A Comparative Study of Deep Learning Models for Crowd Counting and Localization

Aaarat Chadda, Nipun Yadav, Taher Merchant

ABSTRACT

Crowd counting, the task of estimating the number of individuals in a given area, is critical for numerous applications, including urban planning, public safety management, traffic flow analysis, and large-scale event monitoring. Despite significant advancements in deep learning, accurately counting and localizing people in highly congested and diverse real-world scenarios remains a formidable challenge. Current approaches struggle with issues such as severe occlusion, significant scale variations, non-uniform crowd distribution, and perspective distortion, leading to considerable errors in dense environments.

This paper presents a comprehensive comparative study of four distinct deep learning models, each designed to address the complexities of crowd counting and localization. Our research explores various architectural strategies, including novel multi-scale feature learning and attention mechanisms, to enhance density map estimation and refine individual localization. Beyond quantitative evaluation on standard benchmarks, we validate the practical utility of these models by deploying and testing their performance on a live camera feed, demonstrating their real-time applicability and robustness in dynamic, uncontrolled environments. This work aims to provide insights into effective architectural choices and robust post-processing techniques for real-world crowd analysis.

VARIOUS MODELS USED :

1) Yolo V8 fine tuning on database with heads
2) CC Trans (ViT based)
3) Crowd Diff
4) Multi Scale Attention Density
5) DM count ++ with attention
6) P2P net
7) SE net

CCTRANS

Thought Process

When I first set out to simplify crowd counting, I was struck by how most CNN-based methods juggle multiple branches, attention modules, and complex fusion schemes just to capture different scales and densities. The CCTrans paper's core idea to leverage a Vision Transformer backbone for its global receptive field and then stitch together multi-scale features with minimal decoder overhead immediately resonated with me

Approach

Data Preprocessing

To start, I built a flexible data loader that could handle a wide variety of .mat annotation formats. Some datasets use different field names or nested structures, so I added a fallback recursive search to automatically find the head coordinates. Once extracted, I generate narrow-kernel density maps (to avoid overlap and maintain clarity), cache them as .npy files, and reload them during training to save time and computation.

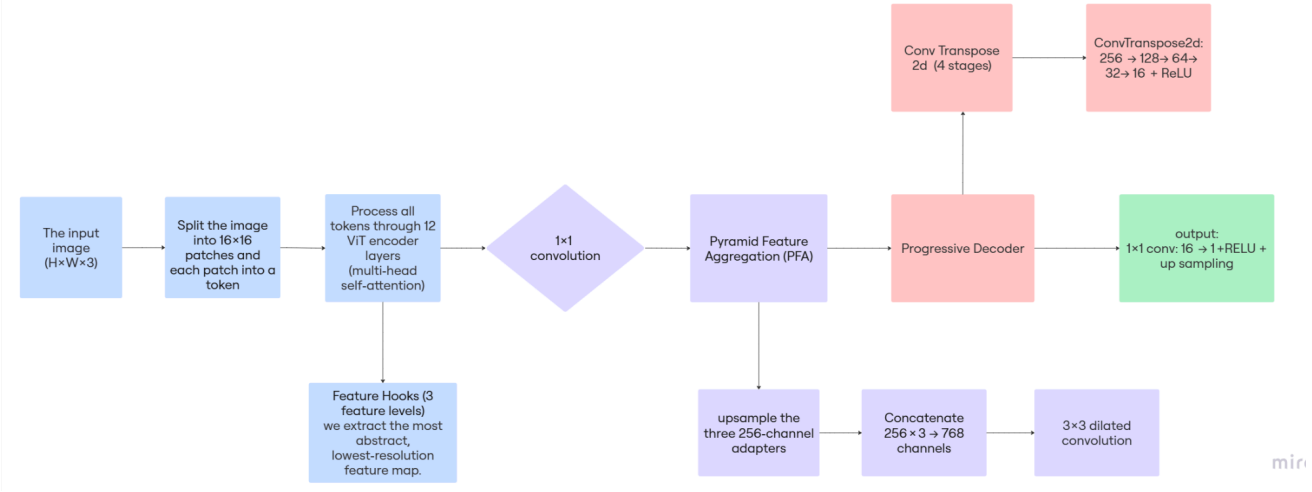
1) Backbone & Feature Extraction
   For feature extraction, I used a pre trained vit_b_16 model. I removed its classification head and tapped into three specific depths right after patch embedding, mid-way through the transformer stack, and at the final layer. These give me a nice spread of features from fine to coarse, all without needing to handcraft a multi-column CNN setup.
2) Decoder & Aggregation
   Each of the three extracted feature maps is first passed through a 1×1 convolution to reduce its channels to 256. I then upsample them to the same spatial resolution, concatenate them, and apply a single dilated 3×3 convolution. This step my Pyramid Feature Aggregation block efficiently fuses local and global context into one clean feature map.
3) Regression Head
   To produce the final density map, I designed a lightweight regression head. It has three parallel branches, each with a conv followed by a dilated conv (with dilation rates 1, 2, and 3), and a shortcut connection alongside. I concatenate the outputs of all branches and run them through a 1×1 convolution to produce a single-channel density map precise, clean, and ready for loss calculation.

Blockers and solutions:

1) Small conv kernels only see local context, so distant dependencies (e.g. sparse vs dense regions) aren't modeled. self-attention across all patches ensures every location can attend to every other, capturing both all details

2) Scale Variation Heads vary in size and density across the images. So using Pyramid Feature Aggregation (PFA) Explicitly taps three resolutions (high/mid/low), aligns them spatially, and fuses with dilation to handle scale diversity.

Architecture overview



I use a pretrained ViT backbone to extract features at three levels—patch embedding, mid-layer, and final layer.Each feature map is reduced to 256 channels, upsampled to the same size, and fused using a simple Pyramid Feature Aggregation block.This fused map is passed through a lightweight decoder made of transposed convolutions to reconstruct spatial details.Finally, I use a 1×1 conv to generate the density map, followed by ReLU and bilinear upsampling to match the original resolution.The architecture is clean, efficient, and handles scale variation really well .

How we manually finetuned our model :

1) Backbone Changes: ViT vs. Twins-SVT : Twins-SVT's alternating attention was complex to reimplement. ViT offered a quicker path to validate the global-context idea. Also VIT offered Multi-scale feature extraction .

2) Changed Regression Head from Simplified to Multi-scale Dilated Convolution(MDC): Prioritized speed over optimal scale handling.but gave decent results.

Novelty

1. Transformer-Powered Feature Pyramid CCTrans replaces cumbersome multi-column CNNs with a simple, unified ViT backbone that natively captures both local and global context, then aggregates its multi-scale outputs with minimal overhead .
2. Multi-Head Parallelism Splits attention into multiple heads, each learning a different relation pattern (e.g. local texture vs. global layout), then recombines them providing richer feature interactions than a single attention map.

Evaluation metrics

```
--- Evaluation Complete ---
MAE: 71.36, RMSE: 94.35

ALL STEPS COMPLETED!
```

Crowd diff

Thought process:

When I first encountered the problem of accurate crowd counting, I was struck by two persistent challenges in density-based methods: (1) the accumulation of background noise when summing broad Gaussian kernels, and (2) degradation of density fidelity in highly congested regions due to kernel overlap. My goal was to explore whether recent advances in generative diffusion models could help address these issues. In particular, I was inspired by the CrowdDiff paper's insight that diffusion models can faithfully learn and reproduce a pixel-wise density distribution, even when using very narrow kernels

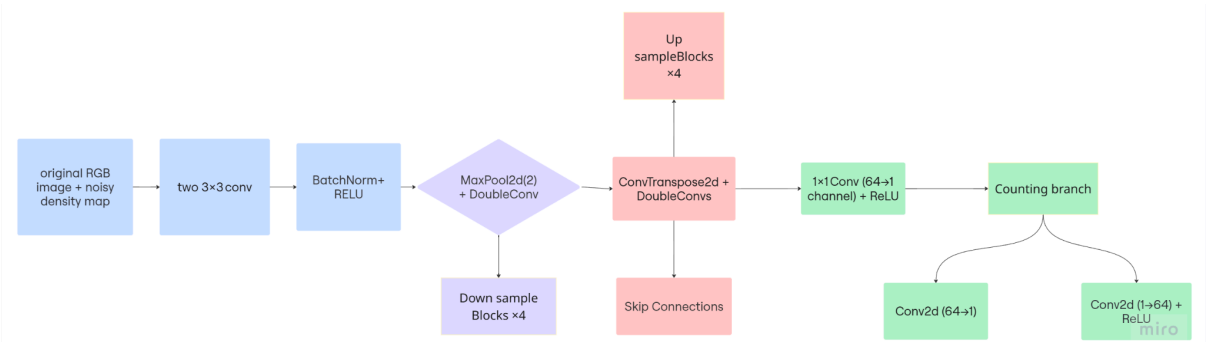
Approach

1. Data Preparation & Density Map Generation
   I start by writing a flexible PyTorch dataset that pulls both the raw images and their .mat annotations. Rather than keeping large precomputed density maps on disk, I generate tight, narrow-kernel ground-truth maps on the fly during the first pass and save each one as a simple .npy file. This way, I avoid head-point bleed and speed up subsequent epochs by reloading the cached maps.
2. Model Design & Training
   For the core model, I lean on a U-Net as my denoising backbone, then tack on a lightweight regression branch that learns to predict the total crowd count from intermediate features (only used during training). In each training step, I sample a random diffusion timestep, corrupt the true density map accordingly, run it through the U-Net to estimate the added noise, and compute a hybrid loss MSE on the noise prediction plus an L1 count loss weighted by the step's SNR. This keeps both the fine-grained map and the overall count honest.
3. Inference & Multi-Hypothesis Fusion
   When it's time to test, I run DDIM sampling a few times per image to get multiple "guesses" at the clean density map. I threshold each sample into discrete peaks, then cleverly merge them: I rank hypotheses by SSIM similarity, accept unique blobs within a small neighborhood radius, and reject duplicates. The result is a fused density map that's sharper and more reliable than any single sample.

Blockers and solutions

1) Count Variance Across Samples Single deterministic predictions can under or over count depending on local ambiguities. Multi-Hypothesis Sampling & Fusion CrowdDiff generates multiple plausible density realizations (via DDIM) and fuses them intelligently recovering missed points and averaging out spurious noise.
2) Background Noise Amplification Increases false positives and inflates counts in background areas.Treating density estimation as a task forces the model to actively remove noise at each timestep, yielding cleaner density maps.

Architecture overview



I built the model around a U-Net backbone that denoises corrupted density maps as part of a diffusion process.During training, I also added a small regression branch that predicts the total crowd count from intermediate features.At each step, the model learns to reverse noise added at random diffusion timesteps using a hybrid loss function.For inference, I generate multiple samples using DDIM and merge them by filtering duplicates based on similarity and distance.This approach helps produce sharper, more accurate density maps and count estimates, even in very dense scenes.

How we manually finetuned our model

1) I added BatchNorm and ReLU after every convolution for better training stability.
2) implemented a small Conv2d(1→64) + ReLU + Conv2d(64→1) module, followed by a sum() for obtaining final counting.
3) Loss Function Fine-Tuning implemented the hybrid loss using SNR-weighted MSE, which is critical for balancing early vs. late timesteps in the diffusion process.

Novelty

1) Diffusion-Based Density Mapping: Treating density estimation as a conditional reverse diffusion task allows the model to learn a full pixel-value distribution, avoiding distortions from broad kernels .

2) Multi-Hypothesis Fusion: Exploiting the stochastic nature of diffusion, we generate multiple density realizations and fuse them intelligently rather than simply averaging counts to recover missed points and reduce noise.
3) Addition of skip connections In each UNet up-block, concatenate both image and density encoder features via dual skip connections.
   This enriches spatial detail, improves head/background separation, and stabilizes diffusion training.

Evaluation metrics

```
--- Final Evaluation ---
Loading best model for final evaluation.
Evaluating: 100%|██████████| 316/316 [00:10<00:00, 30.48it/s]
Final Test MAE (Best Model): 67.66
Final Test RMSE (Best Model): 91.99

--- Generating Visualizations ---
Visualizations saved to: /content/drive/MyDrive/cctrans_visualizations
```

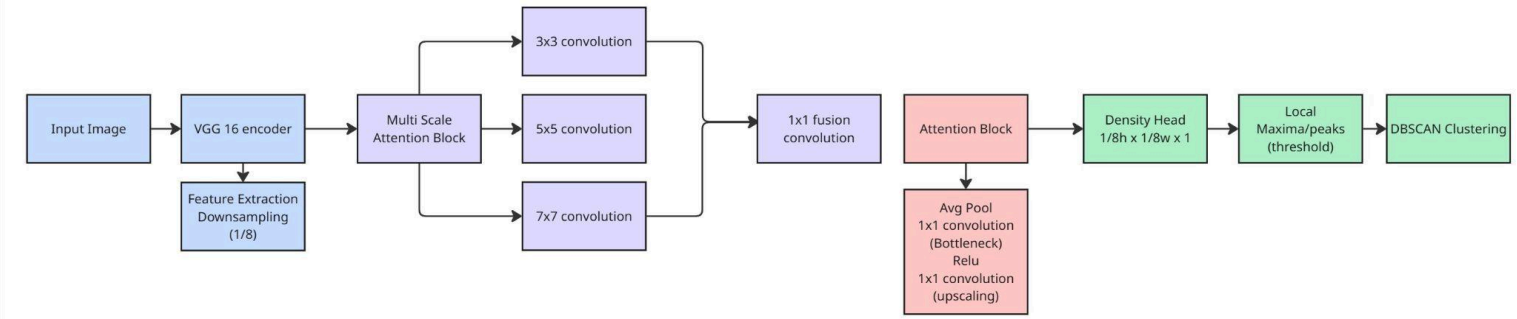
Multi-Scale Attention Density Network based approach

Our method leverages a deep convolutional neural network, adapted from VGG-16, to directly estimate high-resolution crowd density maps from input images. The proposed architecture incorporates Multi-Scale Feature Aggregation and Channel Attention Mechanisms to enhance feature representation and contextual understanding, crucial for handling varying crowd densities and perspectives. A key aspect of our approach is the post-processing pipeline that not only sums the predicted density map for precise crowd count estimation but also integrates a DBSCAN-based clustering mechanism applied to detected density peaks. This allows for robust individual localization, transforming the continuous density estimation into discrete point predictions. Experimental results on the given dataset (crowd_wala_dataset) benchmark demonstrate the effectiveness of our model in achieving competitive crowd counting accuracy (MAE, RMSE) while also providing valuable localized crowd distribution information.

BLOCKERS:

Density maps were returning a large number of points and were not able to separate between the points pertaining to a person and the background. Initial approaches also gave outputs with the entire image not detecting a single head and giving all pixels the same colour as background. This meant the model was not learning enough data and thus we had to add layers as well as an attention block. Next we were often receiving multiple detections for the same head which was leading to an increased count in the number of heads. Thus we introduced a post processing step of DBSCAN as it is robust to noise and we don't need to specify number of points for a cluster.

ARCHITECTURE:



Multi-Scale Attention Density Network, is a deep learning model for crowd counting and localization. It utilizes a VGG-16 backbone for initial feature extraction. The core innovation lies in its Multi-Scale Block (MSB), which uses parallel convolutions of varying kernel sizes (3x3, 5x5, 7x7) to capture diverse spatial features from a single input. Following the MSB, a Channel Attention Block refines these multi-scale features by adaptively weighing channel importance.

Finally, a Density Head outputs the crowd density map. For localization, a DBSCAN-based post-processing step clusters peaks in the predicted density map, providing discrete person detections in addition to the overall crowd count.

NOVELTY:

1. Explicit Multi-Scale Feature Learning

- Instead of using complex multi-column networks or dilated convolutions, this model uses parallel convolutions (3×3, 5×5, 7×7) on the same feature map.
- Why it matters: This is a simple yet effective way to capture scale variation in crowds, allowing efficient and end-to-end trainable multi-scale feature extraction.

2. Strategic Channel Attention

- A Channel Attention Block is applied *after* the multi-scale fusion step.
- Why it matters: It helps the network focus on the most relevant features for density prediction, suppressing noise and refining feature quality.

3. DBSCAN for Robust Localization

- Instead of basic peak detection, this uses DBSCAN clustering on local maxima of the density map.
- Why it matters: DBSCAN can handle varying crowd densities and filter out outliers, enabling precise head localization, not just total count.

RESULTS:

```
--- PyTorch Test Set Evaluation Metrics ---
Original Method (NMS):
  Mean Absolute Error (MAE): 76.22
  Root Mean Squared Error (RMSE): 85.37
DBSCAN Post-Processing (EPS=5, MIN_SAMPLES=1):
  Mean Absolute Error (MAE): 32.64
  Root Mean Squared Error (RMSE): 62.54
```

4. DMCount++ with Self Attention Module:

The DMCount++ model is a deep learning-based crowd counting model designed to accurately estimate the number of people in an image by predicting a high-resolution density map. It enhances the traditional CNN approach by integrating multi-scale contextual features and self-attention to adapt to varying crowd densities and spatial layouts. The model is trained to minimize the error between the predicted and ground truth density maps, enabling it to handle both sparse and highly congested scenes effectively.

Thought Process

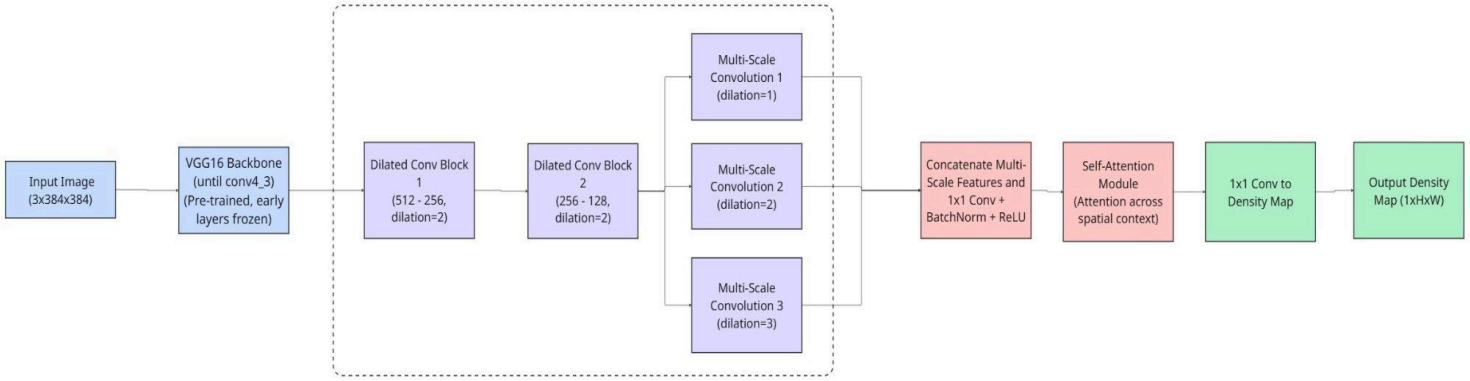
When I started working on crowd counting, I realized that traditional CNNs, while effective in extracting features, often struggle to adapt across scenes with different crowd densities and layouts. A single receptive field just isn't enough dense regions need wide context, while sparse ones benefit from finer detail.

That's what drew me to the DMCount++ architecture. It combines the strengths of CNNs with multi-scale context and self-attention, which felt like a natural evolution. I liked that it doesn't rely on multiple separate branches or complex heuristics instead, it uses dilated convolutions with different rates to simulate varying receptive fields in parallel. That made it more adaptable across scale variation, which is a huge issue in real-world crowd images.

On top of that, the self-attention module adds another layer of intelligence it helps the model focus more on actual crowd regions and ignore background noise, which is critical in scenes like stadiums or streets where clutter is everywhere.

The model outputs a high-resolution density map, which I found more interpretable than a single count value it tells you not just *how many* people, but *where* they are. And training it by minimizing the error between predicted and ground truth maps gave it the precision needed for both sparse and highly congested scenes.

ARCHITECTURE:



The architecture starts with a VGG-16 backbone truncated at the conv4_3 layer for feature extraction, followed by two dilated convolution blocks to expand the receptive field. This is succeeded by three parallel convolution blocks with different dilation rates (1, 2, 3) to capture multi-scale features. These outputs are concatenated and passed through a fusion layer (1×1 conv + BN + ReLU), refined using a self-attention module to enhance spatial focus, and finally projected to a single-channel density map through a 1×1 convolution. The output is optionally upsampled to match ground truth resolution.

For my model, I started with a VGG-16 backbone, but only used it up to the layer. That gave me strong mid-level features with enough spatial detail to work with, without overcomplicating the architecture.

Right after the backbone, I added two dilated convolution blocks to expand the receptive field. This helped the model gather broader contextual cues something really important when people are densely packed or far from the camera.

To handle scale variation more effectively, I used three parallel convolution blocks with different dilation rates (1, 2, and 3). These act like different “zoom levels,” allowing the model to pick up on small heads in the distance and larger ones up close. I then concatenated all three outputs and passed them through a 1×1 convolution with BatchNorm and ReLU to fuse them into a single, compact feature representation.

To sharpen the model’s attention on crowd regions and suppress background noise, I added a self-attention module. This step really helps guide the model to “look” where people actually are, especially in cluttered or complex scenes.

Finally, I used a 1×1 convolution to generate the density map. In some cases, I also upsampled the output to match the original image size or the resolution of the ground truth, making comparison and evaluation easier. The overall pipeline felt intuitive, efficient, and well-suited to both sparse and dense scenes.

APPROACH:

To build my model, I started by using a pretrained VGG-16 backbone, specifically up to the conv4_3 layer. This gave me access to strong low-level features that preserve spatial detail essential for identifying heads in dense scenes.

Next, I added a multi-scale feature extraction block using three parallel dilated convolutions with dilation rates of 1, 2, and 3. The idea was to capture crowd features at different receptive fields so the model could adapt to both sparse and dense regions without needing separate branches.

I then fused these outputs (each 64 channels) by concatenating them into a 192-channel tensor and passing it through a 1×1 convolution followed by BatchNorm and ReLU. This helped merge multi-scale information into a more unified and compact representation.

To help the model focus on meaningful regions, I included a self-attention module. By using query–key–value attention, it learns to highlight true crowd areas while suppressing background clutter something especially useful in complex scenes with distractions.

Finally, I predicted the density map using a simple 1×1 convolution. The output map is continuous, and its integral gives the estimated crowd count for the image. This straightforward pipeline helped balance spatial precision with scale adaptability and contextual awareness.

1. Backbone: Uses a pretrained VGG-16 network up to conv4_3 for rich low-level feature extraction.
2. Multi-Scale Feature Extraction: Three Dilated Convolutional Blocks with varying dilation rates (1, 2, 3) to extract features at different receptive fields. This helps the model adapt to variable object scales.
3. Feature Fusion: Outputs of the three blocks are concatenated (192 channels total), then passed through a 1x1 conv + BN + ReLU fusion layer.
4. Self-Attention Module: Enhances spatial contextual understanding using query-key-value operations to focus on relevant crowd regions and suppress background noise.
5. Density Map Prediction: A 1x1 convolution predicts the final density map, where the integral of the map approximates the person count.

Blockers Faced During Development:

- Colab GPU Performance Issues:
  The limited GPU resources on Google Colab, especially with free-tier usage, caused inconsistent runtime speeds and frequent memory bottlenecks. This led to slow training and evaluation, particularly when working with high-resolution images and large batch sizes. The environment sometimes disconnected during long training cycles, making it difficult to sustain model training across multiple epochs.
- Video Test Set Challenges:
  The model struggled to generalize well on the video test set. Since it was trained primarily on still images, temporal consistency and motion-related blur in video frames introduced noise and reduced counting accuracy. Additionally, processing video frames sequentially without temporal modeling made the predictions unstable across frames.
- Output Resolution Mismatch:
  The model outputs density maps at a lower resolution due to downsampling in the VGG-16 backbone. This required careful upsampling and interpolation to align with ground truth maps, introducing minor discrepancies that impacted evaluation metrics.
- Inconsistent Ground Truth Formats:
  The training and test ground truth .mat files had varying internal structures (e.g., image_info, annPoints, unnamed arrays), which required extensive exception handling and pre-processing to extract annotation points reliably.
- Limited Real-Time Capability:
  The inference time, especially with added self-attention layers, was not optimal for real-time applications. This was more pronounced in the Colab environment where GPU allocation time and I/O latency compounded the delay.
- Outputs predicted count as sum(density_map)
- Visualizes original image, ground truth density, and predicted density

Novelty of the DMCount++ Architecture:

The novelty of the DMCount++ model lies in its integration of dilated convolutions and a self-attention module within a crowd counting framework to enhance spatial feature representation and scale adaptability. Unlike standard CNN-based models that rely on fixed receptive fields, DMCount++ leverages multi-scale dilated convolutions to capture both fine-grained and coarse contextual features, allowing it to effectively handle variations in crowd density and object size. Additionally, the introduction of a self-attention module enables the network to focus on more informative regions in the feature map by modeling long-range spatial dependencies a capability missing in conventional convolutional architectures. These enhancements significantly improve the model's ability to distinguish crowd regions from

background noise and refine density map predictions, especially in complex, high-density scenes.

```
Evaluating model...
Found 316 images in /content/drive/MyDrive/crowd_wala_dataset/crowd_wala_dataset/test_data/images
Evaluation Results:
MAE: 25.00
MSE: 1239.61
RMSE: 35.21

Generating visualizations...
Found 316 images in /content/drive/MyDrive/crowd_wala_dataset/crowd_wala_dataset/test_data/images
```

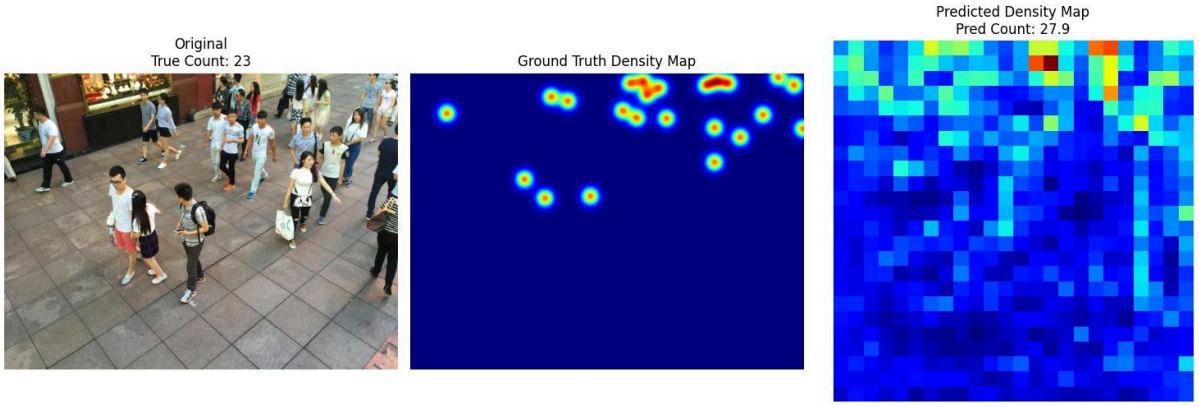
Results:

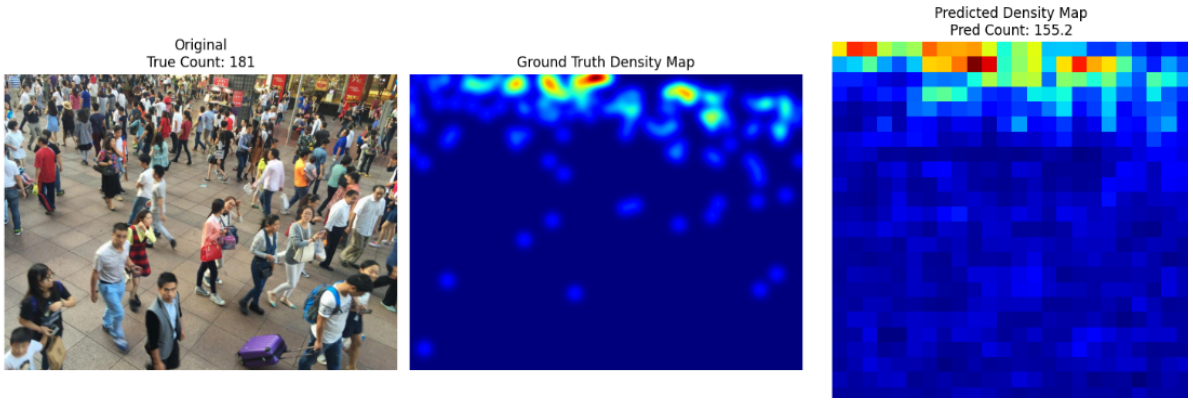
Comparative Study about Models

Model Name	MAE	RMSE	Strengths and Weakness of Models
1) CCTrans	71.36	94.35	Robust to scale,Clean Multi-Scale Design, very high computation cost
2)CrowdDiff	67.66	91.99	Generates multiple density samples and fuses them, Slower Inference
3)MultiScale Attention Density Network	32.64	62.54	Highly Accurate on Sparse Data. Fails to generalise for denser crowds
4) DMCount++	27.58	42.04	**Highly Accurate on Dense Data. Dilated Conv. help to capture more context due to increased Receptive Fields.**
5) SENet	122.89	154.54	Lightweight but has Limited Spatial Context:
6) DMCount++ Attention Modules	25.00	35.21	Highly Robust to scale, but more complex
7)P2P Net (No finetuning)	52.44	85.78	Simple, Not accurate for dense crowds
8) CSRNet	51.25	74.36	Strong Feature Extraction,uncertainty estimation .
9) YoloV8	-	-	Fast, Doesn't generalise well due to NMS and scaling

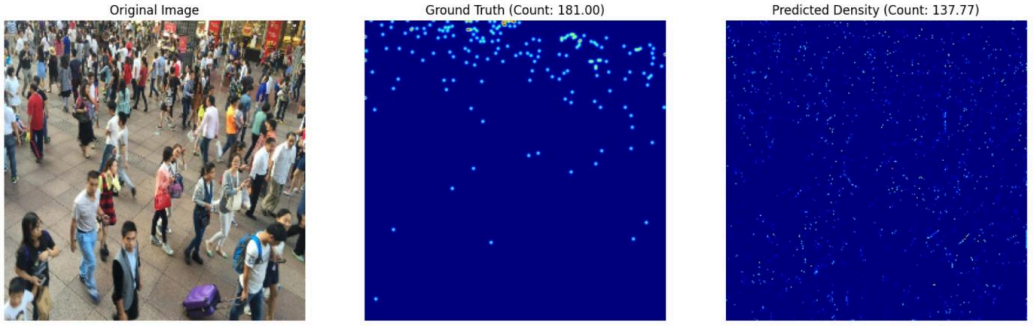
Density Map Visualisations/ Comparative Studies:

MultiScale Attention Density Network on (i) Sparse and (ii) Dense respectfully

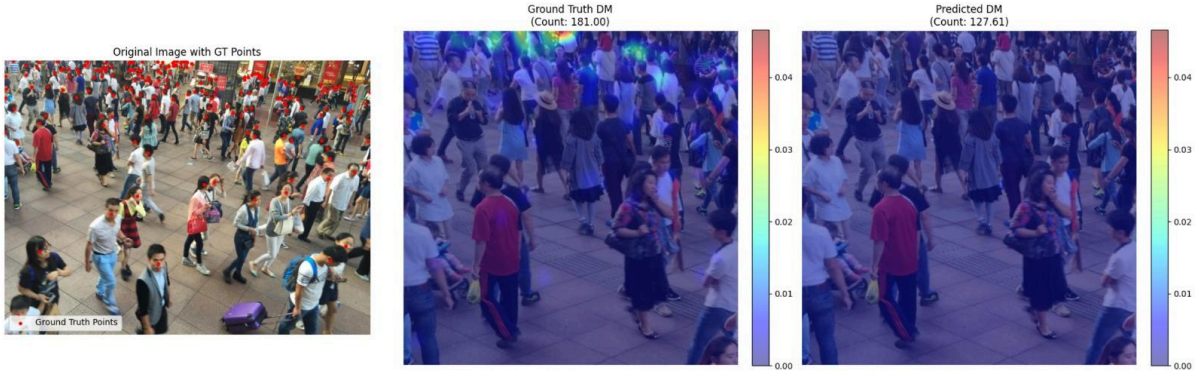




DMcount++self attention on (i) Sparse and (ii) Dense respectfully



Crowd Diff on (i) Sparse and (ii) Dense respectfully



CC Trans on (i) Sparse and (ii) Dense respectfully

Use case in Disaster Struck areas and rescue operations

This Emergency Response Crowd Detection System offers a vital tool for enhancing situational awareness and operational efficiency in disaster-struck areas and during rescue missions. By leveraging deep learning, the system provides real-time crowd density estimation and precise individual localization from live video feeds.

In emergency scenarios, this capability is critical. First responders can quickly ascertain the number of people in a given area, identify high-density zones, and pinpoint the exact locations of individuals, even amidst chaos and occlusion. This immediate, actionable intelligence allows for optimized resource deployment, ensuring medical teams, search-and-rescue units, and aid supplies are directed where they are most needed. We can deploy this tool to identify stranded personnel as well as a risk for stampede to identify most at risk personnel for evacuation.

The code for this feature is also added on the git repository