Integrating Convolutional and Graph Neural Networks for Enhanced Dense Crowd Counting

Romain TREFAULT
(supervisor: Pr. Nicolas Boutry)

Technical Report n^ooutput, June 2023
revision

This research project seeks to improve crowd counting and density estimation accuracy by combining Convolutional Neural Networks (CNNs) with Graph Neural Networks (GNNs), an innovative approach not yet widely explored in the field of crowd analysis. The main goal is to explore how these two architectures can work together effectively for crowd analysis tasks. By leveraging the spatial feature extraction capabilities of CNNs alongside the structural information processing strengths of GNNs, we aim to create a novel framework that improves performance in crowd counting. This research includes an explanation of the state of the art, detailing current methodologies and their limitations, followed by a presentation of our work that integrates CNNs and GNNs, and our results. Finally, we discuss potential improvements and future work to further enhance crowd management and surveillance techniques in different real-world situations.

Ce projet de recherche vise à améliorer la précision du comptage des foules et de l'estimation de la densité en combinant les réseaux neuronaux convolutifs (CNN) et les réseaux neuronaux graphiques (GNN), une approche innovante qui n'a pas encore été largement explorée dans le domaine de l'analyse des foules. L'objectif principal est d'explorer comment ces deux architectures peuvent fonctionner ensemble de manière efficace pour les tâches d'analyse des foules. En tirant parti des capacités d'extraction des caractéristiques spatiales des CNN et des forces de traitement de l'information structurelle des GNN, nous visons à créer un nouveau cadre qui améliore les performances en matière de comptage des foules. Cette recherche comprend une explication de l'état de l'art, détaillant les méthodologies actuelles et leurs limites, suivie d'une présentation de notre travail qui intègre les CNN et les GNN, et de nos résultats. Enfin, nous discutons des améliorations potentielles et des travaux futurs pour améliorer encore la gestion des foules et les techniques de surveillance dans différentes situations du monde réel.

Keywords
Crowd counting, Density estimation, Convolutional Neural Networks (CNNs), Graph Neural Networks (GNNs), Crowd analysis, Surveillance, Structural information, Spatial features

Laboratoire de Recherche de l'EPITA
14-16, rue Voltaire – FR-94276 Le Kremlin-Bicêtre CEDEX – France
Tél. +33 1 53 14 59 22 – Fax. +33 1 53 14 59 13
romain.trefault@epita.fr – http://www.lre.epita.fr/

Copying this document

Contents

Chapter 1

Introduction

Crowd counting is a technique used in various domains to estimate the number of individuals present in images or videos. With the increasing need for efficient crowd management and surveillance, accurate crowd counting has become a significant area of research. Imagine a crowded event or a busy public space captured in an image—estimating the number of people in such scenarios manually can be a daunting task for humans due to the sheer volume of individuals and complexities such as occlusions and variations in scale and perspective.

Over the years, various techniques have been developed to address the challenges of crowd counting. From traditional methods relying on basic machine learning and computer vision algorithms such as detection, regression, and density-based approaches, to CNN-based approaches showing significant promise in overcoming the limitations of traditional methods by effectively capturing spatial features and learning complex representations.

This technical report aims to explore the integration of Convolutional Neural Networks (CNNs) with Graph Neural Networks (GNNs) for dense crowd counting. By combining the strengths of both architectures, we seek to address the challenges associated with crowd counting and density estimation. Specifically, we propose a framework that combines a CNN for spatial feature extraction with a GNN for processing structural information within the crowd. The key contributions of this work are:

- A hybrid model architecture that integrates CNNs and GNNs for crowd counting, leveraging their respective strengths in capturing local features and global structural patterns.
- A novel graph construction method that transforms crowd scene representations into graphs, enabling the GNN to process structural information effectively.
- Experiments on the *JHU-CROWD++ dataset*, where our approach was tested under various crowd densities and environmental conditions.

The report is structured as follows: Firstly, we will provide an overview of the context and the state of the art in crowd counting techniques. Following this, we will detail our proposed approach, which includes the integration of CNNs and GNNs, along with the design principles behind our model architecture. Subsequently, we will outline the experimental setup, encompassing datasets, training procedures, and evaluation metrics. Next, we will present both the quantitative and qualitative results obtained from our experiments. Additionally, we will delve deeper into the implications of our findings and explore potential future directions. Finally, we will conclude the report by summarizing the key points and contributions.

Chapter 2

Context and State of the Art

Static crowd counting and estimation involve determining the number of people in a crowd using images. This task is important in several areas such as public safety, urban planning, and event management. Accurate crowd estimation is essential for ensuring safety at large events, efficiently managing resources, and making informed decisions.

2.1 Challenges in Crowd Counting

In crowd counting, several challenges arise:

- **Occlusion**: Individuals within a crowd often obstruct each other's visibility, making accurate counting difficult.
- **Scale Variation**: People may appear at different sizes in images due to their varying distances from the camera.
- **Density Variation**: Crowds can exhibit a wide range of densities, from sparse to highly congested areas.
- **Perspective Distortion**: Camera angles can distort the perceived size and shape of individuals, impacting counting accuracy.

2.2 State of the Art in Crowd Counting

Recent advancements in machine learning, especially in deep learning, have enhanced the accuracy and resilience of crowd counting systems. Here are the key approaches and techniques that have been employed and are currently in use:

2.2.1 Methods that Rely on Detection

Counting by detection can be classified into three types based on the features used to identify the crowd in images:

- **Monolithic Detection**: This approach trains a classifier using the full-body appearance available in the training images, using typical features such as Haar wavelets and gradient-based features like the histogram of oriented gradients (HOG). Learning methods such as

Support Vector Machines (SVMs) and Random Forests are employed using a sliding window approach. Nevertheless, these methods are limited to sparse crowds. For dense crowds, part-based detection is often more effective.

- **Part-based Detection**: Instead of considering the whole human body, this technique focuses on parts, such as the head or shoulders, and applies a classifier to them. While using the head alone is not sufficient for reliable person detection, the combination of head and shoulders is preferred in this technique.

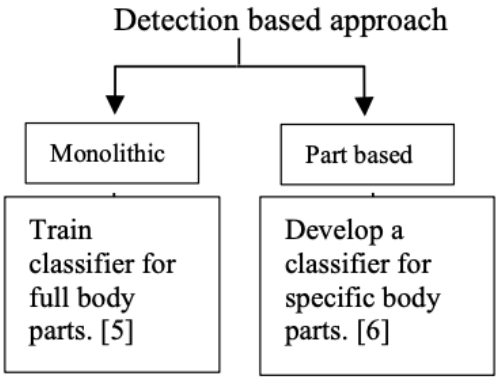

Figure 2.1: Representation of the two main detection based approaches. Source: [1]

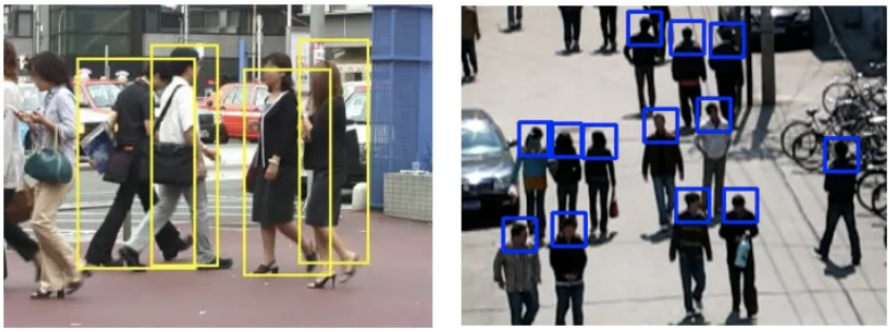

Figure 2.2: Example of the two main detection based approaches. Source: [2]

Below are two figures. The first one (Figure 2.1) illustrates the two methods of detection. The second one (Figure 2.2) displays the two images corresponding to crowd counting by detection. The left image represents monolithic detection, while the right image represents part-based detection.

2.2.2 Methods that Rely on Regression

Counting by detection methods often struggle with accuracy in dense crowds and high background clutter. To address these challenges, counting by regression techniques have been employed. These methods involve mapping features extracted from local image patches directly to the count, without the need for segmentation or individual tracking.

One approach involves extracting low-level features such as edge details and foreground pixels, and applying regression modeling to map these features to the count. For instance, Chan et al. [1] proposed a Bayesian regression-based method to estimate crowd sizes on subway platforms, where the backdrop is eliminated and various foreground pixels are measured. These pixels, such as total area or texture, are then used as inputs to the regression function. This approach bypasses intermediate vision operations like object identification or feature tracking, providing a direct estimation of crowd size based on low-level features.

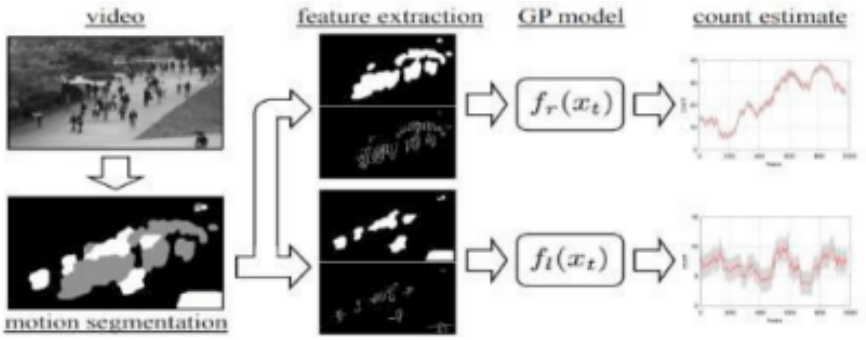

Figure 2.3: Illustration of the Bayesian regression-based method. Source: [3]

2.2.3 Methods that Rely on Density Estimation

Another approach in crowd counting involves estimating density rather than focusing solely on individual counts, thereby incorporating crucial spatial information from images. By learning the mapping between local features and object density maps, this approach tracks groups of individuals simultaneously, rather than learning each individual separately. The mapping can be linear or nonlinear, with the latter demonstrated through techniques such as a Random Forest Classifier.

One such method using a random forest regressor is described in the paper of Viet-Quoc Pham et al. [2]. Here, the regressor votes for densities of multiple target objects, learning a nonlinear mapping between patch features and the relative locations of objects within the patch. Additionally, a crowdedness prior parameter is defined to differentiate to help the system decide whether a particular area of the picture is crowded with objects or not. Based on this parameter, they create two separate sets of guesses (or *forests*), one for crowded areas and one for less crowded areas. This way, the system can handle different scenarios appropriately.

Density estimation approaches offer significant advantages over traditional counting methods, particularly in addressing occlusion and clutter issues. These methods provide not only a count but also a spatial distribution of individuals, as depicted in Figure 2.4 showing the density map for an image with ground truth and prediction. A summary of conventional crowd counting methods is provided in Table 2.1.

2.2.4 Methods that Rely on Deep Learning

Deep learning methods have garnered interest from researchers worldwide, particularly in the field of image processing, where Convolutional Neural Networks (CNNs) have demonstrated remarkable learning capabilities. This has led to a surge in CNN-based crowd counting projects. While earlier applications of CNNs focused solely on computing the density of people without

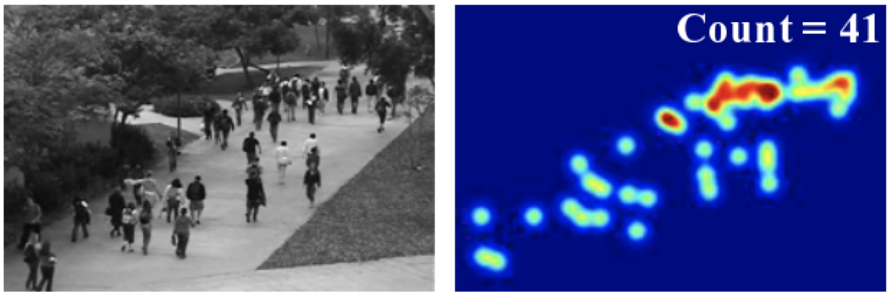

Figure 2.4: Estimation of the object density map and the number of objects (right) from an input image (left). Source: [4]

Table 2.1: Comparison of Traditional Crowd Counting Approaches

Traditional Approach	**Mechanism**	**Pros & Cons**
Detection-based	Utilizes sliding window for body part detection.	+ High accuracy for sparse crowds
		- Not suitable for dense crowds due to computational limitations
Regression-based	Employs feature extractor and regression function.	+ Provides better accuracy overall
		- May overlook important spatial details in favor of a generalized count
Density estimation	Utilizes density maps to estimate crowd density.	+ Incorporates spatial information for enhanced accuracy
		- Requires robust preprocessing and parameter tuning

considering the actual count, recent advancements have significantly improved CNN-based crowd counting models. These modern approaches excel in handling diverse challenges such as varying head scales, non-uniform density distributions, and changes in perspective and scene, making them the predominant choice in current crowd counting research. In contrast to traditional approaches, contemporary computer vision techniques based on CNNs are achieving superior accuracy. A plethora of CNN architectures has been devised specifically for crowd density estimation, categorized into distinct groups for clarity:

- **Basic CNNs**: These encompass the initial deep learning methodologies, characterized by fundamental convolutional layers, kernels, and pooling layers.
- **Scale-aware models**: These advanced CNN architectures incorporate multi-column or multi-resolution designs to enhance robustness.
- **Context-aware models**: These models integrate both local and global contextual information into CNN frameworks.
- **Multi-task frameworks**: In addition to crowd counting, these frameworks tackle auxiliary tasks such as crowd velocity estimation and foreground-background subtraction.

Wang et al. [3] introduced one of the pioneering approaches utilizing a CNN regression model. They used an AlexNet as the base neural network, replacing the final layer of 4096 neurons with a single neuron to estimate crowd count. Additionally, the training data was augmented with samples whose ground truth count was zero. Below (Figure 2.5) is a depiction of the five-layer convolutional architecture initially employed for crowd identification. Zhang et al. [4]

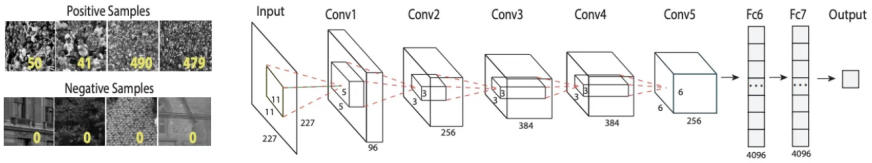

Figure 2.5: The Five-Layer Convolutional Architecture for Crowd Identification in static images. Source: [5]

presented a Multi-Column CNN architecture mapping images to their respective crowd density maps. This model utilizes filters with varying receptive fields, enabling each column CNN to adaptively learn features considering variations in people/head size due to perspective effects or image resolution. The density map is accurately computed based on geometry-adaptive kernels.

Boominathan et al. [5] proposed a hybrid approach combining deep and shallow fully convolutional neural networks. This strategy aims to capture both low-level and high-level features for crowd counting. Their deep network, akin to the renowned VGG-16 architecture, captures high-level semantics essential for crowd counting and density map generation. Additionally, they developed a shallow network tailored for identifying low-level head blob patterns of people distant from the camera. The representation of this approach is depicted in Figure 2.6. Oñoro-Rubio et al. [6] present a significant contribution comprising two neural networks.

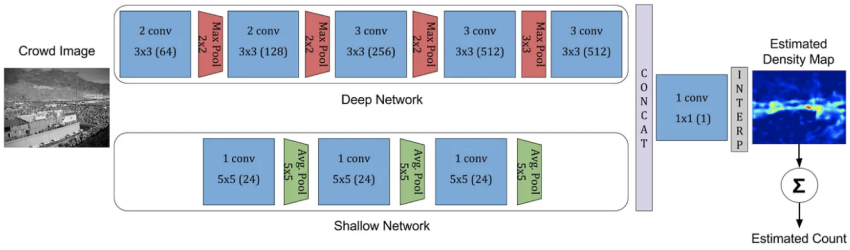

Figure 2.6: Representation of the Deep and Shallow Network. Source: [6]

Firstly, a Counting CNN serves as a regression model learning to map appearance features of image patches to corresponding density maps. Secondly, a Hydra CNN is introduced as a scale-aware counting model, utilizing image patches extracted at multiple scales to estimate the final density. The architecture illustrating both networks is depicted in Figure 2.7.

The advancements in crowd counting methodologies, ranging from traditional detection and regression techniques to modern density estimation and deep learning approaches, have significantly enhanced the ability to accurately estimate crowd sizes under various conditions. Despite the progress achieved with CNNs, challenges such as varying head scales, non-uniform

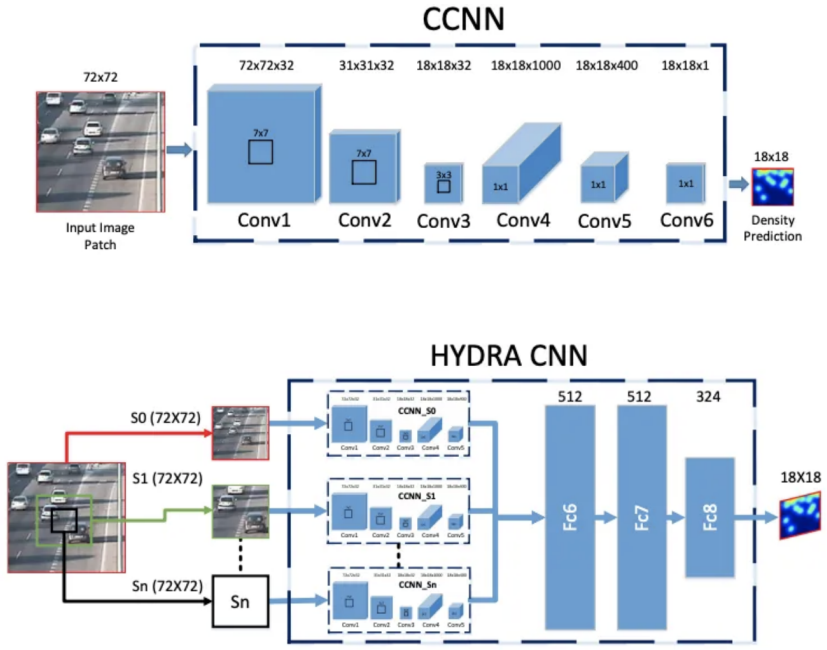

Figure 2.7: Architecture Illustrating Counting and Hydra Convolutional Neural Networks. Source: [7]

density distributions, and scene variations persist. To further address these challenges, this report introduces a new approach that combines CNNs with GNNs. The subsequent chapters will delve into my contributions, detailing the development and integration of this hybrid model for crowd counting in static images.

Chapter 3

Proposed Approach to Integrate CNNs and GNNs for Crowd Counting

In our work, we propose a novel approach that combines Convolutional Neural Networks (CNNs) with Graph Neural Networks (GNNs) to quantify crowds. This unique combination, not yet explored in the realm of crowd analysis, harnesses the respective strengths of both architectures to create a robust and high-performing framework for crowd counting tasks.

Our approach begins with a CNN to extract dense spatial features from crowd images. CNNs are adept at capturing detailed visual features, making them ideal for identifying individual people and areas within an image. Specifically, the CNN's goal is to predict rectangles containing people's faces, identifying individuals within a crowd. These predicted rectangles are then used to build a graph where each node represents a specific predicted face in the image, and the edges capture the spatial relationships between these faces. This graph structure is processed by a GNN, which excels at understanding higher-order interactions and structural dependencies between different regions. This two-stage process allows our model to utilize local feature information and global structural patterns for our crowd counting task.

3.1 Description of CNN and GNN Usage

3.1.1 CNN for Spatial Feature Extraction

U-Net

In our crowd analysis framework, we adopted the U-Net architecture for spatial feature extraction. Initially designed for biomedical image segmentation, the U-Net offers an encoder-decoder structure complemented by skip connections, facilitating seamless integration of both local and global context. The UNet architecture, known for its strong performance in segmentation tasks, is particularly well-suited for identifying and localizing individual people within a crowd.

Let I represent the input image, where I is a matrix with dimensions $W \times H \times C$. Here, W and H denote the width and height of the image, respectively, while C represents the number of channels, RGB channels in our case.

1. **Input Processing**: Crowd images are fed into the U-Net, which consists of an encoder chapter followed by a decoder chapter.

2. **Encoder Convolutional Blocks**: The encoder consists of multiple convolutional blocks, each extracting hierarchical features from the input image. These blocks typically include convolutional layers followed by activation functions such as ReLU and pooling layers for downsampling. Let I represent the input image, F_i denote the i-th filter, and b_i the corresponding bias term. The output of the i-th convolutional layer is computed as:

$$O_i = \text{ReLU}(I * F_i + b_i)$$

where $*$ denotes the convolution operation.

Activation Functions: Non-linear activation functions, such as ReLU, introduce non-linearity into the model, enabling it to capture complex spatial patterns.

Batch Normalization: Applied after each convolutional layer, batch normalization stabilizes and accelerates training by normalizing activations. Given the input x to the batch normalization layer, the output y is computed as:

$$y = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}} \times \gamma + \beta$$

where μ and σ^2 are the mean and variance of x, γ and β are learned parameters, and ϵ is a small constant to prevent division by zero.

3. **Decoder Convolutional Blocks**: The decoder chapter reconstructs the spatial information from the encoded features, gradually upsampling them to the original image resolution. Skip connections from the encoder to the decoder help preserve fine-grained details.

4. **Output Feature Map**: The U-Net model generates a feature map showing key details about the crowd in the image. This map identifies and distinguishes individual people or groups. It contains rectangles that indicate the locations of faces within the crowd.

3.1.2 GNN for Structural Information Processing

A graph G consists of a set of nodes V and a set of edges E. We represent graphs as $G = (V, X, A)$, where $X \in \mathbb{R}^{n \times d}$ is a matrix of node features with n nodes and d dimensions, and $A \in \{0, 1\}^{n \times n}$ is the adjacency matrix representing the connections between nodes. Graph Neural Networks (GNNs) leverage message-passing techniques to learn node representations and perform various tasks on graphs, such as node classification, graph classification, link prediction, or clustering.

In our crowd counting approach, the Graph Neural Network (GNN) plays a crucial role in processing the graph structure derived from the embeddings extracted from the U-Net bottleneck. Within this graph, each node represents a predicted face, and edges signify the proximity between these faces within the crowd. The representation of each node $u \in V$ undergoes refinement through message-passing and aggregation of information from its neighboring nodes. This iterative update process spans multiple layers of the GNN, enabling nodes to accumulate insights from their multi-hop neighbors and generate structure-aware representations.

We define $h_i^{(l)}$ as the representation of node i at iteration (layer) l, and $\texttt{AGGR}(.,.)$ as the aggregation method. The update rule for node representations can be expressed as follows:

$$h_u^{(l)} = \texttt{AGGR}(h_u^{(l-1)}, \{h_i^{(l-1)} | i \in N(u)\})$$

Here, $h_u^{(l-1)}$ represents the previous layer's representation of node u, and $\{h_i^{(l-1)} | i \in N(u)\}$ denotes the set of representations of node u's neighbors, where $N(u)$ is the set of neighbors of node u. The aggregation function combines these representations to compute the updated representation $h_u^{(l)}$ for node u at layer l.

In our GNN model, we implement this message-passing mechanism using graph convolutional layers. The architecture of the GNN is designed to iteratively refine node embeddings by aggregating information from neighboring nodes. The process is detailed as follows:

1. **Graph Construction**: We construct a graph where nodes correspond to the predicted rectangles (faces) in the image, while edges denote spatial relationships (proximity) between these faces.

2. **Graph Convolutional Layers**: The GNN processes the graph using graph convolutional layers, which iteratively aggregate and transform the features of each node based on its neighbors' features.

3. **Higher-Order Interactions**: Stacking multiple graph convolutional layers enables the GNN to capture higher-order interactions and dependencies, facilitating the learning of complex structural patterns essential for accurate crowd counting.

4. **Global Mean Pooling**: Following the convolutional layers, a global mean pooling operation is applied. This operation aggregates node features, generating a comprehensive graph-level representation that summarizes information from all nodes.

5. **Fully Connected Layer**: The pooled representation is fed into a fully connected layer, which maps the hidden features to the final output dimension. This layer produces the ultimate crowd count estimate for the input image.

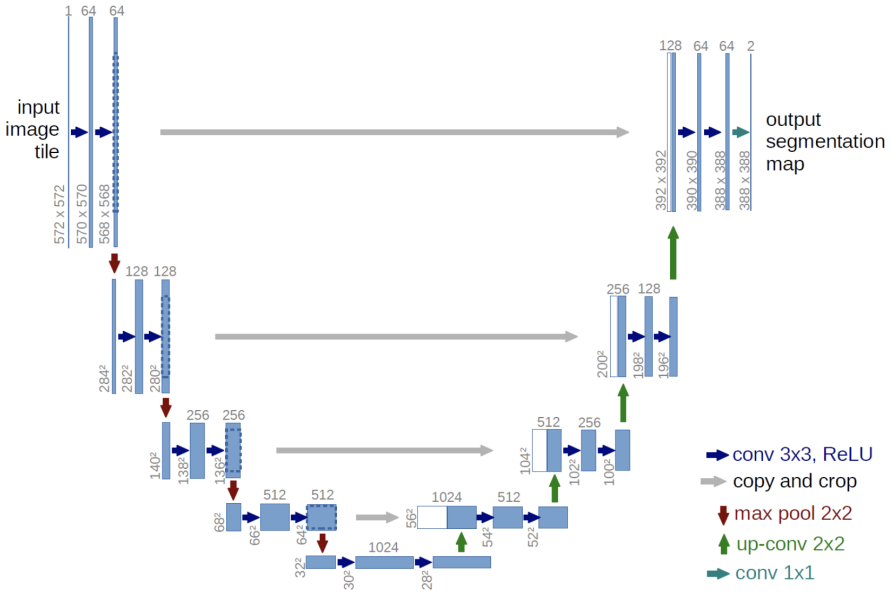

Figure 3.1: UNet Architecture. Source: [8]

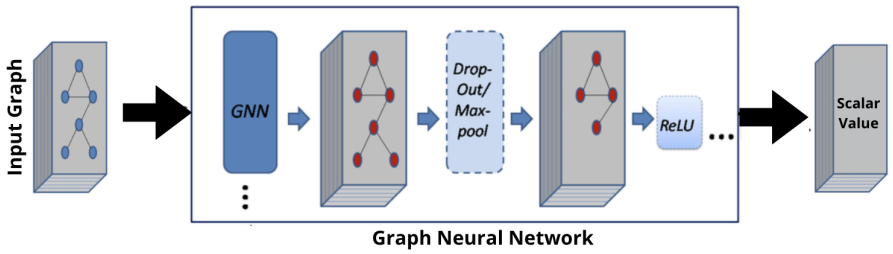

Figure 3.2: GNN Architecture.

3.2 Combined Model Architecture and Design Rationale

1. **UNet Backbone**: The UNet architecture serves as the backbone for our model, particularly its downsampling component, which is critical for extracting embeddings from the bottleneck layer corresponding to the predicted rectangles.

2. **Embedding Extraction**: The embeddings from the UNet's bottleneck layer represent predicted faces. By knowing the number of pooling layers and coordinates of each face, we can determine their scaled coordinates within the bottleneck. These embeddings provide the feature representations for each predicted face.

3. **Feature Map to Graph Conversion**: Using the embeddings from the UNet's bottleneck, we construct a graph where each node represents a face embedding of shape $1 \times 1 \times 1024$. Edges between nodes are established based on the spatial proximity of the face coordinates. Specifically, we compute the average distance between centroids of rectangles. Let uss denote the centroid of a rectangle as $C_i = (x_i, y_i)$, where x_i and y_i are the coordinates of the centroid of rectangle i. The distance between two centroids C_i and C_j is given by the Euclidean distance formula:

$$d_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

The average distance between centroids of all pairs of rectangles is calculated by summing up all such distances and dividing by the total number of pairs of rectangles, which is $\frac{n(n-1)}{2}$, where n is the number of rectangles.

So, mathematically, the average distance D between centroids can be expressed as:

$$D = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d_{ij}}{\frac{n(n-1)}{2}}$$

$$D = \frac{\sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}}{\frac{n(n-1)}{2}}$$

Where:

- n is the number of predicted rectangles.
- (x_i, y_i) represents the centroid of the i-th rectangle.

- d_{ij} represents the distance between the centroids of the i-th and j-th rectangles.

This average distance captures the spatial relationships between faces in the crowd. If the distance between any two centroids is less than or equal to this average, an edge is created, reflecting the proximity of the corresponding rectangles.

4. **Graph Neural Network**: The GNN component processes this constructed graph using multiple graph convolutional layers. These layers aggregate information from neighboring nodes, iteratively refining the feature representation to capture higher-order structural patterns within the crowd.

5. **Integration and Output**: After processing through the GNN, we get a final scalar value representing the crowd count estimate.

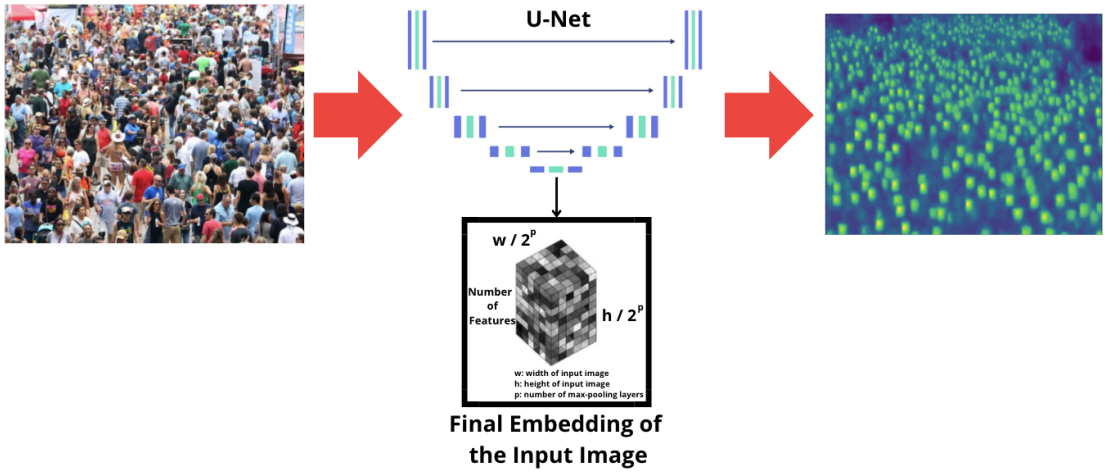

Figure 3.3: CNN Component of the Framework.

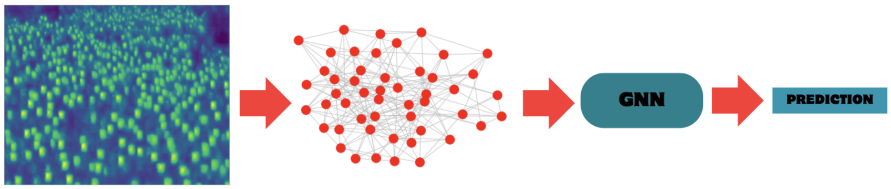

Figure 3.4: GNN Component of the Framework.

The rationale behind this architecture is to leverage the complementary strengths of UNet and Graph Neural Networks (GNNs). UNet excels at capturing fine-grained spatial features, while GNNs are adept at understanding complex structural relationships. By transforming the feature embeddings into a graph and utilizing a GNN to process this graph, the model integrates both local feature details and global structural patterns. This synergy allows for dynamic refinement of face embeddings, where the graph structure and GNN facilitate better differentiation of overly similar embeddings and merging of overly dissimilar ones, based on their spatial relationships and feature similarities.

Chapter 4

Experiments

4.1 Description of Datasets Used for Evaluation

To evaluate our proposed model, we used a widely recognized dataset in the field being known for its diversity and comprehensive annotations, *JHU-CROWD++* dataset.

- ***JHU-CROWD++* Dataset**: This dataset contains a large number of images with varying crowd densities, ranging from sparse to extremely dense crowds. The dataset is challenging due to the presence of occlusions, varying perspectives, and diverse environmental conditions. Each image is annotated with the precise location of each person, providing reliable ground truth for both training and evaluation. The dataset includes various subsets that allow for extensive evaluation across different types of scenes and crowd densities.

The *JHU-CROWD++* dataset was selected for its rich variety and detailed annotations, ensuring that our model is tested across different scenarios, including both low and high-density crowds, as well as diverse environmental settings.

4.2 Details of Training Procedures and Hyperparameters

The training procedures and hyperparameters are essential components of our model's development, ensuring optimal performance for crowd counting.

- **Training Procedure**:

  Data Preprocessing: Due to variations in size among images in the *JHU-CROWD++* dataset, resizing to (480×384) was performed to standardize dimensions, ensuring consistency in input size for the UNet, and accelerating training by simplifying the computational workload and memory requirements. Additionally, normalization of pixel values was conducted to scale the intensity range between 0 and 1, facilitating convergence during model training and enhancing its generalization capabilities.

  Training Split: The dataset is divided into three subsets: training, validation, and test sets. The training set, which constitutes 80% of the data, is used for model training. The validation set, comprising 10% of the data, aids in hyperparameter tuning and mitigating overfitting. The remaining 10% forms the test set, which is employed to assess the final

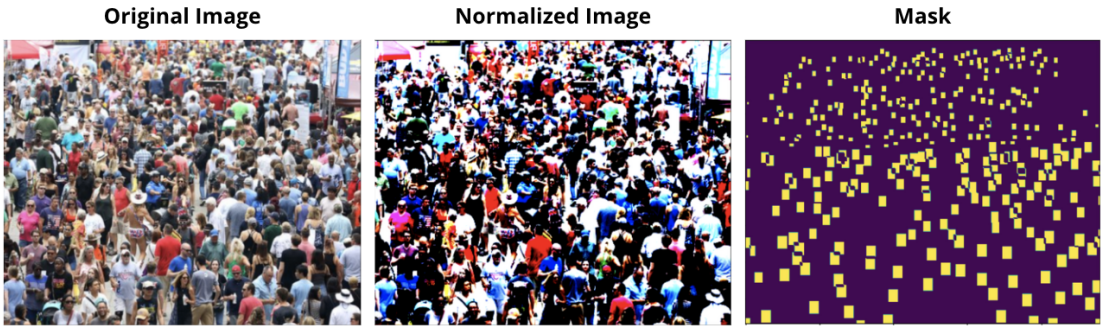

Figure 4.1: A Sample from Custom Dataset.

model performance. For our dataset, this distribution translates to 2272 samples in the training set, 500 samples in the validation set, and 1600 samples in the test set.

The UNet model was trained using original images as inputs and corresponding masks as targets, enabling it to learn to accurately segment objects within the images. Subsequently, the Graph Neural Network (GNN) was trained using graphs derived from the UNet's predictions, along with the ground truth data specifying the number of people in the crowd as targets.

- **General Hyperparameters**:

  Learning Rate: The initial learning rate is set to 1×10^{-3}, with a learning rate scheduler that reduces the rate by a factor of 0.1. This factor in the learning rate scheduler determines the rate at which the learning rate is decreased when the validation loss stagnates for a certain number of epochs.

  Batch Size: A batch size of 8 is employed, mitigating potential computational efficiency issues.

  Optimizer: The Adam optimizer is employed to minimize the loss function, with default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

- **UNet-specific Hyperparameters**:

  Training Duration: The model underwent 500 epochs of training with early stopping implemented, which halts the training process if the validation loss fails to improve for 25 consecutive epochs.

  Loss Function: The loss function is a combination of Binary Cross-Entropy Loss and Dice Loss, computed as:

$$\text{Loss} = \text{Binary Cross-Entropy Loss} + \text{Dice Loss}$$

  where the Binary Cross-Entropy Loss measures the dissimilarity between the predicted and ground truth masks. It is calculated as:

$$\text{Binary Cross-Entropy Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(p_i) + (1 - y_i)\log(1 - p_i)]$$

where N is the number of samples, y_i represents the ground truth label (0 or 1), and p_i is the predicted probability of the positive class (between 0 and 1). This loss function penalizes deviations from the ground truth labels, encouraging the model to produce accurate predictions. The Dice Loss is defined as:

$$\text{Dice Loss} = 1 - \text{Dice Coefficient}$$

The Dice Coefficient measures the similarity between two samples and is defined as:

$$\text{Dice Coefficient} = \frac{2 \times \text{Intersection} + \epsilon}{\text{Summation} + \epsilon}$$

where ϵ is a small constant (typically 1×10^{-6}) added to avoid division by zero, the Intersection represents the overlap between predicted and ground truth masks, and the Summation is the sum of the pixels in both masks. This loss function penalizes deviations from the intersection over union (IoU) metric, encouraging the model to produce accurate and spatially coherent segmentation masks.

- **GNN-specific Hyperparameters**:

  Training Duration: The model underwent 200 epochs of training.

  Input Dimension: The input dimension aligns with the dimensionality of embeddings extracted from the UNet ($1 \times 1 \times 1024$).

  Hidden Dimension: Set at 64, the hidden dimension for GNN layers enables the model to capture complex representations.

  Number of Layers: The GNN architecture comprises 2 graph convolutional layers, facilitating the capture of higher-order interactions between nodes.

  Loss Function: The Mean Squared Error (MSE) loss function is employed to quantify the disparity between predicted and true values, helping the model in generating accurate predictions. This metric computes the average of the squares of the errors between predicted and actual counts, offering insight into the variability of these errors.

$$\text{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Here, y_i denotes the true value, $\hat{y}_i$ represents the predicted value, and N is the total number of samples. This loss function guides the model to minimize the average squared differences between predictions and ground truth, enhancing its predictive accuracy.

4.3 Evaluation Metrics Used for Assessing Crowd Counting Performance

To comprehensively evaluate the performance of our crowd counting model, we used 2 standard metrics tailored to different components of our model:

4.3.1 CNN

- **Dice Coefficient**: This metric was used for the UNet component to evaluate the overlap between the predicted rectangles and the ground truth. It provides a measure of spatial

accuracy.

$$\text{Dice} = \frac{2\sum_{i=1}^{N} y_i \hat{y}_i}{\sum_{i=1}^{N} y_i + \sum_{i=1}^{N} \hat{y}_i}$$

The Dice coefficient ranges from 0 to 1, with 1 indicating perfect overlap.

4.3.2 GNN

- **Mean Absolute Error (MAE)**: This metric measures the average absolute difference between the predicted and actual crowd counts, providing a direct measure of the model's accuracy.

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

where y_i is the ground truth count, $\hat{y}_i$ is the predicted count, and N is the number of test samples.

These metrics offer a comprehensive evaluation of the model's performance, illuminating its accuracy and robustness across various crowd counting scenarios. On one hand, BCE and Dice Coefficient metrics provide insights into the spatial accuracy of the UNet's predictions, capturing nuances in segmentation performance. On the other hand, MAE and MSE metrics offer a balanced perspective on the GNN's efficacy in crowd count estimation, shedding light on the model's ability to minimize errors and provide reliable count estimates.

Chapter 5

Results

5.1 Quantitative and qualitative results obtained from experiments.

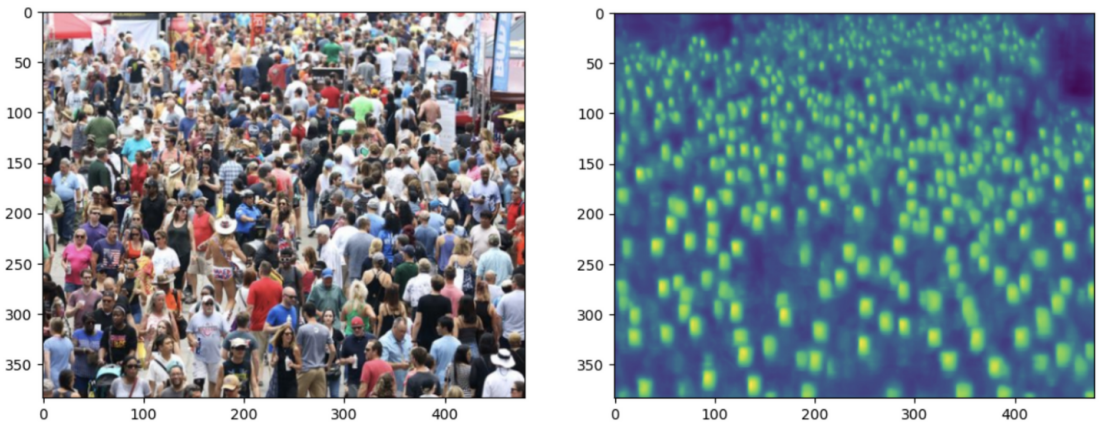

Figure 5.1: CNN Prediction Example.

In Figure 5.1, we present an illustrative example of one of our Convolutional Neural Network (CNN) prediction. The image on the left captures a crowded scene depicting 359 individuals gathered together. On the right, we showcase the CNN's segmentation of the crowd scene, wherein each person is outlined by a rectangle. Notably, some rectangles may not perfectly conform to the shape of individuals, as observed in the bottom-left corner. This deviation occurs as the network attempts to predict non-overlapping rectangles.

Figure 5.2 illustrates various post-processing techniques implemented on our segmentation outcomes. In the top row, binary representations stem from the CNN prediction map using a 0.5 threshold. Below, the original crowd image is juxtaposed with identifying red points denoting individual instances based on their prediction maps.

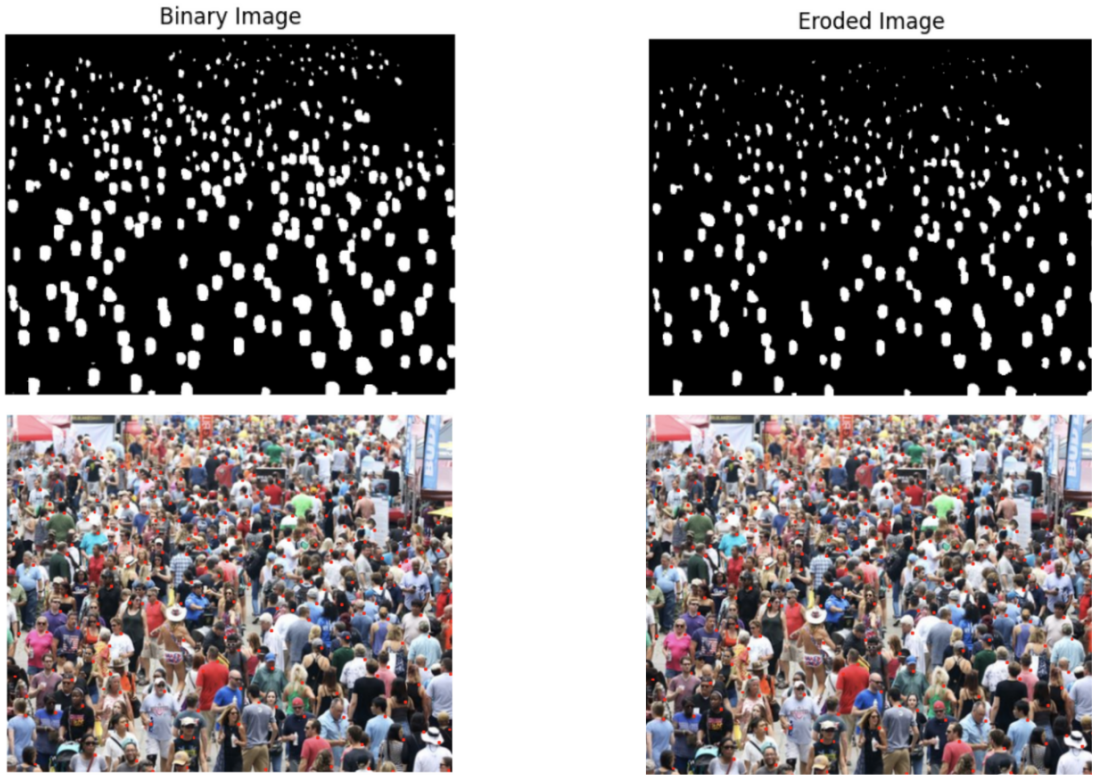

Figure 5.2: Post-Processing Examples.

In the right column, an additional refinement step involves erosion.

First, the output matrix O is converted into a binary image B using a threshold value, defined as:

$$B_{ij} = \begin{cases} 1 & \text{if } O_{ij} > \text{threshold} \\ 0 & \text{otherwise} \end{cases}$$

Erosion is then applied to B using a 3×3 kernel K, resulting in an eroded binary image E. This morphological operation shrinks the boundaries of objects, defined as:

$$E_{ij} = \min_{(k,l) \in K} B_{i+k,j+l}$$

This step refines instance segmentation by reducing the size of objects, contributing to more precise delineation. However, erosion can cause very small predicted rectangles to disappear.

The application of erosion decreased the number of connected components from 291 to 259. This reduction is notable, especially given that the target was 359 connected components.

Below, an image of the graph generated from the embeddings of the UNet bottleneck is presented in Figure 5.3. This graph illustrates the relationships and connections between different features extracted during the segmentation process. By leveraging the embeddings from the UNet bottleneck, we can visualize the inherent structure and patterns within the segmented data. Analyzing this graph provides insights into the underlying representation of features,

aiding in understanding the segmentation process and potentially informing further refinement strategies.

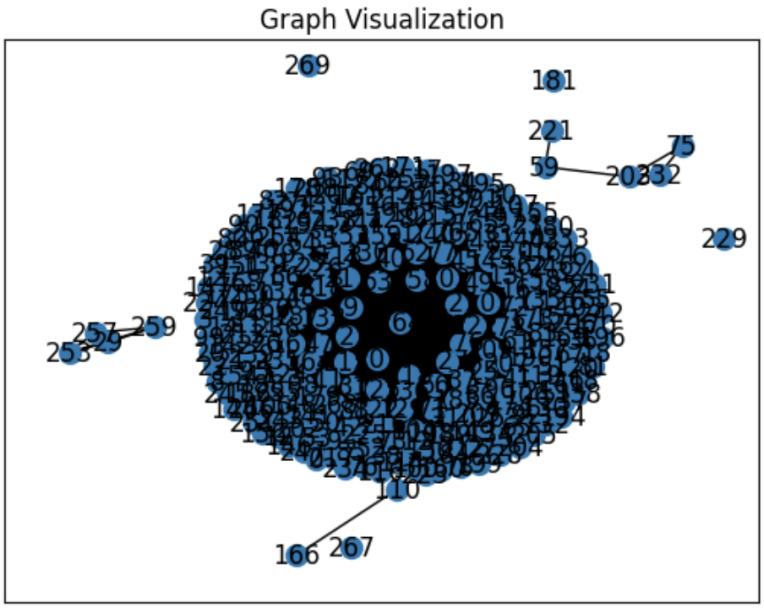

Figure 5.3: Graph Visualization Example.

We can observe that, in this case, there are 5 connected components.

5.2 Evaluation Phase

```
Number of connected components w/o erosion (CNN):  tensor([ 73, 396,  44,  31, 225, 120,  10,  61])
Number of connected components w erosion (CNN):  tensor([ 75, 294,  34,  21, 227,  93,   4,  54])
Actual output:  tensor([  92., 2552.,  106.,  125.,  278.,  113.,  154.,  106.])
Predicted output:  tensor([ 122.3437, 1127.0400,  157.4355, 1019.1498,  382.2008,  483.1833,
        2416.0127,  569.6404])
```

Figure 5.4: GNN Predictions.

Figure 5.4 displays the results from testing the GNN model for predicting the number of people's faces in a given input graph. The first row shows the ground truth, representing the actual number of connected components without erosion on the CNN feature map output. The second row displays the ground truth number of connected components with erosion applied to the CNN output. The *Actual output* row presents the actual ground truth values (i.e., the number of people in the crowd image) for the corresponding inputs. The *Predicted output* row presents the predictions made by the GNN model for these inputs. Notably, the predictions and actual outputs are presented in batches of eight.

It is evident that the results did not meet our expectations, as there is a significant discrepancy between the predicted values and the actual outputs. In the next section, we will delve into potential reasons for the discrepancies, explore possible avenues for improvement, and provide insights gained from evaluating the performance of the GNN model.

Chapter 6

Discussion

This report introduces a novel approach that merges CNN and GNN networks to quantify the number of people in a crowd. Nonetheless, our experiments have yielded unsatisfactory results, which deviate from the current state-of-the-art scores.

One potential challenge may arise from the use of erosion in post-processing, as it can inadvertently remove too-small components, leading to a loss of accuracy in predictions. Additionally, the rectangular shape of predictions may not accurately represent the true oval shapes of faces, potentially impacting the performance of our approach.

Moreover, the resizing of original images could result in shape deformation within the crowd, further complicating the segmentation process and potentially affecting the accuracy of our predictions. The distortion of faces' original shapes caused by image resizing may also hinder the CNN's performance, introducing inconsistencies in segmentation results and affecting GNN interpretation.

Future research could explore alternative post-processing techniques that better preserve the original shapes, investigate methods to handle the ovular or round shape of faces within crowds more accurately, and develop strategies to mitigate the impact of image resizing on shape deformation.

Furthermore, it would be beneficial to examine the impact of hyperparameters in models, or kernel size in our erosion refinement step, impacting the performance of our method. Understanding these parameters' effects can provide insights into enhancing the robustness and effectiveness of our approach in quantifying dense crowds.

Chapter 7

Conclusion

In this technical report, we have presented a new approach that combines Convolutional Neural Networks (CNNs) and Graph Neural Networks (GNNs) for dense crowd counting. By leveraging the strengths of both architectures, our model aims to capture both local spatial features and global structural patterns within crowd scenes, offering a fresh perspective on tackling the challenge of crowd counting.

Our approach uses a CNN backbone to extract spatial features from crowd images, followed by the construction of a graph where nodes represent predicted faces and edges capture their spatial relationships. This graph is then processed by a GNN, which refines the node embeddings by aggregating information from neighboring nodes, capturing higher-order interactions and structural dependencies.

In experiments conducted on the challenging JHU-CROWD++ dataset, we sought to evaluate the efficacy of our approach in managing diverse crowd densities and environmental conditions. Despite our efforts, our results did not surpass existing benchmarks, indicating that our method did not demonstrate superior effectiveness in this context.

Future research directions could explore alternative post-processing strategies, investigate methods to handle the ovular or round shape of faces within crowds more accurately, and develop techniques to mitigate the impact of image resizing on shape deformation. Additionally, conducting a comprehensive analysis of hyperparameter sensitivity and its impact on model performance could offer valuable insights, ultimately leading to the refinement and enhancement of our model's robustness and effectiveness in real-world applications.

In conclusion, this work represents a step towards integrating CNNs and GNNs for crowd analysis tasks, paving the way for more advanced techniques that can leverage the complementary strengths of these architectures. By addressing the limitations identified in this report, future research could unlock even greater potential for accurate and reliable crowd counting, fostering advancements in various real-world applications, including crowd management, urban planning, and surveillance systems.

References

1. Chan, A. B., & Vasconcelos, N. (2008). Counting People With Low-Level Features and Bayesian Regression. *IEEE Transactions on Image Processing*.

2. Pham, V.-Q., Kozakaya, T., Yamaguchi, O., & Okada, R. (2015). COUNT Forest: CO-Voting Uncertain Number of Targets Using Random Forest for Crowd Density Estimation.

3. Wang, C., Zhang, H., Yang, L., Liu, S., & Cao, X. (2018). Deep People Counting in Extremely Dense Crowds. *State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences*.

4. Zhang, Y., Zhou, D., & Ma, Y. (2016). Single-Image Crowd Counting via Multi-Column Convolutional Neural Network. Published in *Computer Vision and Pattern Recognition*.

5. Boominathan, L., Kruthiventi, S. S. S., & Babu, R. V. (2016). CrowdNet: A Deep Convolutional Network for Dense Crowd Counting. *Video Analytics Lab, Indian Institute of Science, Bangalore, India*.

6. Oñoro-Rubio, D., & López-Sastre, R. J. (2016). Towards Perspective-Free Object Counting with Deep Learning.

7. Han, K., Wang, Y., Guo, J., Tang, Y., Wu, E. (2022). Vision GNN: An Image is Worth Graph of Nodes.

8. Lu, Y., Chen, Y., Zhao, D., Liu, B., Lai, Z., & Chen, J. (2020). CNN-G: Convolutional Neural Network Combined with Graph for Image Segmentation with Theoretical Analysis.

9. Luo, A., Yang, F., Li, X., Nie, D., Jiao, Z., Zhou, S., Cheng, H. (2020). Hybrid Graph Neural Networks for Crowd Counting.

10. Peng, F., Lu, W., Tan, W., Qi, K., Zhang, X., & Zhu, Q. (2022). Multi-Output Network Combining GNN and CNN for Remote Sensing Scene Classification.

11. Advances in Convolution Neural Networks Based Crowd Counting and Density Estimation by Rafik Gouiaa, Moulay A. Akhloufi, and Mozhdeh Shahbazi (2022).

12. Patwal, A., Diwakar, M., Tripathi, V., & Singh, P. (2023). Crowd counting analysis using deep learning: a critical review. *Department of CSE, Graphic Era Deemed to be University, Dehradun, Uttarakhand, India*.

13. Sindagi, V. A., & Patel, V. M. (2017). A Survey of Recent Advances in CNN-based Single Image Crowd Counting and Density Estimation. *Dept. of Electrical and Computer Engineering, 94 Brett Road, Piscataway, NJ 08854, USA*.

14. Loy, C. C., Chen, K., Gong, S., & Xiang, T. (2013). Crowd Counting and Profiling: Methodology and Evaluation.

List of Figures