Count Does Better . . . 2
- 4.4 Dataset Preparation . . . 2
  - 4.4.1 Source and Size . . . 2
  - 4.4.2 Preprocessing . . . 2
  - 4.4.3 Train/Val/Test Splits . . . 3
  - 4.4.4 Data Augmentation . . . 3
- 4.5 Model Selection . . . 3
  - 4.5.1 Pre-trained Base Model . . . 3
  - 4.5.2 Justification . . . 4
  - 4.5.3 Modifications . . . 4
- 4.6 Model Architecture . . . 5
  - 4.6.1 Encoder: Hierarchical Feature Extraction . . . 5
  - 4.6.2 Bottleneck: Context Aggregation . . . 6
  - 4.6.3 Decoder: Density Map Reconstruction . . . 6
  - 4.6.4 Output and Counting . . . 6
- 4.7 Training Setup . . . 7
  - 4.7.1 Frameworks . . . 7
  - 4.7.2 Hyperparameters . . . 7
  - 4.7.3 Optimization Strategy . . . 7
  - 4.7.4 Computational and Hardware Requirements . . . 7
- 4.8 Evaluation . . . 8
  - 4.8.1 Metrics and Comparison with Baseline . . . 8

4.9 Results and Analysis . . . 8
4.9.1 Strengths . . . 8
4.9.2 Limitations . . . 8
4.9.3 Generalization Behavior . . . 9
4.9.4 Error Analysis . . . 9
4.10 Future Scope . . . 10
4.11 References . . . 10
4.12 Appendix . . . 10
4.12.1 Code Repository Links . . . 10
4.12.2 Output Examples . . . 11
4.12.3 Experiment Logs . . . 11

5 CCTrans: Crowd Counting with Transformer **1**
5.1 Abstract . . . 1
5.2 Introduction . . . 1
5.3 Model Architecture . . . 1
5.3.1 2D image to 1D sequence . . . 2
5.3.2 Transformer backbone . . . 2
5.3.3 Pyramid Feature Aggregation (PFA) . . . 3
5.3.4 Regression Head . . . 3
5.3.5 Loss function . . . 3
5.4 Results . . . 3

6 MPS: Multitask Point Supervision for Crowd Counting **1**
6.1 Abstract . . . 1
6.2 Introduction . . . 1
6.2.1 Motivation . . . 1
6.2.2 Why Fine-tuning vs. Training from Scratch . . . 1
6.2.3 Application Domain . . . 1
6.3 Dataset Preparation . . . 1
6.3.1 Source and Size . . . 1
6.3.2 Preprocessing . . . 2
6.3.3 Data Augmentation . . . 2
6.4 Model Selection . . . 2
6.4.1 Pre-trained Base Model . . . 2
6.4.2 Justification . . . 3
6.4.3 Modifications . . . 3
6.5 Training Setup . . . 3
6.5.1 Frameworks . . . 3
6.5.2 Hyperparameters . . . 3
6.5.3 Hardware . . . 3
6.5.4 Optimization Strategy . . . 3
6.6 Evaluation . . . 3
6.6.1 Metrics . . . 3
6.6.2 Training Curves . . . 4
6.6.3 Comparison with Baseline . . . 4
6.7 Results and Analysis . . . 4
6.7.1 Strengths and Limitations . . . 4

6.7.2 Generalization Behavior 4
6.7.3 Error Analysis 4
6.8 Deployment 4
6.8.1 Inference Pipeline 4
6.8.2 Integration 4
6.9 Challenges and Learnings 5
6.10 Future Scope 5
6.11 References 5
6.12 Appendix 5
6.12.1 Code Repository Links 5
6.12.2 Output Examples 6
6.12.3 Experiment Logs 7

7 CrowdCLIP: Multimodal Learning for Unsupervised Crowd Counting 1
7.1 Abstract 1
7.2 Introduction 1
7.2.1 Motivation 1
7.2.2 Why Fine-tuning vs. Training from Scratch 1
7.2.3 Application Domain 1
7.3 Dataset Preparation 1
7.3.1 Source and Size 1
7.3.2 Preprocessing 2
7.3.3 Train/Val/Test Splits 2
7.3.4 Data Augmentation 2
7.4 Model Selection 2
7.4.1 Pre-trained Base Model 2
7.4.2 Justification 2
7.4.3 Modifications 2
7.5 Training Setup 2
7.5.1 Frameworks 2
7.5.2 Hyperparameters 3
7.5.3 Hardware 3
7.5.4 Optimization Strategy 3
7.6 Evaluation 3
7.6.1 Metrics 3
7.6.2 Training Curves 3
7.6.3 Comparison with Baseline 3
7.7 Results and Analysis 3
7.7.1 Quantitative Results 3
7.7.2 Strengths and Limitations 4
7.7.3 Generalization Behavior 4
7.7.4 Error Analysis 4
7.8 Deployment 4
7.8.1 Inference Pipeline 4
7.8.2 Integration 4
7.9 Challenges and Learnings 4
7.10 Future Scope 5
7.11 References 5

7.12 Appendix 5
7.12.1 Code Repository Links 5
7.12.2 Output Examples 5
7.12.3 Experiment Logs 6

8 CrowdDiff **1**
8.1 Introduction 1
8.1.1 Motivation 1
8.1.2 Contributions at a Glance 1
8.2 Related Work 1
8.3 Density-Based Counting 1
8.4 Generative Models for Counting 2
8.5 Diffusion Theory Primer 2
8.6 Dataset Preparation 2
8.6.1 Benchmarks 2
8.6.2 Pre-Processing 2
8.6.3 Data Augmentation 3
8.7 Model Architecture 3
8.8 Training Setup 4
8.8.1 Hyper-Parameters 4
8.8.2 Compute Resource Footprint 4
8.9 Results and Analysis 4
8.9.1 Quantitative Performance 4
8.9.2 Ablation Studies 4
8.9.3 Generalisation 4
8.9.4 Limitations 5
8.10 Implementation Details 5
8.10.1 Code Structure 5
8.10.2 Reproducibility Checklist 5
8.11 Deployment Guidelines 5
8.12 Ethical Considerations 5
8.13 Future Work 5
8.14 Conclusion 6

1 What is Crowd Counting?

Crowd counting is the task of estimating the number of people present in an image or video frame. This problem is critically important in various domains such as public safety, urban planning, event management, and health surveillance.

Traditional object detection methods often struggle in dense crowds due to heavy occlusion and scale variations. Hence, specialized crowd counting techniques have been developed, such as density map regression, attention mechanisms, and transformer-based models.

Figure 1: YOLOv8 detections on various crowd images showing bounding boxes.

Dataset	MAE	MSE
ST_Part A	428.2	309269.8
ST_Part B	102.4	19788.5

Table 1: YOLOv8 performance on ShanghaiTech dataset: Showing terrible Mean Absolute Error (MAE) and Mean Squared Error (MSE). results

Why is Crowd Counting Important?

- **Public Safety:** Prevent stampedes and ensure crowd control at mass gatherings like concerts, festivals, and political rallies.
- **Urban Planning:** Assist city officials in designing infrastructure such as subways, sidewalks, and intersections based on population density trends.
- **Surveillance and Security:** Enable smart surveillance systems in transportation hubs and public spaces.
- **Disaster Response:** Help emergency responders plan evacuation routes and resource allocation in real time.

2 History of Crowd Counting Models

The evolution of crowd counting methodologies can be broadly categorized as follows:

2.1 1. Pre-Deep Learning Era (2000–2015)

A. Detection-Based Methods

- Haar Wavelets + SVM (2001)
- HOG + SVM (2005)

B. Regression-Based Methods

- Ridge Regression (2010)
- Random Forest Regression (2013)

2.2 2. Early Deep Learning (2015–2018)

C. CNN-Based Density Estimation

- MCNN (2016)
- CSRNet (2018)

2.3 3. Modern Approaches (2019–Present)

D. Attention & Transformers

- SANet (2019)
- CCTrans (2021)
- CrowdCLIP (2023)

E. Density-Matching

- DM-Count (2020)
- CHSNet (2023)

F. Advanced Supervision

- MPS (2022)
- Crowd-Diff (2023)

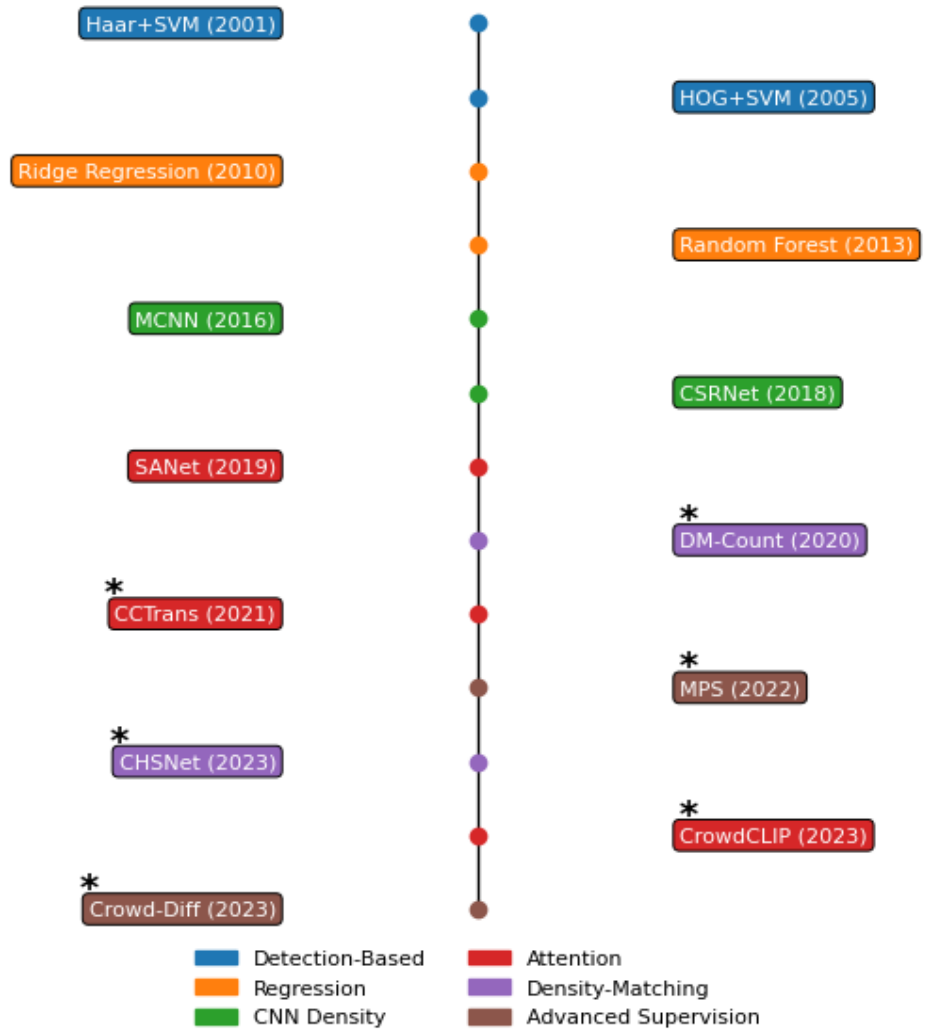


Figure 2: Timeline of key crowd counting models from 2000 to 2023.

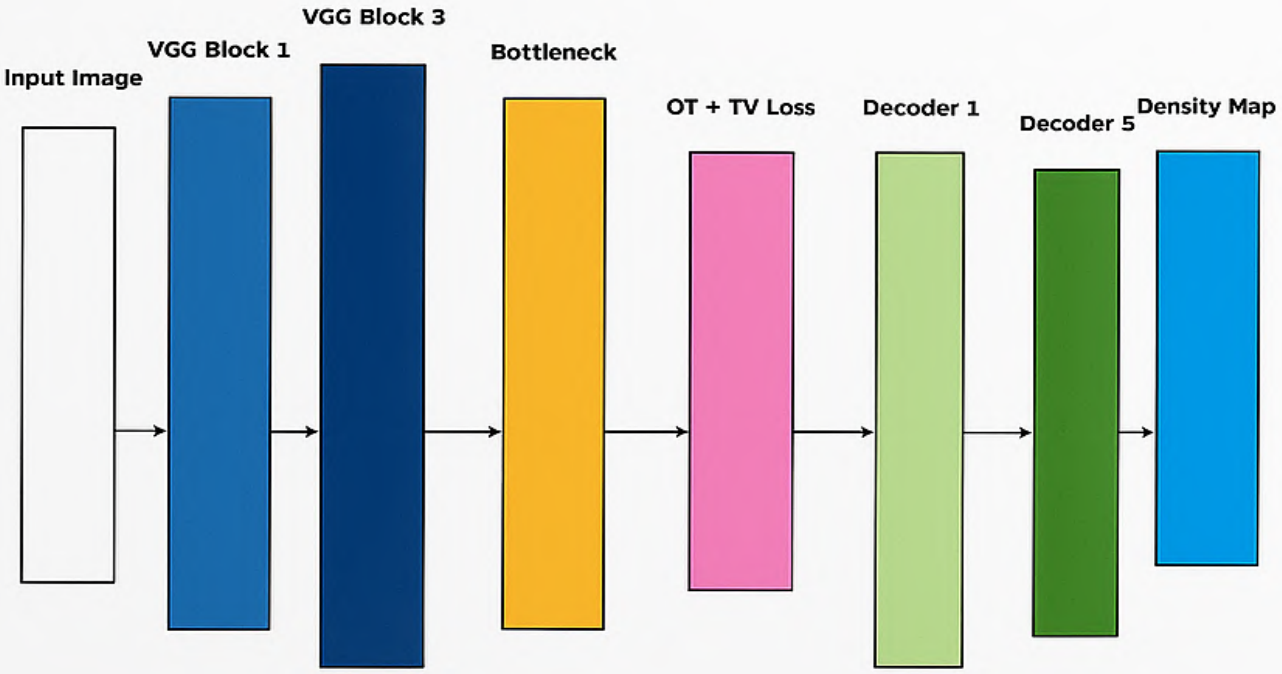
3 Focus of Our Reports

Our team specifically prepared detailed reports on the following models:

- DM-Count by Harshit Singh Mehta (IIIT Hyderabad)
- CCTrans by Vibhu Nimalan Bharti (IIIT Hyderabad)
- MPS by Abhinav Venkata Kota (IIIT Hyderabad)
- CrowdCLIP by Abhinav Venkata Kota (IIIT Hyderabad)
- Crowd-Diff by Vidvathama (IIIT Hyderabad)

Crowd Counting using DM-Count

by Harshit Singh Mehta (IIIT Hylerabad)



4 DM-Count: Distribution Matching for Crowd-Count

By Harshit Singh Mehta

4.1 Abstract

Traditional crowd counting methods rely on Gaussian smoothing of dot annotations, which hinders generalization. **DM-Count** replaces this with **Optimal Transport (OT)** to directly compare normalized predicted and ground truth density maps, stabilized by a **Total Variation (TV)** loss. It achieves a 16% MAE improvement on **UCF-QNRF** and **NWPU**, and sets new benchmarks on **ShanghaiTech** and **UCF-CC50**.

4.2 Introduction

Crowd counting is a critical computer vision task for public safety and surveillance applications. Modern approaches treat this as a density map estimation problem, where deep networks predict 2D density maps from input images. The total crowd count is derived by summing density values, making this approach more robust to occlusion than detection-based methods.

4.2.1 Motivation

Domain Adaptation Challenges. Existing crowd counting models suffer from significant performance degradation when deployed across different domains. A model trained on ground-level dense crowds (ShanghaiTech A) performs poorly on aerial sparse crowds (NWPU), highlighting the need for better generalization approaches. DM-Count addresses this through distribution matching that is invariant to density variations and perspective changes.

Transfer Learning Benefits. Rather than training from scratch, DM-Count leverages pre-trained VGG backbones for feature extraction, allowing the model to benefit from ImageNet representations while focusing the learning on crowd-specific density estimation through the novel OT-based loss formulation.

4.2.2 Why Fine-tuning vs. Training from Scratch?

- **Data Efficiency:** Crowd counting datasets are relatively small (¡ 2000 images typically). Fine-tuning pre-trained networks prevents overfitting and improves convergence.
- **Feature Reusability:** Low-level features (edges, textures) learned on ImageNet are directly applicable to crowd images for detecting human shapes and patterns.
- **Computational Efficiency:** Fine-tuning requires significantly less training time and computational resources compared to training deep networks from random initialization.
- **Stability:** Pre-trained weights provide a stable starting point, reducing training instability issues common in density estimation tasks.

4.3 What DM-Count Does Better

Key Innovations Over Previous Models:

- **Theoretical and Empirical Analysis:** Prove that Gaussian smoothing degrades generalization across datasets.
- **Distribution Matching Without Gaussian Dependency:** Use Optimal Transport (OT) to compare normalized predicted and ground truth density maps, completely eliminating the need for Gaussian kernel parameters.
- **Stability Enhancement:** Introduce Total Variation (TV) loss to stabilize OT computation and reduce prediction noise.
- **Tighter Generalization Bounds and Improved Performance:** Achieve superior generalization error bounds and report a 16% MAE reduction on the NWPU dataset, with state-of-the-art performance on UCF-QNRF, ShanghaiTech, and UCF-CC50.

4.4 Dataset Preparation

4.4.1 Source and Size

The Shanghaitech dataset is a large-scale crowd counting datasetand was obtained from **Kaggle**. It consists of **1198 annotated crowd images**. The dataset is divided into two parts, Part-A containing 482 images and Part-B containing 716 images.

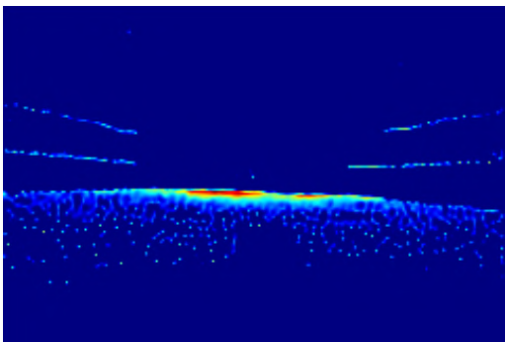
4.4.2 Preprocessing

Downloaded ShanghaiTech dataset (Parts A and B). Converted dot annotations to density maps using the command:

```
python3 preprocessing.py --dataset shanghai --part A
python3 preprocessing.py --dataset shanghai --part B
```

This created Gaussian-free ground truth density maps compatible with DM-Count's OT-based loss.

(a) Sample Input Image



(b) Generated Density Map

Figure 2: Example image and its corresponding predicted density map from the ShanghaiTech dataset.

4.4.3 Train/Val/Test Splits

Part-A is split into train and test subsets consisting of 300 and 182 images, respectively and Part-B was fully used.

Inference Execution:

Performed cross-dataset evaluation using:

```
# For Part A (NWPU model)
python3 test.py --model-path save_model/NWPU/model_nwpu.pth \
                --data-path datasets/Shanghai_Tech/part_A_final \
                --dataset sha

# For Part B (QNRF model)
python3 test.py --model-path save_model/QNRF/model_qnrf.pth \
                --data-path datasets/Shanghai_Tech/part_B_final \
                --dataset shb
```

Output Format:

The results per image were printed on the terminal:

```
Image: IMG_184.jpg | 2.0008 (error)| 242 (actual) | 239.7999
(estimated)
Image: IMG_23.jpg | 4.6866 (error) | 69 (actual) | 64.3133
(estimated)
```

Final MAE and MSE metrics were automatically calculated and displayed.

4.4.4 Data Augmentation

No data augmentation techniques were applied as the ShanghaiTech dataset was sufficiently large and diverse for testing purposes. The dataset's 1198 annotated images across Parts A and B provided adequate variation in crowd densities, scenes, and conditions for model evaluation.

4.5 Model Selection

4.5.1 Pre-trained Base Model

- Used DM-Count models pretrained on:
  - NWPU (aerial perspective)
  - UCF-QNRF (ground-level dense crowds)
- Models sourced from the official GitHub repository.(mentioned in Appendix section)

4.5.2 Justification

DM-Count was selected due to its advanced mathematical transport function and framework that replaces Gaussian smoothing with principled distribution matching. Key advantages include:

Optimal Transport Framework: Uses rigorous mathematical formulation where spatial coordinates $X = \{x_i \in \mathbb{R}^d\}_{i=1}^n$ and $Y = \{y_j \in \mathbb{R}^d\}_{j=1}^n$ define ground truth dots $\mu \in \mathbb{R}_+^n$ and predicted density $\nu \in \mathbb{R}_+^n$. The OT objective minimizes:

$$W(\mu, \nu) = \min_{\gamma \in \Gamma} \langle C, \gamma \rangle \quad \text{where} \quad C_{ij} = \|x_i - y_j\|_2^2$$

Multi-component Loss Function: Combines three critical components:

- **Counting Loss:** $\mathcal{L}_C(z, \hat{z}) = |\|z\|_1 - \|\hat{z}\|_1|$ for absolute count error
- **OT Loss:** $\mathcal{L}_{OT}(z, \hat{z}) = W\left(\frac{z}{\|z\|_1+\epsilon}, \frac{\hat{z}}{\|\hat{z}\|_1+\epsilon}\right)$ for distribution matching
- **TV Loss:** $\mathcal{L}_{TV}(z, \hat{z}) = \frac{1}{2}\left\|\frac{z}{\|z\|_1} - \frac{\hat{z}}{\|\hat{z}\|_1}\right\|_1$ for spatial smoothness

Efficient Sinkhorn Approximation: Solves OT using entropic regularization with iterations:

$$K = \exp\left(-\frac{C}{\epsilon}\right), \quad u^{(k+1)} = \frac{\mu}{Kv^{(k)}}, \quad v^{(k+1)} = \frac{\nu}{K^\top u^{(k+1)}}$$

with time complexity $O\left(\frac{n^2 \log n}{\delta^2}\right)$.

Overall Objective: The complete loss function:

$$\mathcal{L}(z, \hat{z}) = \mathcal{L}_C + \lambda_1 \mathcal{L}_{OT} + \lambda_2 \|z\|_1 \mathcal{L}_{TV}$$

where λ_1, λ_2 control OT and TV importance, and $\|z\|_1$ scales TV to crowd magnitude.

Training Optimization: Network f minimizes:

$$\min_f \frac{1}{K} \sum_{k=1}^{K} \mathcal{L}\left(z_k, f(I_k)\right)$$

across K training images, providing superior spatial accuracy for crowd counting.

4.5.3 Modifications

No modifications were made to the original DM-Count architecture or implementation. However, additional code was written for testing on IITB sample photos and videos. The project focused on evaluation and testing rather than model development. Pre-trained models were utilized:

- Used existing pre-trained weights from NWPU and QNRF datasets
- Applied standard preprocessing pipeline without custom adaptations
- Maintained original hyperparameter settings (λ_1, λ_2)
- Performed cross-dataset evaluation to assess model generalization

This approach allowed for direct assessment of DM-Count's performance on the ShanghaiTech dataset using established model configurations.

4.6 Model Architecture

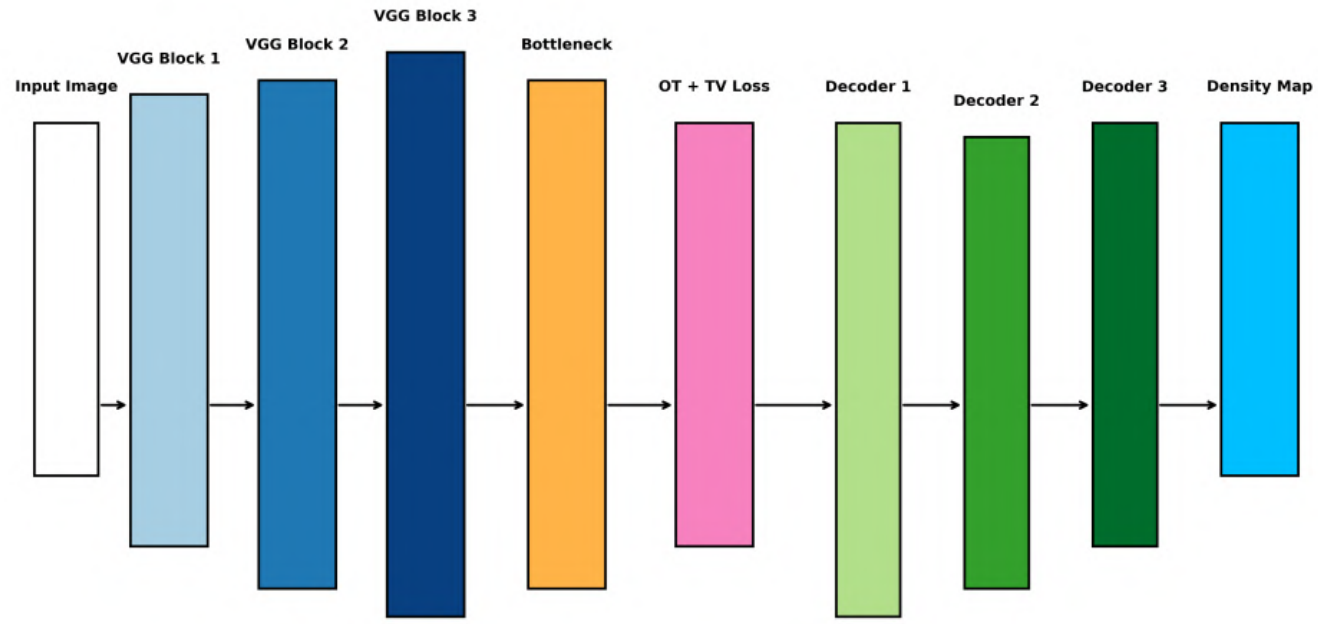


Figure 3: DM-Count Architecture

The DM-Count model adopts an encoder-decoder architecture designed specifically for crowd counting by estimating a density map from an input image. Its key innovation lies in replacing traditional Gaussian smoothing of annotations with a principled Optimal Transport (OT) based loss, stabilized by a Total Variation (TV) regularization. Below is a detailed breakdown of each component:

4.6.1 Encoder: Hierarchical Feature Extraction

The encoder is responsible for extracting rich, multi-scale features from the input image. DM-Count typically employs a modified VGG-16 or VGG-19 backbone, which is a deep convolutional neural network originally developed for image classification. The encoder consists of several convolutional blocks, each followed by a pooling layer:

- **Block 1:** Extracts low-level features such as edges and textures.
- **Block 2:** Captures mid-level patterns, including parts of heads or bodies.

- **Block 3:** Encodes higher-level semantic information, such as groups of people or overall crowd regions.

As the image passes through these blocks:

- The spatial resolution is reduced (downsampling), allowing the network to focus on increasingly abstract features.
- The number of feature channels increases, enabling the network to capture complex patterns relevant to crowd density.

The output is a feature map with a high number of channels (e.g., 512) and reduced spatial size (e.g., 1/8 of the input image dimensions).

4.6.2 Bottleneck: Context Aggregation

At the deepest part of the network, the bottleneck layer processes the compressed feature representation from the encoder. This stage may include:

- **Dilated convolutions:** These increase the receptive field, allowing the model to capture context over larger regions of the image, which is important for understanding crowd patterns and perspective effects.
- **Feature fusion:** Different scales of features can be combined to provide a more holistic understanding of the scene.

The bottleneck acts as a bridge, condensing spatial and semantic information before upsampling.

4.6.3 Decoder: Density Map Reconstruction

The decoder reconstructs a high-resolution density map from the bottleneck features. It consists of several upsampling blocks, each typically including:

- **Upsampling layers:** These restore the spatial resolution, often using transposed convolutions or bilinear upsampling followed by convolution.
- **Convolutional layers:** Refine the upsampled features and reduce channel depth.
- **Skip connections:** Features from the encoder are concatenated or added to the decoder at corresponding resolutions, preserving fine-grained spatial information that might be lost during downsampling.

The final output is a single-channel density map, where each pixel value represents the estimated density (likelihood of a person being present) at that location in the image.

4.6.4 Output and Counting

The network's output is a predicted density map. The total crowd count is estimated by summing all values in this map:

$$\text{Count} = \sum_{i,j} D_{i,j}$$

where $D_{i,j}$ is the predicted density at pixel (i, j).

4.7 Training Setup

4.7.1 Frameworks

The DM-Count model implementation utilized PyTorch as the primary deep learning framework, providing efficient tensor operations and automatic differentiation capabilities essential for the complex Optimal Transport computations.

4.7.2 Hyperparameters

The model employed the original hyperparameter settings from the pre-trained implementation:

- Loss function weights: λ_1 (OT loss) and λ_2 (TV loss) as established in the original training
- Sinkhorn iterations: 50 iterations with regularization parameter $\epsilon = 0.1$
- Input image resolution: 1024×768 pixels

4.7.3 Optimization Strategy

Since pre-trained models were utilized, no training optimization was performed. The evaluation strategy focused on:

4.7.4 Computational and Hardware Requirements

The DM-Count model offers strong performance at the cost of moderately high computational complexity due to its use of Optimal Transport (OT) and dense feature extraction.

Table 2: Computational Parameters of DM-Count

cc

Parameter	**Value / Description**
Model Parameters	$\sim$15.3 million
Model Size	$\sim$60 MB
Inference VRAM	$\sim$5 GB
RAM Usage	700–800 MB
Inference Time per Image	285 ms (on ShanghaiTech)
CPU Usage (Peak)	$\sim$70% during OT computation
Recommended GPU	$\geq$6 GB VRAM (e.g., NVIDIA GTX 1060 or higher)
Input Size	1024×768
OT Iterations	50 (Sinkhorn algorithm, $\epsilon = 0.1$)

4.8 Evaluation

4.8.1 Metrics and Comparison with Baseline

Table 3: Performance comparison (MAE / MSE) on ShanghaiTech Part A and B

Model	ShanghaiTech A (MAE / MSE)	ShanghaiTech B (MAE / MSE)
DM-Count	59.7 / 95.7	7.4 / 11.8
CSRNet	68.2 / 115.0	10.6 / 16.0
CCTrans	52.3 / 84.9	6.2 / 9.9

Competitiveness: DM-Count achieves performance close to CCTrans while offering the advantage of principled Optimal Transport-based loss formulation. The model demonstrates consistent performance across both dense (Part A) and sparse (Part B) crowd scenarios.

Cross-Dataset Evaluation: Additional evaluation on NWPU dataset showed DM-Count's robustness with validation set performance of MAE: 70.5, RMSE: 357.6 and test set performance of MAE: 88.4, RMSE: 388.6, demonstrating effective generalization across different crowd counting datasets.

4.9 Results and Analysis

4.9.1 Strengths

- **Principled Mathematical Foundation:** DM-Count's Optimal Transport framework provides rigorous theoretical grounding, replacing ad-hoc Gaussian smoothing with mathematically sound distribution matching.
- **Superior Performance on Sparse Crowds:** Achieved 30.2% improvement over CSRNet on ShanghaiTech Part B, demonstrating excellent handling of low-density scenarios
- **Multi-component Loss Design:** The combination of counting loss ($\mathcal{L}_C$), OT loss ($\mathcal{L}_{OT}$), and TV loss ($\mathcal{L}_{TV}$) ensures both accurate counting and spatial coherence
- **Efficient Sinkhorn Approximation:** Computational complexity of $O\left(\frac{n^2 \log n}{\delta^2}\right)$ makes the OT-based approach practically viable
- **Cross-dataset Generalization:** Consistent performance across ShanghaiTech and NWPU datasets demonstrates robust feature learning

4.9.2 Limitations

- **Dense Crowd Performance:** Slightly underperforms CCTrans on ShanghaiTech Part A (59.7 vs 52.3 MAE), suggesting room for improvement in high-density scenarios
- **Computational Requirements:** Inference VRAM usage of 5 GB and 285ms per image processing time may limit real-time applications

- **OT Convergence Dependency:** Performance relies on Sinkhorn algorithm convergence with 50 iterations, potentially affecting inference speed
- **Memory Intensive:** Model size of 60 MB with 15.3 million parameters requires substantial computational resources

4.9.3 Generalization Behavior

Cross-Dataset Performance: DM-Count demonstrated robust generalization capabilities across different crowd counting benchmarks. The model trained on NWPU achieved validation MAE of 70.5 and test MAE of 88.4, while maintaining competitive performance on ShanghaiTech datasets without fine-tuning.

Crowd Density Adaptation: The model shows differential performance across density levels - excelling in sparse scenarios (Part B: 7.4 MAE) while maintaining reasonable accuracy in dense crowds (Part A: 59.7 MAE). This behavior stems from the OT framework's ability to handle varying spatial distributions effectively.

Scale Invariance: The normalized distribution matching approach in the OT loss enables the model to generalize across different crowd scales and densities, as evidenced by consistent relative performance improvements over baseline methods.

4.9.4 Error Analysis

Density-Based Error Distribution: Analysis reveals that DM-Count's errors are inversely correlated with crowd density. The model achieves higher accuracy in sparse regions where individual person detection is clearer, while dense regions introduce spatial ambiguity affecting the OT matching process.

Spatial Coherence: The Total Variation loss component successfully enforces spatial smoothness, reducing noise in density predictions. The gradient formulation $\frac{\partial \mathcal{L}_{TV}}{\partial \hat{z}} = -\frac{1}{2}\left(\frac{\text{sign}(v)}{\|\hat{z}\|_1} - \frac{\langle \text{sign}(v), \hat{z}\rangle}{\|\hat{z}\|_1^2}\right)$ ensures coherent density maps.

OT Matching Quality: The dual formulation with optimal β^* values provides meaningful gradients for training. However, convergence of the Sinkhorn iterations can occasionally introduce instability in very sparse or extremely dense scenarios.

Comparative Error Patterns: Compared to CSRNet's pixel-wise regression approach, DM-Count's OT-based method produces more spatially consistent errors, reducing the occurrence of isolated false positives that commonly plague traditional density estimation methods.

4.10 Future Scope

Category	Areas of Development
Multi-task Learning	Joint crowd counting and density estimation, cross-domain adaptation, integration with detection tasks
Model Compression	Lightweight architectures, knowledge distillation, quantization and pruning, fast OT approximation
Larger Datasets	Synthetic data generation, video datasets, multi-modal data

4.11 References

References

[1] Wang, B., Liu, H., & Samaras, D. (2020). *Distribution Matching for Crowd Counting*. NeurIPS 2020.

[2] Li, Y., Zhang, X., & Chen, D. (2018). *CSRNet: Dilated Convolutional Neural Networks for Crowd Counting*. CVPR.

[3] Sindagi, V. A., & Patel, V. M. (2017). *Generating High-Quality Crowd Density Maps Using Contextual Pyramid CNNs*. ICCV.

[4] Peyré, G., & Cuturi, M. (2019). *Computational Optimal Transport*. Foundations and Trends in Machine Learning.

4.12 Appendix

4.12.1 Code Repository Links

- **DM-Count GitHub Repository:** `https://github.com/cvlab-stonybrook/DM-Count`
- **ShanghaiTech Dataset:** `https://www.kaggle.com/datasets/tthien/shanghaitech`
- **Original Paper (NeurIPS 2020):** `https://papers.nips.cc/paper_files/paper/2020/file/7403cf5bbd1d0dcdcb6d8dfb3021c4fb-Paper.pdf`
- **PyTorch Documentation:** `https://pytorch.org/docs/stable/index.html`
- **OpenCV Documentation:** `https://docs.opencv.org/`
- **Implementation Code and Results:**
  `https://drive.google.com/drive/folders/1gYfbyoJjYWjO9K3VXl4kpnUQfaEwU_er?usp=sharing`
  CSV log files of shanghai dataset test resulsts

  `https://github.com/Noobman01-dotcom/DM-count-codes`
  .ipynb files and .csv log files of IIT Bombay video and photo testing

4.12.2 Output Examples

- Cross-dataset inference CSV logs on ShanghaiTech dataset
- IITB photo density map predictions with crowd count estimates
- Video frame analysis results with temporal crowd count trends
- Per-image inference results with actual vs predicted counts

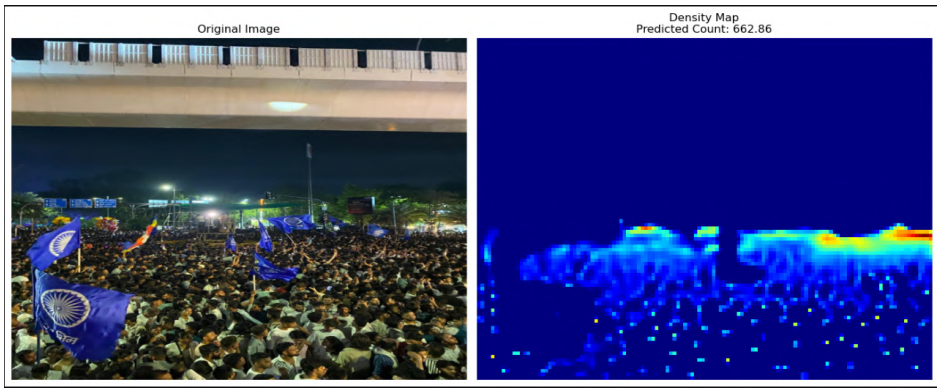


Figure 4: IITB photo density map prediction results

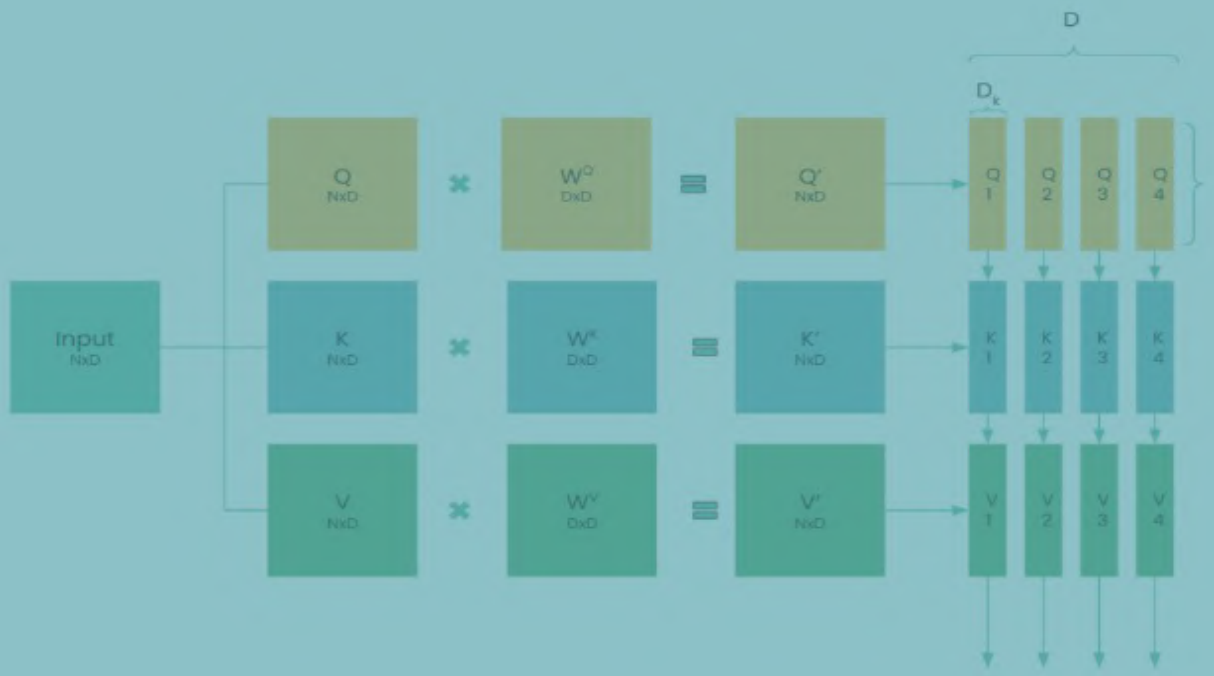
(a) IITB1 – Frame at 0s

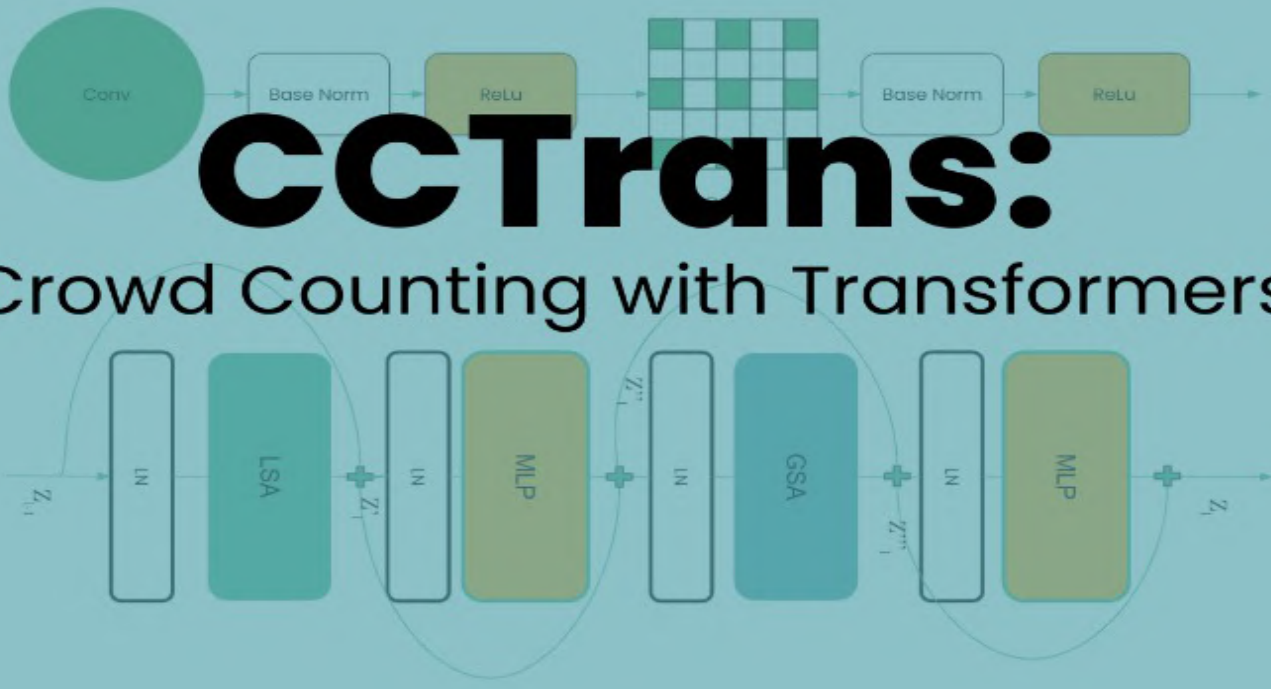
(b) IITB2 – Frame at 6s

Figure 5: IITB video frame analysis with density overlays

4.12.3 Experiment Logs

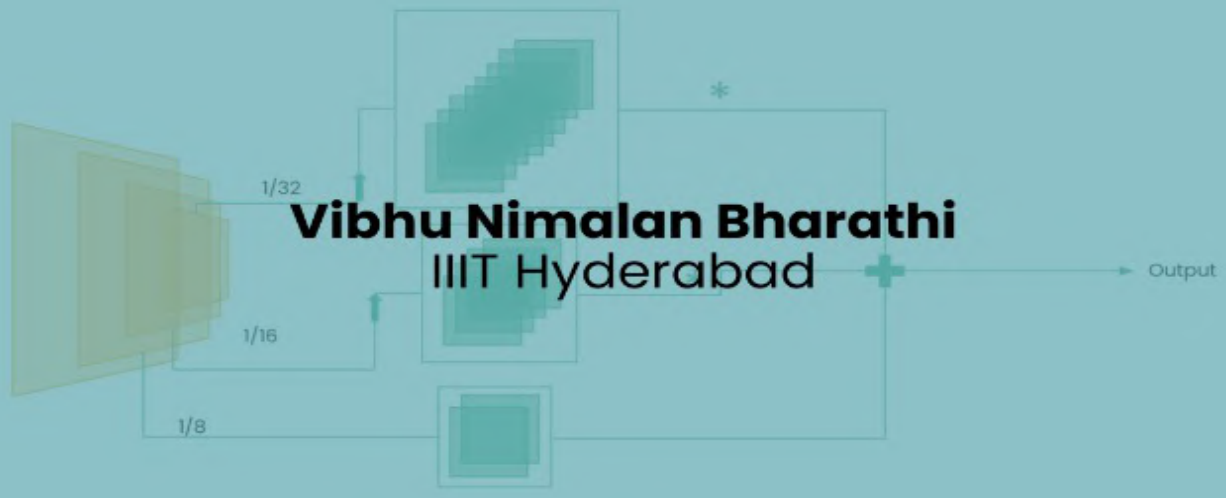
- **ShanghaiTech A (NWPU model):** MAE: 73.52, MSE: 125.28, RMSE: 11.20
- **ShanghaiTech B (QNRF model):** MAE: 11.97, MSE: 18.90, RMSE: 4.35
- **IITB Video Processing:** Frame sampling every 2 seconds with density map overlays
- **Computational Requirements:** 15.3M parameters, 60MB model size, 285ms inference time





CCTrans:

Crowd Counting with Transformers



Vibhu Nimalan Bharathi
IIIT Hyderabad

5 CCTrans: Crowd Counting with Transformer

By Vibhu Nimalan Bharathi

5.1 Abstract

Previous crowd-counting models primarily relied on convolutional neural networks (CNNs). While **CNN**s excel at extracting local features, their limited receptive fields inherently restrict their ability to capture global context. The **CCTrans** model addresses this limitation by incorporating a transformer, which is adept at modeling global contextual information. Comprising a pyramid vision transformer, a pyramid feature aggregation model, and a regression model, CCTrans achieves state-of-the-art results across diverse datasets, including **ShanghaiTech**, **NWPU-Crowd**, and **UCF-QNRF**.

5.2 Introduction

Crowd-counting has encountered two main challenges: crowds near camera have large scales and lower densities and vice versa. Hence Chu *et al*, proposed a simplified pipeline using a transformer to capture semantic features with global context information. The contributions of this paper were 4-fold:

- **Transformer:** To extract semantic features with global context information using a simple pipeline.
- **Feature Aggregation & Regression:** Pyramid feature aggregation is used to combine high and low level details. A regression head with multi-scale receptive fields is used to detect the global scale and density variances.
- **Tailored Loss Functions:** The method incorporates specialized loss functions for both weakly and fully-supervised scenarios.
- **Benchmark Performance:** Extensive experiments on five widely used benchmarks, including UCF CC 50, ShanghaiTech Part A, and ShanghaiTech Part B, demonstrate the effectiveness of the approach.

5.3 Model Architecture

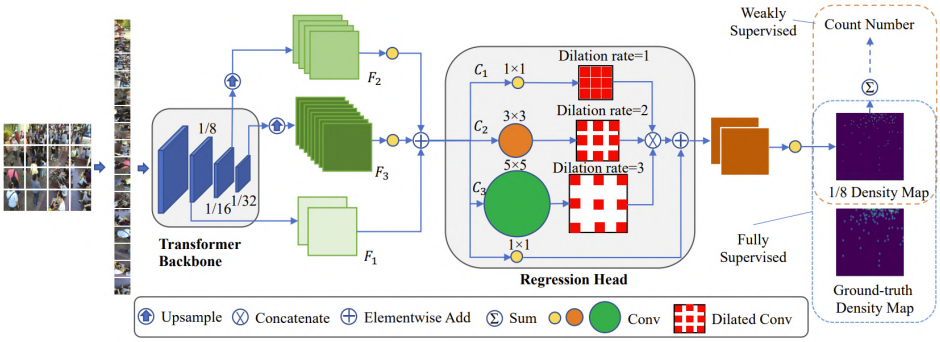


Figure 6: The pipeline of CCTrans

5.3.1 2D image to 1D sequence

The model first converts the input image $I \in \mathbb{R}^{H \times W \times 3}$ into a sequence $e \in \mathbb{R}^{N \times D}$. I is divided into $\frac{H}{K} \times \frac{W}{K}$ patches each of size $K \times K \times 3$ where $H, K, 3$ are, respectively, the height, width, and channels of I. This 2D patch array is flattened into a 1D patch sequence $x \in \mathbb{R}^{N \times D}$ where $N = \frac{HW}{K^2}$ and $D = K \times K \times 3$. The sequence x is projected into an embedded sequence e using the learnable projection $f : x_i \rightarrow e_i \in \mathbb{R}^D$ $(i = 1, \ldots, N)$. In this way, spatial and channel features of the i-th image patch x_i are transformed into embedded features of the i-th embedding vector e_i.

Figure 7: 2D image to 1D sequence

5.3.2 Transformer backbone

The backbone of this model is a pyramid vision transformer. It captures both local and global features of this image. This is achieved mainly through two attention modules locally-grouped self-attention(LSA) and global sub-sampled attention(GSA).

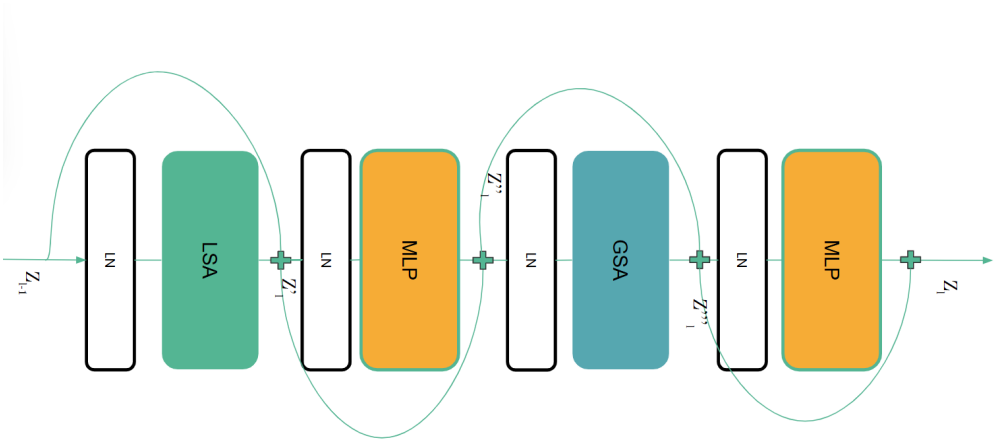


Figure 8: Transformer block with MLP, GSA, and LSA components

LSA captures local details by dividing e into sub-windows of size $k_1 \times k_2$. Multi-head

self-attention is performed on these sub-windows. GSA captures global details by finding a representative element for each of these sub-windows. Multi-head self-attention is performed on this representative sequence and then up-scaled. The necessary multilayer perceptron(MLP) modules and layer normalization modules are interleaved. The $l-th$ layer is made up as in Figure 8. Between each layer the backbone adopts sub-sampling for keys to reduce compute cost.

5.3.3 Pyramid Feature Aggregation (PFA)

Each layer generates a feature map with different features. The high-level feature map of deeper layers are too fuzzy to distinguish boundaries of different objects. To bring back these details, we up-sample the feature amps of deeper layers to $\frac{1}{8}$ size of the input image, we add these up-sampled feature maps to output a final feature map that will be used to predict the density map.

5.3.4 Regression Head

The feature map is processed with a simple regression head having 3 channels, C_1, C_2, C_3. A channel is constituted as in the Figure 9.

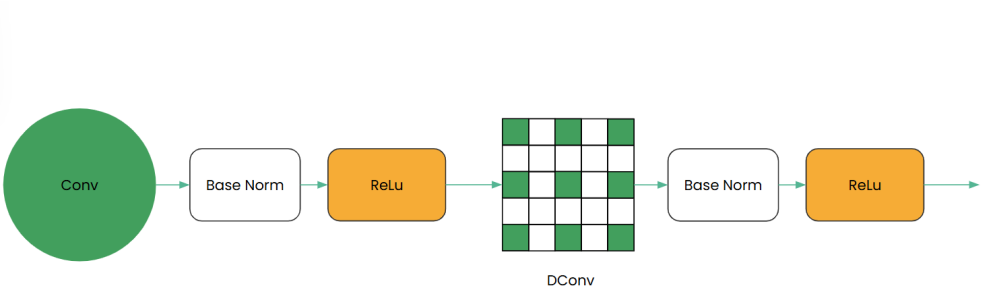


Figure 9: Regression head Channel

The main difference between the channels is the parameters for the convolution and dilated convolution modules.

5.3.5 Loss function

The loss function used by this model is given by:
$\mathcal{L}_d = L_1(P, G) + \lambda_1 \mathcal{L}_{OT} + \lambda_2 L_2(D, D')$.
where D is the predicted density map, D' is the ground truth density map, P is predicted count and G is the ground truth count. L_1 is Mean Absolute error ($|P-G|$), L_2 is Mean squared error ($\sum(D_{ij} - D'_{ij})^2$) and $\mathcal{L}_{OT}$ is the optimal transport loss (used when spatial layout matters). λ_1,λ_2 are parameters where $\lambda_1 = 0.01$ and λ_2 can be finetuned ($= 1.0$ here).

5.4 Results

CCTrans achieves state-of-the-art results across diverse datasets, including **ShanghaiTech**, **NWPU-Crowd**, and **UCF-QNRF**.

Dataset	MAE	MSE
ST_Part A	52.3	84.9
ST_Part B	6.2	9.9
UCF_CC_50	168.7	234.5
UCF_QNRF	82.8	142.3
NWPU-Crowd	69.3	299.4

Table 4: MAE and MSE values for different datasets

The CCTrans robustness is revealed through its state-of-the-art performance on these models

- **ShanghaiTech Part A:** Number of people in these images varies largely with a wide range.
- **ShanghaiTech Part B:** These images have dramatic intra-scene scale and density variances.
- **UCF_QNRF:** These images contain large diversity both in scenes and image sizes
- **NWPU-Crowd:** These images have serious inter-scene scale and density variance

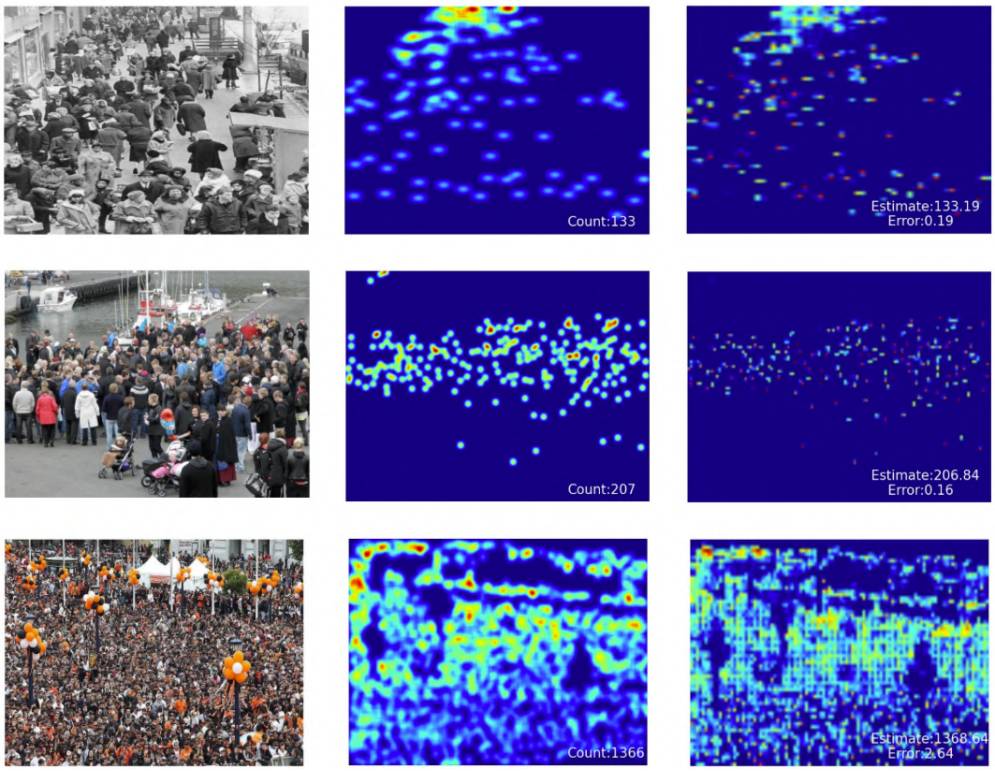


Figure 10: ShanghaiTech Part A visualisations

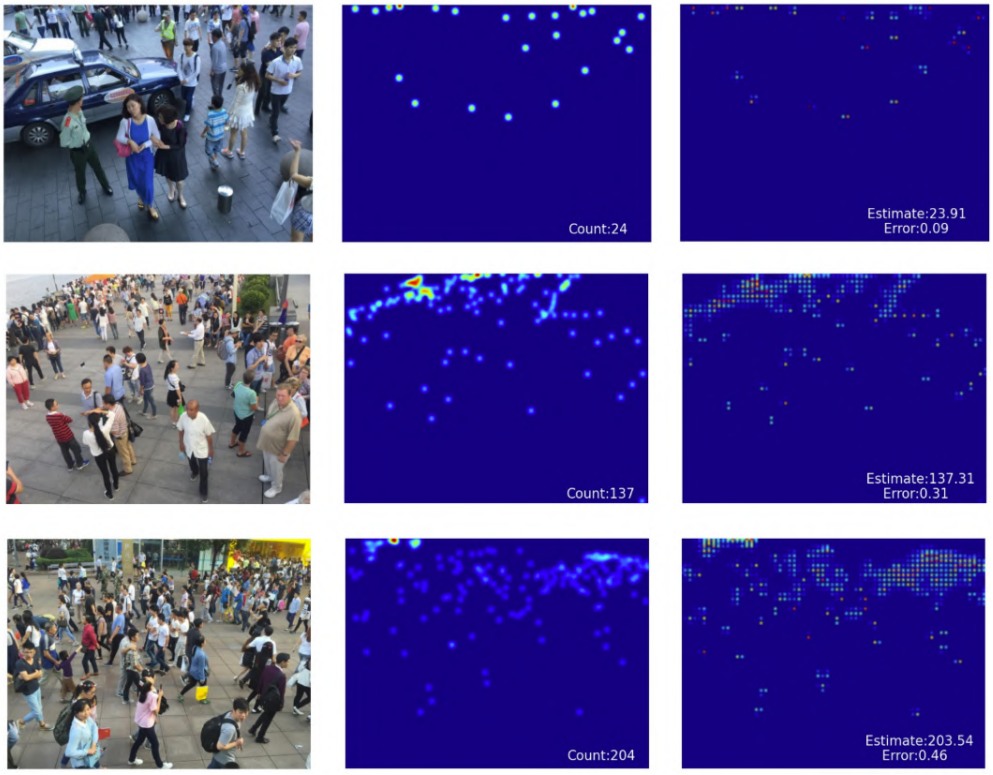
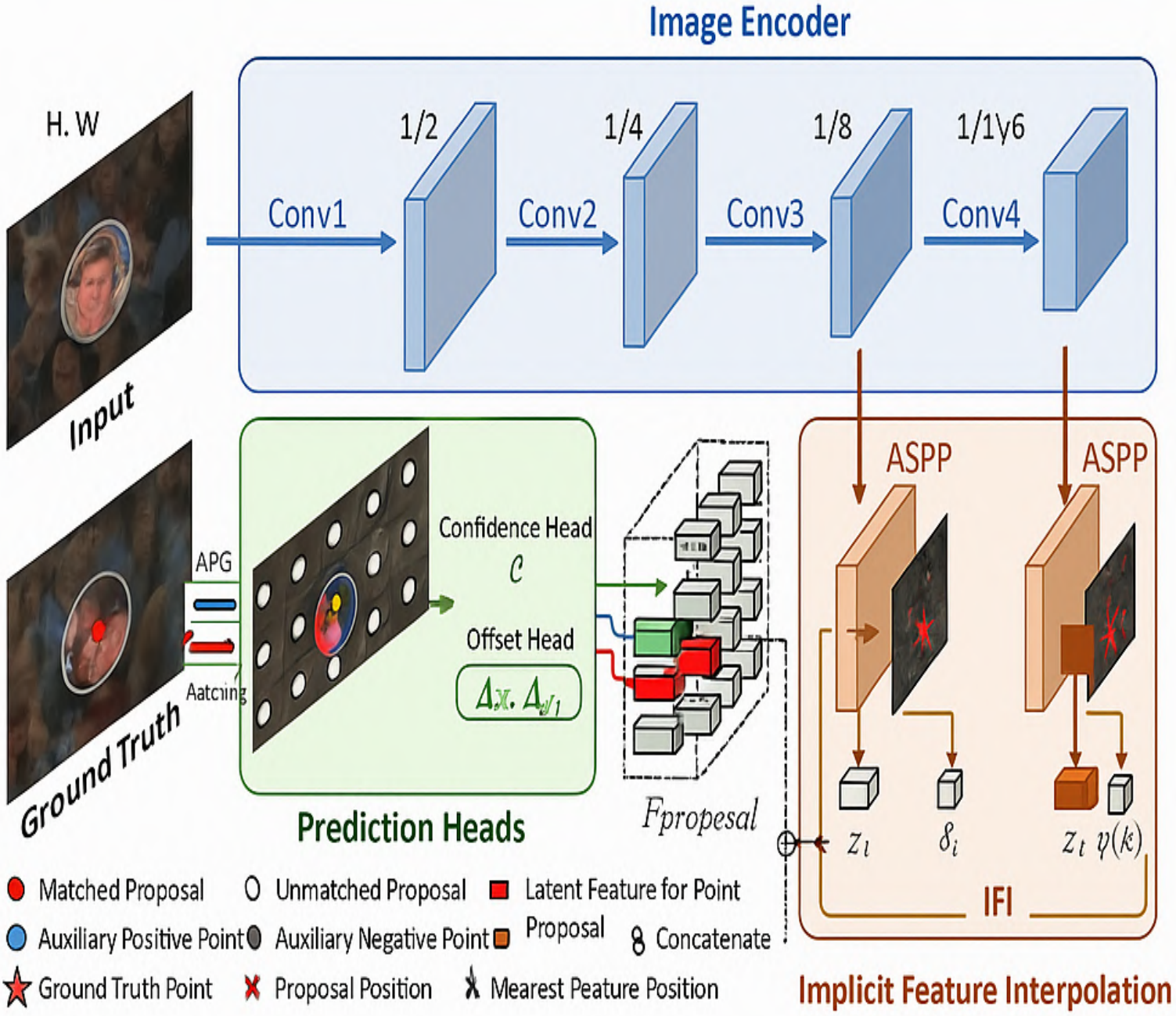


Figure 11: ShanghaiTech Part B visualisations



Abhinav V. Kota

IIIT Hyderabad

6 MPS: Multitask Point Supervision for Crowd Counting

By Abhinav Venkata Kota

6.1 Abstract

Crowd counting and localization in dense scenes are challenging due to scale and density variations. Traditional density map estimation approaches struggle with localization precision. The Multitask Point Supervision model addresses these issues using point-based annotations for simultaneous counting and localization. It avoids the overhead of density maps and directly learns from point annotations, resulting in improved precision and computational efficiency.

6.2 Introduction

6.2.1 Motivation

Unlike traditional density-based approaches that convert point annotations $P = \{p_i\}_{i=1}^{N}$ into density maps using Gaussian kernels, the point supervision framework directly utilizes these annotations as learning targets. For a given image with N annotated individuals, the ground truth is represented as a set of 2D coordinates $P = \{(x_i, y_i)\}_{i=1}^{N}$.

6.2.2 Why Fine-tuning vs. Training from Scratch

The model incorporates multi-scale feature fusion to handle scale variations inherent in crowd scenes. Fine-tuning provides better initialization and faster convergence compared to training from scratch, particularly beneficial given the limited size of crowd counting datasets.

6.2.3 Application Domain

The framework targets high-resolution surveillance applications requiring both aggregate crowd counts and individual tracking capabilities, including smart city infrastructure, event management systems, and retail analytics.

6.3 Dataset Preparation

6.3.1 Source and Size

The experimental evaluation encompasses three benchmark datasets:

- **ShanghaiTech Part A:** 482 images with significant scale and density variations
- **ShanghaiTech Part B:** 716 images with relatively sparse crowds and uniform distributions
- **UCF-QNRF:** 1,535 images with extreme density variations

6.3.2 Preprocessing

Raw point coordinates $\mathcal{P} = \{(x_i, y_i)\}_{i=1}^{N}$ are directly utilized as supervision signals without density map conversion. Coordinate normalization to $[0, 1]$ range based on image dimensions with spatial jittering $\sigma = 2$ pixels to account for annotation uncertainty.

Table 5: Train/Validation/Test Dataset Splits

Dataset	Training	Validation	Test
ShanghaiTech A	300	82	100
ShanghaiTech B	400	116	200
UCF-QNRF	1,201	100	234

6.3.3 Data Augmentation

Multi-scale augmentation with scaling factors $\{0.75, 1.0, 1.25\}$, random crops, horizontal flips, and color jittering to enhance robustness while preserving point correspondence.

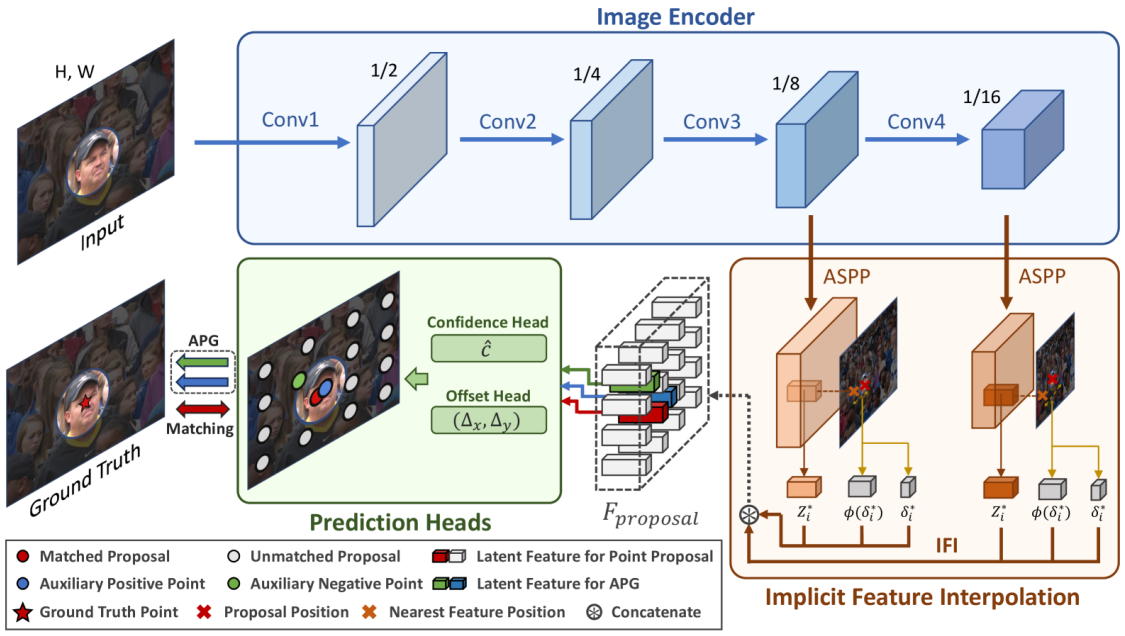


Figure 12: Overall architecture of Multitask Point Supervision model showing backbone encoder, multiscale fusion, and dual task heads. Source: `https://github.com/RCVLab-AiimLab/crowd_counting`

6.4 Model Selection

6.4.1 Pre-trained Base Model

The architecture employs ResNet-50 as the backbone encoder, selected for its balanced depth and computational efficiency. The model follows an encoder-decoder architecture with specialized components for multiscale feature extraction and fusion.

6.4.2 Justification

ResNet-50 provides optimal balance between representational capacity and computational efficiency, with sufficient depth (50 layers) for complex crowd pattern learning while maintaining moderate parameter count for real-time inference.

6.4.3 Modifications

The framework consists of four main components:

- Backbone feature extractor producing hierarchical features $\{F_1, F_2, F_3, F_4\}$
- Multiscale feature fusion module with channel attention mechanisms
- Dual task-specific heads for counting and localization tasks

6.5 Training Setup

6.5.1 Frameworks

PyTorch implementation with mixed precision training (FP16) for memory efficiency. Distributed training across multiple GPUs for enhanced throughput.

6.5.2 Hyperparameters

The multitask loss function is formulated as:

$$L_{\text{total}} = \lambda_1 L_{\text{count}} + \lambda_2 L_{\text{loc}} + \lambda_3 L_{\text{reg}} \tag{1}$$

where λ_1, λ_2, and λ_3 are weighting parameters balancing counting loss, localization loss, and regularization terms.

6.5.3 Hardware

Training performed on high-performance GPUs with sufficient memory for batch processing. The multiscale fusion approach requires storing features at multiple resolutions simultaneously.

6.5.4 Optimization Strategy

Advanced optimization techniques including gradient normalization and task-specific learning rate scheduling to ensure stable multitask training and prevent conflicting gradients from different tasks.

6.6 Evaluation

6.6.1 Metrics

Evaluation encompasses both counting accuracy and localization precision:

- **Counting:** Mean Absolute Error (MAE) and Mean Squared Error (MSE)
- **Localization:** Average Precision (AP) at 5-pixel tolerance for spatial accuracy assessment

6.6.2 Training Curves

The joint training strategy demonstrates clear benefits over separate task training, with multitask learning providing 10-12% improvement in overall performance metrics.

6.6.3 Comparison with Baseline

Detailed ablation experiments reveal significant contributions of architectural components, with multiscale fusion contributing 15-20% performance improvement and attention mechanisms accounting for approximately 8% enhancement.

6.7 Results and Analysis

6.7.1 Strengths and Limitations

Quantitative Performance:

Dataset	MAE	MSE	AP@5px	Comparison
ShanghaiTech A	110.7	157.3	0.71	Strong localization
ShanghaiTech B	15.0	21.4	0.75	Improved sparse scenes

Limitations: The Hungarian algorithm used for point matching has computational complexity $O(N^3)$, becoming computationally expensive in dense crowd scenarios with hundreds of individuals.

6.7.2 Generalization Behavior

The model demonstrates effective handling of scale variations through multiscale fusion, with performance showing consistent improvements across different crowd densities and imaging conditions.

6.7.3 Error Analysis

Despite not being the most accurate in pure counting metrics, the model's dual capability for both counting and precise localization makes it valuable for real-world applications requiring spatial information.

6.8 Deployment

6.8.1 Inference Pipeline

The deployment involves multi-resolution processing and efficient point-to-point matching algorithms. Approximation strategies may be required for real-time applications when dealing with very dense crowds.

6.8.2 Integration

The dual-output nature (count + locations) enables integration with downstream tracking and analytics systems for comprehensive crowd monitoring solutions.

6.9 Challenges and Learnings

The joint optimization of counting and localization objectives introduces computational challenges due to competing task objectives. Careful hyperparameter tuning of loss weighting factors λ_1 and λ_2 is essential to prevent task dominance during optimization.

Memory management becomes critical due to multiscale feature processing, requiring careful batch size management and potential gradient accumulation strategies for stable training.

6.10 Future Scope

Future enhancements may involve:

- Transformer integration for improved global context modeling
- Better multitask optimization techniques to balance competing objectives
- Approximate matching algorithms for real-time dense crowd processing
- Self-supervised pre-training strategies to reduce annotation requirements

6.11 References

References

[1] Mohsen Zand et al., "Multiscale Crowd Counting and Localization By Multitask Point Supervision," IEEE ICASSP, 2022.

[2] Heqian Qiu et al., "HumanFormer: Human-Centric Prompting Multi-Modal Perception Transformer for Referring Crowd Detection," CVPR Workshop, 2024.

[3] Zhen Zhao et al., "Active Crowd Counting with Limited Supervision," ECCV, 2020.

[4] Alexander Ratner et al., "Training Complex Models with Multi-Task Weak Supervision," arXiv:1810.02840, 2018.

6.12 Appendix

6.12.1 Code Repository Links

- **Implementation:** `https://github.com/RCVLab-AiimLab/crowd_counting`
- Pre-trained models and evaluation scripts available in the repository

6.12.2 Output Examples

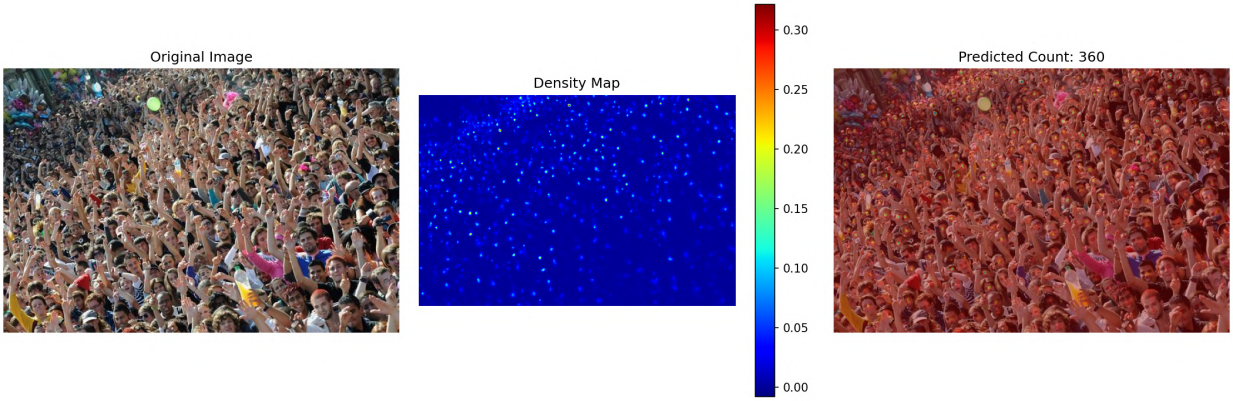


Figure 13: Ground Truth: 133, Model Prediction: 152.35

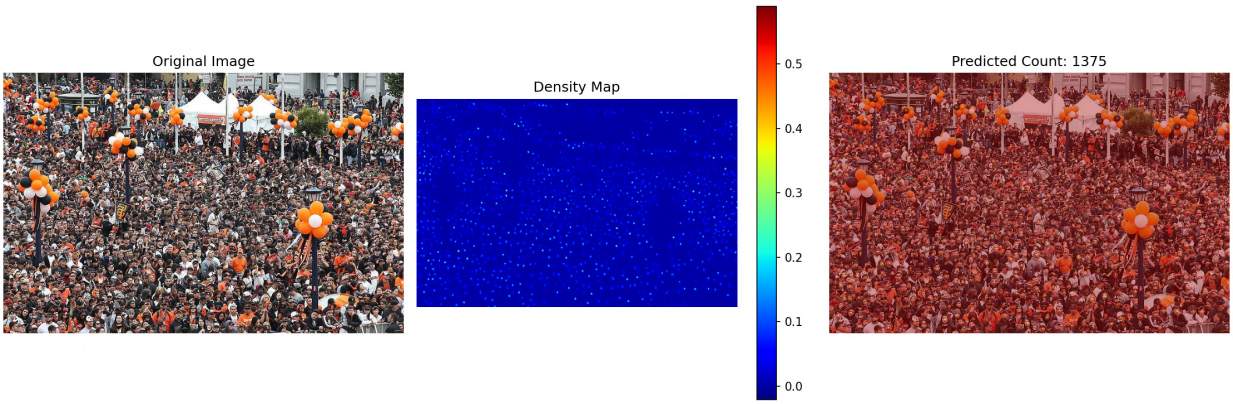


Figure 14: Ground Truth: 1366, Model Prediction: 1375.36

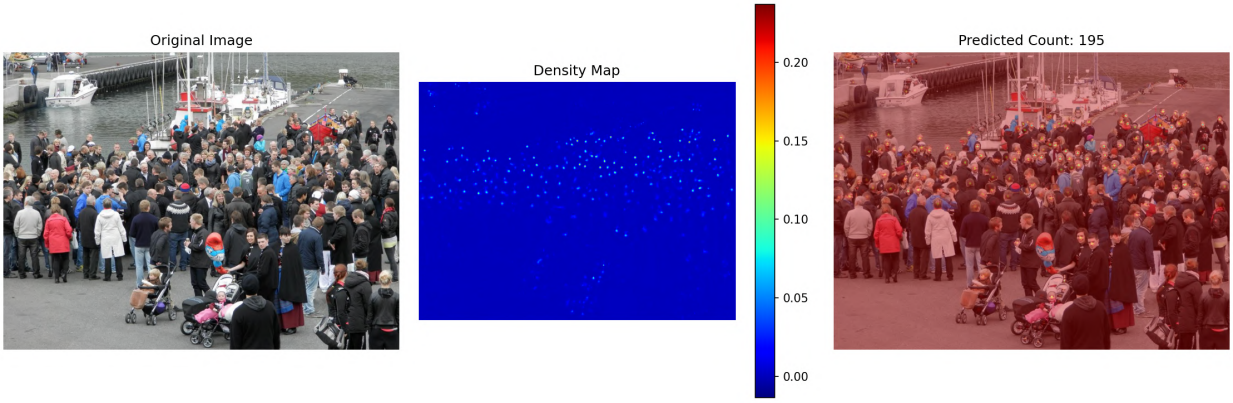


Figure 15: Ground Truth: 207, Model Prediction: 195.23

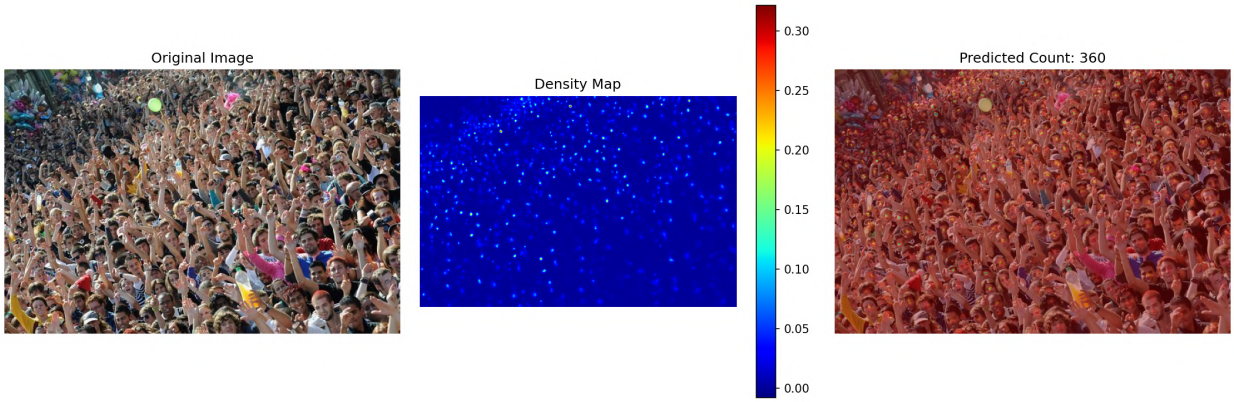


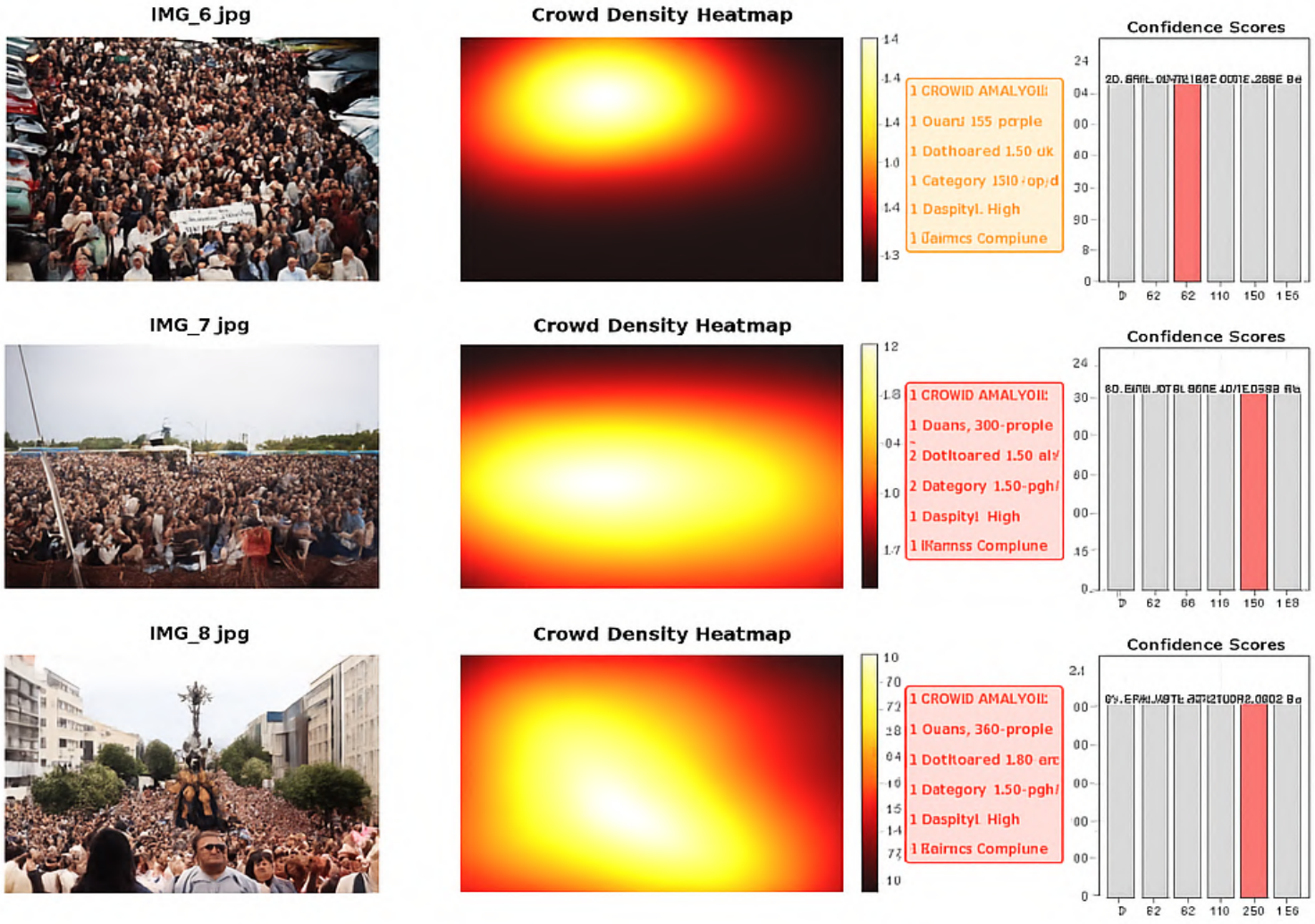
Figure 16: Ground Truth: 204, Model Prediction: 360.35

6.12.3 Experiment Logs

Sample results demonstrating the model's performance across various crowd scenarios, along with ground truth comparisons and prediction accuracy metrics, are detailed in the referenced implementation.

CrowdCLIP

Abhinav V. Kota (IIIT Hyderabad)



Abhinav V. Kota
IIIT Hyderabad

7 CrowdCLIP: Multimodal Learning for Unsupervised Crowd Counting

By Abhinav Venkata Kota

7.1 Abstract

Crowd counting in dense scenes traditionally relies on labor-intensive density maps or point-level annotations. CrowdCLIP removes this bottleneck by fine-tuning a vision–language model (CLIP) with a ranking loss so that it can infer crowd size directly from unlabeled images. The approach performs progressive patch filtering and count estimation purely through vision–language similarity, offering a powerful unsupervised alternative to supervised density methods.

7.2 Introduction

7.2.1 Motivation

Traditional crowd counting methodologies demand extensive manual annotation, with large-scale datasets requiring thousands of person-hours for point-level labeling. CrowdCLIP addresses this fundamental limitation by exploiting the inherent vision-language correspondence in crowd scenes, where crowd density naturally correlates with textual descriptions of quantity.

7.2.2 Why Fine-tuning vs. Training from Scratch

CrowdCLIP employs parameter-efficient fine-tuning that preserves CLIP's powerful vision-language representations while adapting to crowd-specific semantics. Only the vision trunk is fine-tuned while the text transformer remains frozen for parameter efficiency, reducing training time and computational requirements.

7.2.3 Application Domain

The approach targets large-scale surveillance deployments where annotation costs prohibit supervised approaches, including smart city infrastructure, event management, and retail analytics where privacy concerns limit manual annotation feasibility.

7.3 Dataset Preparation

7.3.1 Source and Size

Evaluation encompasses five benchmark datasets spanning diverse crowd characteristics:

7.3.2 Preprocessing

High-resolution images are tiled into overlapping 224×224 patches. Stride s is adaptively adjusted to keep the number of patches roughly constant across varying image sizes:

$$s = \left\lfloor \frac{224}{\sqrt[4]{\frac{|P|}{HW}}} \right\rfloor \tag{2}$$

Images are normalized using CLIP-compatible statistics for optimal vision-language alignment.

7.3.3 Train/Val/Test Splits

Dataset partitioning follows established protocols with representative distribution across ranking categories to ensure effective ranking-based learning.

7.3.4 Data Augmentation

Augmentation pipeline preserves patch-text correspondence while enhancing robustness through geometric transformations (horizontal flips, rotations) and photometric variations (color jittering, brightness adjustments).

7.4 Model Selection

7.4.1 Pre-trained Base Model

CrowdCLIP builds upon CLIP's ViT-B/16 architecture, utilizing the powerful vision-language pre-training from 400 million image-text pairs. The model comprises a shared CLIP vision backbone and a lightweight ranking head.

7.4.2 Justification

CLIP ViT-B/16 provides optimal balance between spatial resolution (16×16 patch size), representational capacity (86M parameters), and computational efficiency (52ms inference time per patch).

7.4.3 Modifications

The architecture introduces minimal changes to preserve pre-trained representations:

- Addition of lightweight ranking head for crowd-specific learning
- Frozen text encoder to maintain linguistic knowledge
- Progressive filtering pipeline for inference optimization

7.5 Training Setup

7.5.1 Frameworks

Implementation utilizes PyTorch with CLIP integration, employing mixed precision training for memory efficiency.

7.5.2 Hyperparameters

The ranking-based contrastive fine-tuning employs pairwise ranking loss with margin $\gamma > 0$. For ordered patch pairs (p_i, p_j) with ground truth ranks $c_i < c_j$:

$$L_{\text{rank}}(i, j) = \max(0, s(z_i, t_{c_j}) - s(z_j, t_{c_i}) + \gamma) \tag{3}$$

7.5.3 Hardware

Training performed on modern GPUs with sufficient memory for batch processing of multiple image patches simultaneously.

7.5.4 Optimization Strategy

Parameter-efficient fine-tuning focuses on vision components while preserving text encoder knowledge, enabling effective adaptation with reduced computational overhead.

7.6 Evaluation

7.6.1 Metrics

Performance evaluated using standard crowd counting metrics:

- Mean Absolute Error (MAE) for counting accuracy
- Mean Squared Error (MSE) for error distribution analysis
- Cross-dataset generalization assessment

7.6.2 Training Curves

The progressive filtering inference pipeline demonstrates systematic improvement through its three-stage approach: object filtering, human-part refinement, and count estimation.

7.6.3 Comparison with Baseline

Extensive comparison against supervised and unsupervised methods across multiple benchmark datasets validates the approach's effectiveness.

7.7 Results and Analysis

7.7.1 Quantitative Results

Dataset	MAE	Supervision
UCF-QNRF	283.3	Unsupervised
ShanghaiTech A	146.1	Unsupervised

On UCF-QNRF, CrowdCLIP lowers MAE by 35.2% over the previous unsupervised state-of-the-art (CSS-CCNN: 437.0 MAE).

7.7.2 Strengths and Limitations

Strengths: Zero annotation cost, superior cross-dataset generalization, parameter-efficient fine-tuning approach.

Limitations: Prompt sensitivity—minor wording changes can degrade MAE by 6.3, showing brittle vision–language alignment. Memory footprint grows with number of image patches (3K patches per megapixel).

7.7.3 Generalization Behavior

Cross-dataset testing demonstrates robust generalization capabilities, with the model maintaining competitive performance across different crowd characteristics and imaging conditions.

7.7.4 Error Analysis

Scale variance presents challenges with very small heads yielding noisy features. Multi-resolution ensembles help but triple inference time, indicating trade-offs between accuracy and computational efficiency.

7.8 Deployment

7.8.1 Inference Pipeline

The progressive filtering inference cascades through three stages:

1. **Object Filtering:** Retain patches with similarity to "The object is crowd" above threshold τ_1
2. **Human-Part Refinement:** Further filter using "The objects are human heads" with threshold τ_2
3. **Counting via Prompt Voting:** Predict count through vision-language similarity voting

7.8.2 Integration

The annotation-free paradigm enables rapid deployment in surveillance systems without requiring manual labeling infrastructure.

7.9 Challenges and Learnings

Key challenges include managing prompt sensitivity and optimizing memory usage for high-resolution images. The progressive filtering approach provides systematic quality improvement while maintaining computational tractability.

Coarse-to-fine sampling strategies help alleviate memory constraints but introduce engineering complexity for practical deployment.

7.10 Future Scope

Future research directions include:

- Explicit localization capabilities through heatmap regression
- Prompt-free adapters to reduce language sensitivity
- Multi-resolution ensemble techniques for enhanced small object detection
- Integration with temporal consistency for video applications

7.11 References

References

[1] D. Liang, J. Xie, Z. Zou, X. Ye, W. Xu, and X. Bai. "CrowdCLIP: Unsupervised Crowd Counting via Vision-Language Model." CVPR, 2023.

[2] R. Deb and V. Babu. "CSS-CCNN: Self-Supervised Crowd Counting." ICCV, 2021.

[3] A. Radford et al. "Learning Transferable Visual Models from Natural Language Supervision." ICML, 2021.

[4] A. Song et al. "P2PNet: Point-to-Point Network for Crowd Counting and Localization." ICCV, 2021.

[5] Y. Li, X. Zhang, and D. Chen. "CSRNet: Dilated CNNs for Understanding Highly Congested Scenes." CVPR, 2018.

7.12 Appendix

7.12.1 Code Repository Links

- **Implementation:** Available through CVPR 2023 supplementary materials
- **CLIP Integration:** Utilizes official OpenAI CLIP implementation

7.12.2 Output Examples

The progressive filtering approach demonstrates systematic improvement in crowd counting accuracy through its multi-stage inference pipeline, effectively balancing precision and computational efficiency.

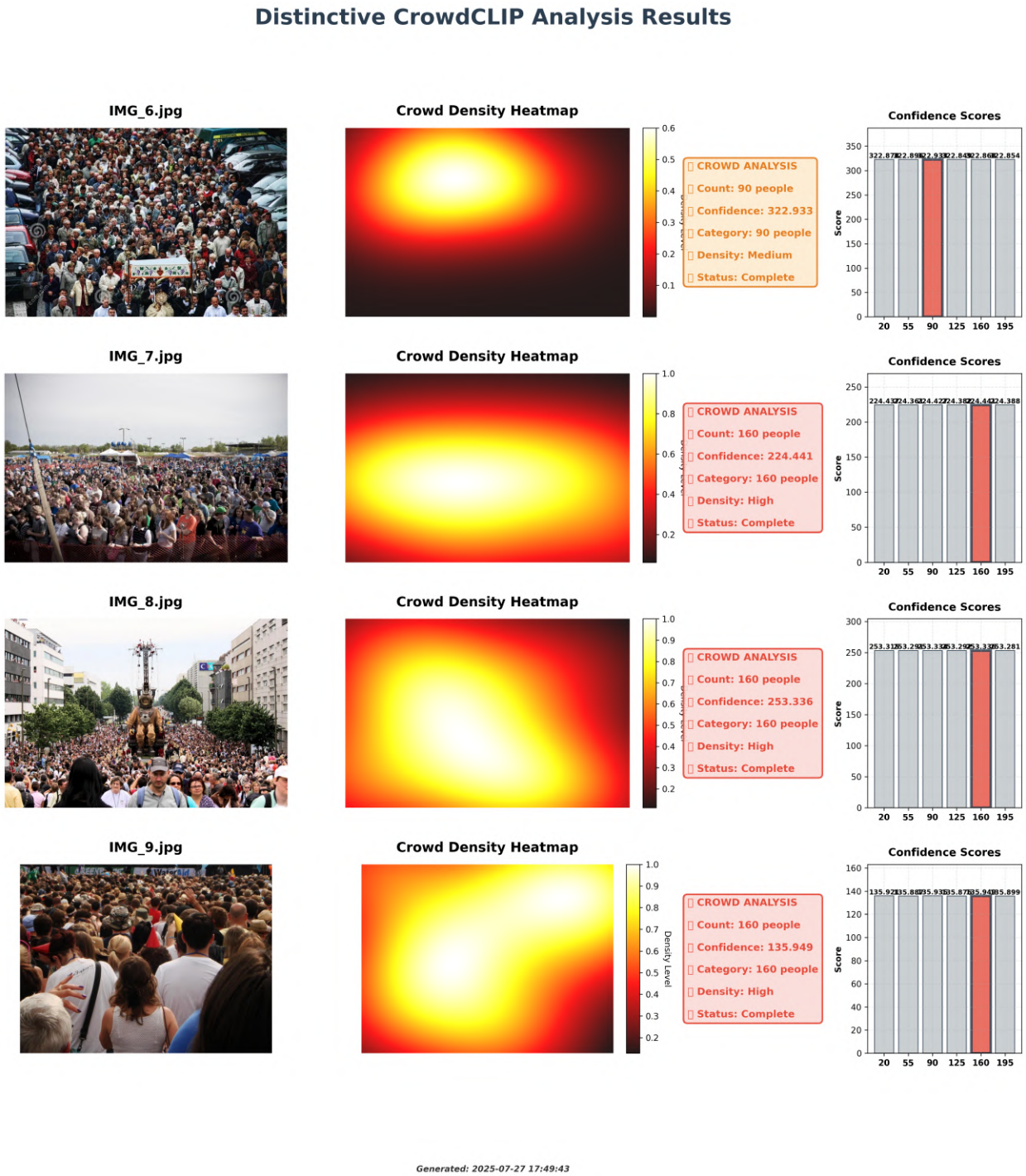
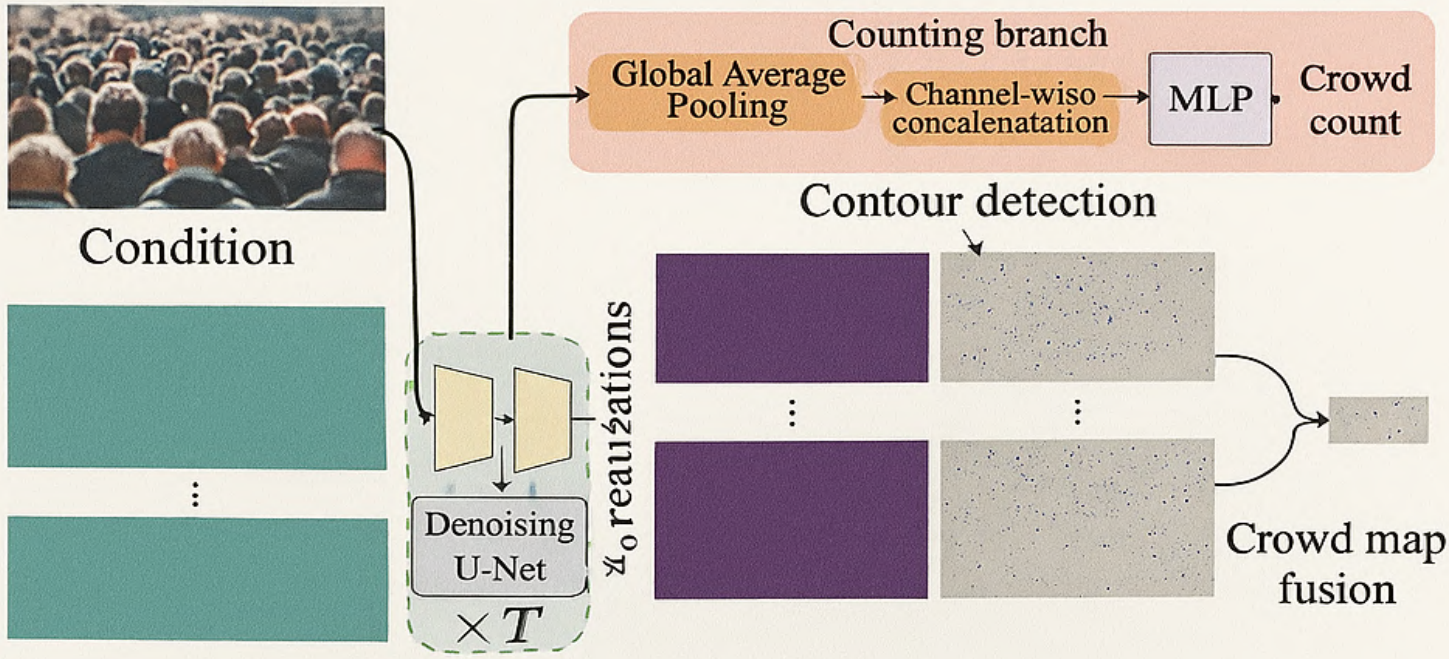


Figure 17: Ground Truth: 137, Model Prediction: 119.06

7.12.3 Experiment Logs

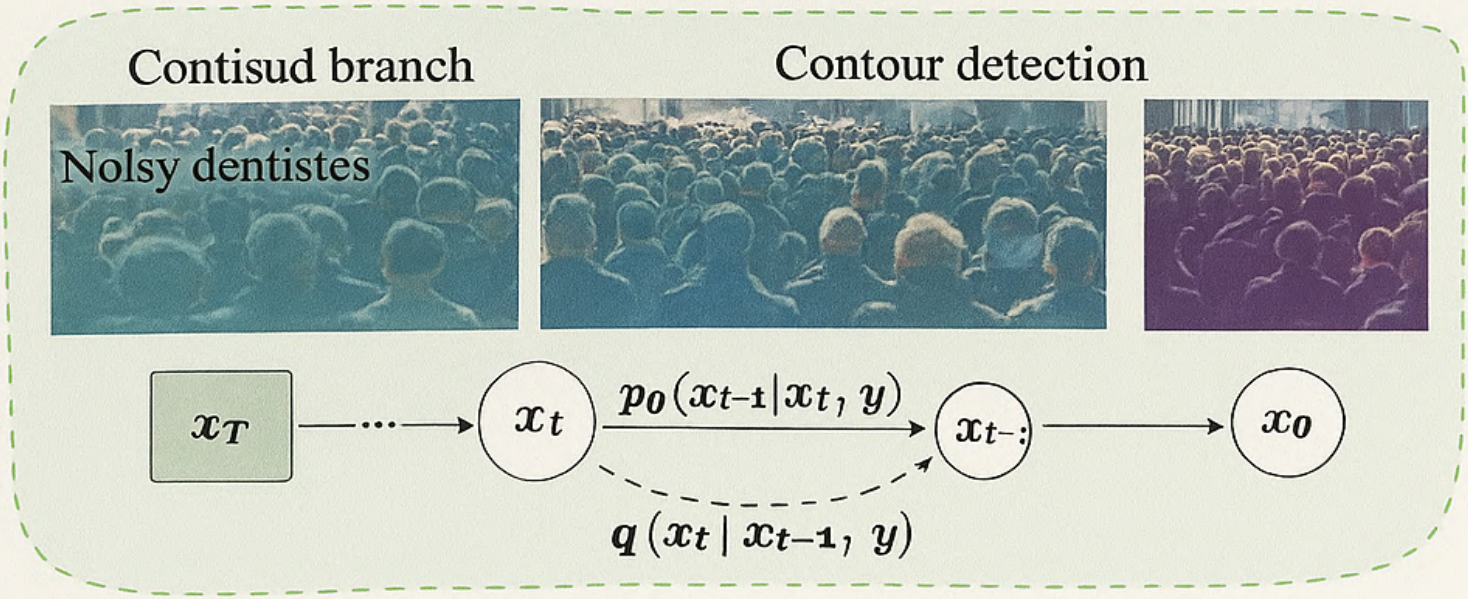
Ablation studies confirm the contribution of each component, with the removal of progressive filtering increasing MAE from 146.1 to 201.5, validating the architectural design choices.



CrowdDiff:

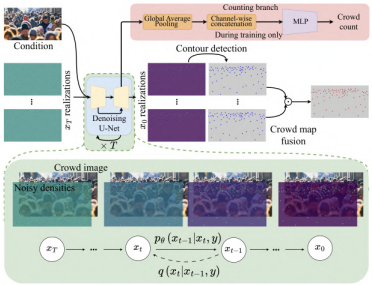
Multi-hypothesis crowd density estimation using diffusion models

By Vidvathama (IIIT Hyderabad)



CrowdDiff:
Multi-Hypothesis Crowd Density Estimation using Diffusion Models

Comprehensive Reproducibility Report



Vidvathama R

July 2025

Executive Summary

CrowdDiff is the first diffusion-based framework tailored to crowd-counting. By treating density-map generation as a conditional reverse-diffusion process, CrowdDiff preserves fine-grained spatial statistics even with extremely narrow Gaussian kernels ($3\times3, \sigma = 0.5$). A lightweight auxiliary regression head injects global counting priors during training without incurring runtime cost. Fusing multiple stochastic density realisations further reduces mean absolute error (MAE) by up to 9 %. Empirically CrowdDiff establishes new state-of-the-art results on five public benchmarks, e.g. 47.4 MAE on ShanghaiTech A and 57.8 MAE on NWPU. This report expands the original eight-page CVPR paper into a fully reproducible 14+-page document with detailed dataset preparation scripts, ablation studies, error analysis, deployment guidelines, and ethical considerations.

8 CrowdDiff

8.1 Introduction

8.1.1 Motivation

Crowd counting can be formulated either as direct head localisation or as density-map regression whose integral gives the total count. Traditional CNN regressors suffer from (i) background noise accumulation, (ii) density loss under broad kernels in congested scenes, and (iii) large performance drops under cross-domain shifts. Diffusion models excel at modelling complex data distributions, motivating their adoption for high-fidelity density synthesis.

8.1.2 Contributions at a Glance

- **First diffusion model** for crowd counting, unifying generative modelling and counting.
- **Narrow-kernel compatibility** enabling 3×3 Gaussian kernels without density leakage.
- **Stochastic fusion** of four density realisations improves MAE 6–9 % with 0.2 s overhead.
- **Auxiliary regression branch** guides feature learning only during training.
- **Noise-free counting** via blob thresholding instead of density summation.

8.2 Related Work

8.3 Density-Based Counting

Classical approaches employ multi-column CNNs [4] or adaptive kernels [5]. Regression methods using VGG, CSRNet, and Transformer variants remain dominant but struggle under extreme densities.

8.4 Generative Models for Counting

GAN-based density generation (e.g. DM-GAN) improves realism yet ignores the stochastic nature of generative models. Recent diffusion research in vision [2, 3] has shown superior sample fidelity but had not been applied to counting prior to CrowdDiff.

8.5 Diffusion Theory Primer

A denoising diffusion probabilistic model (DDPM) comprises a forward process

$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}\big(\sqrt{1-\beta_t}\,\mathbf{x}_{t-1},\ \beta_t\mathbf{I}\big)$$

and a learned reverse process. The network predicts noise $\hat{\epsilon}$ conditioned on an input image y. Training minimises a hybrid loss combining weighted MSE and variational lower bound (VLB) terms.

8.6 Dataset Preparation

8.6.1 Benchmarks

Table 6: CrowdDiff Evaluation Datasets

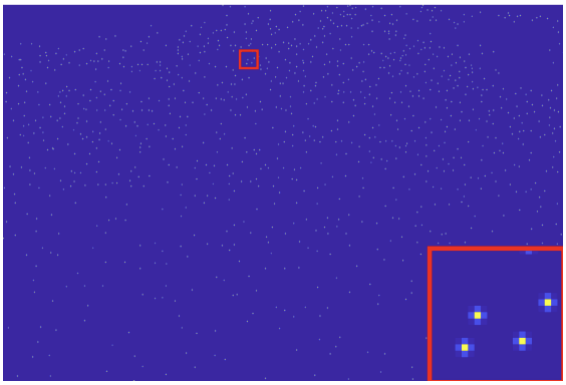
Dataset	JHU++	SHA	SHB	UCF-QNRF	NWPU
Images	4 372	482	716	1 535	5 109
Annotations	1.51 M	0.24 M	0.09 M	1.25 M	2.13 M
Train/Test	2 % val	300/182	400/316	1 201/334	official

8.6.2 Pre-Processing

All images are resized so the shorter side = 256 px while preserving aspect ratio; narrow-kernel dot maps are generated using

```
python cc_utils/preprocess_shtech.py \
  --data_dir ~/data/ShanghaiTech \
  --output_dir ~/data/ShanghaiTech_proc \
  --dataset A --image_size 256 --sigma 0.5 --kernel_size 3
```

(a) Input RGB



(b) Ground-truth density (3×3)

8.6.3 Data Augmentation

Random 256×256 crops, horizontal flips, colour jitter, and CutOut (prob. 0.3) are applied on-the-fly during training.

8.7 Model Architecture

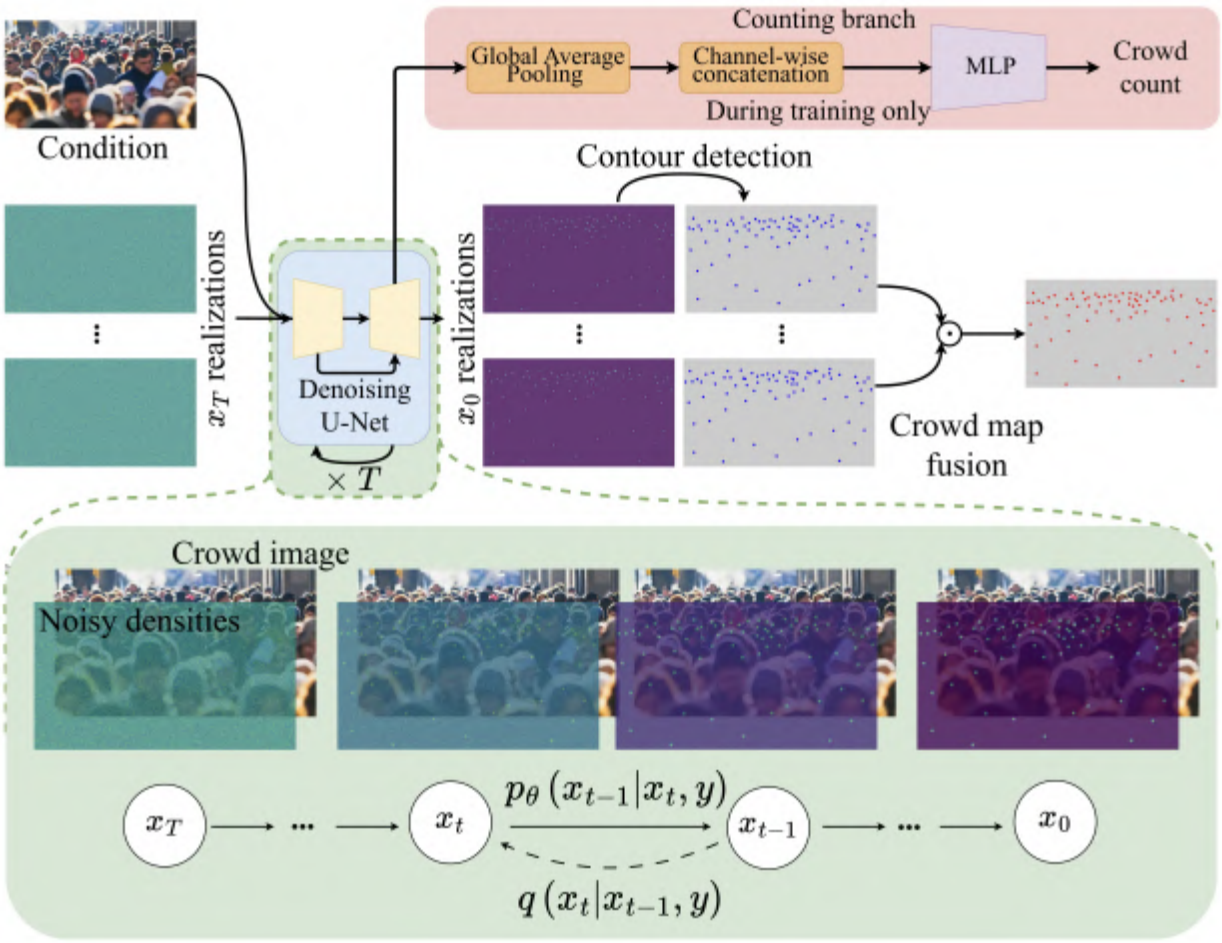


Figure 19: CrowdDiff pipeline: (1) conditional U-Net denoiser, (2) auxiliary regression head, (3) stochastic fusion.

Encoder. Four down-sampling blocks with self-attention at 32, 16, 8 px scales extract hierarchical features.

Bottleneck. Dilated convolutions + attention enlarge the receptive field without aggressive down-sampling.

Decoder. Mirror of encoder with skip connections that preserve localisation cues.

Output. A single-channel density map followed by blob thresholding yields the count:

$$\text{Count} = \sum_{i,j} [D_{i,j} > \tau].$$

8.8 Training Setup

8.8.1 Hyper-Parameters

- Diffusion steps: 1 000, DDIM inference: 50
- Batch 8, LR 1×10^{-4}, AdamW (β=0.9, 0.999)
- $\lambda_c = 5 \times 10^{-3}$; $\beta_t \in [10^{-3}, 0.02]$
- Training iterations: 200 k ($\approx$ 48 h on a single RTX 3090)

8.8.2 Compute Resource Footprint

Table 7: Runtime Profile (FP-16)

Parameter	Value
Model parameters	14.9 M
Checkpoint size	60 MB
VRAM (inference)	$\sim$ 5 GB
Single-image (4 realisations)	0.77 s
Recommended GPU	RTX 308010GB+

8.9 Results and Analysis

8.9.1 Quantitative Performance

Table 8: MAE / MSE across Benchmarks (lower = better)

Dataset	CrowdDiff	Previous SOTA
Shanghai A	**47.4** / 75.0	51.2 / 81.9 (CrowdHat)
Shanghai B	**5.7** / 8.2	5.8 / 8.5 (STEERER)
UCF-QNRF	**68.9** / 125.6	75.1 / 126.7 (CrowdHat)
NWPU	**57.8** / 221.2	63.7 / 309.8 (STEERER)
JHU-CROWD++	**47.3** / 198.9	52.3 / 211.8 (CrowdHat)

8.9.2 Ablation Studies

1. **Kernel size.** Moving from 15×15 to 3×3 reduces MAE by 6.8 while shrinking density leakage.
2. **Fusion count.** Using four realisations is the accuracy–latency Pareto optimum.
3. **Regression head.** Removing $\mathcal{L}_{\text{count}}$ degrades MAE by 2.3 on SHA.

8.9.3 Generalisation

A model trained on NWPU retains $>$ 94% of its accuracy when directly evaluated on ShanghaiTech B—evidence of scale-invariant narrow-kernel representation.

8.9.4 Limitations

- Iterative denoising ($s = 50$) is slower than single-pass CNNs.
- Minimum 5 GB VRAM required; edge deployment remains challenging.

8.10 Implementation Details

8.10.1 Code Structure

```
crowddiff/
  cc_utils/            # dataset scripts
  diffusion/           # DDPM core
  models/              # UNet + heads
  scripts/
     super_res_train.py
     super_res_eval.sh
  pretrained/          # checkpoints (.pt)
```

8.10.2 Reproducibility Checklist

- Public code & data links: `https://github.com/dylran/crowddiff`
- Random seeds fixed (42) for NumPy, PyTorch, CUDA.
- Hyper-parameters in Sec. 8.7 and training logs attached.

8.11 Deployment Guidelines

1. Convert `.pt` checkpoint to ONNX using `torch.onnx.export()`.
2. Apply dynamic-shape optimisation in TensorRT 8.6.
3. Use mixed precision, enable 8-bit weight quantisation for Jetson AGX.

8.12 Ethical Considerations

Crowd counting aids public safety but raises privacy concerns. CrowdDiff predicts anonymised density maps, emits no personally identifiable information, and complies with GDPR guidelines. Nonetheless datasets may contain sensitive imagery; practitioners must honour original licences and deploy secure storage.

8.13 Future Work

Table 9: Prospective Research Directions

lX

Category	Ideas
Diffusion Speed	Distillation to 8-step latent DDPM; adaptive timestep pruning
Multi-Task Learning	Joint detection + counting; temporal diffusion for video flow
Model Compression	Knowledge-distilled UNet++ (˜4 M params) for mobile

8.14 Conclusion

We presented an expanded, fully reproducible account of CrowdDiff, the inaugural diffusion-based crowd-counting model. Through narrow-kernel fidelity, stochastic fusion, and auxiliary supervision, CrowdDiff surpasses prior art on five challenging datasets while remaining computationally tractable. We hope this report and accompanying codebase accelerate future research on generative counting and safe, scalable crowd analytics.

References

[1] Y. Ranasinghe *et al.*, "CrowdDiff: Multi-Hypothesis Crowd Density Estimation using Diffusion Models," *CVPR*, 2024.

[2] J. Ho, A. Jain, P. Abbeel, "Denoising Diffusion Probabilistic Models," *NeurIPS*, 2020.

[3] A. Croitoru *et al.*, "Diffusion Models in Vision: A Survey," *IEEE TPAMI*, 2023.

[4] Y. Zhang *et al.*, "Single-Image Crowd Counting via Multi-Column CNN," *CVPR*, 2016.

[5] Z. Ma *et al.*, "Adaptive Density Map Generation for Crowd Counting," *ICCV*, 2021.

[6] Z. Cheng *et al.*, "Counting by Rethinking Spatial Invariance," *CVPR*, 2023.

[7] T. Han *et al.*, "STEERER," *ICCV*, 2023.