Computer Vision Models for Crowd Counting in Moose Jaw City

Stan Fedyk, Haiying Kuang, Steven Yang, Arjun Singh, Hanran Song, Jiachen Huang, Yining Song
Department of Computing Science
University of Alberta
{sfedyk, hkuang, dingjing, arjun10, hanran, jiache19, song17}@ualberta.ca

Abstract

The accuracy of crowd counting is critical for public safety, resource allocation, and urban planning. Compared to traditional sensor-based methods, modern deep learning approaches have achieved superior performance in terms of both accuracy and reduced human involvement. However, achieving strong generalization remains a significant challenge in real-world applications. In this research, we collaborated with the City of Moose Jaw to explore an effective and universal crowd-counting solution. We evaluated several models based on three common approaches: map-based, detection-based, and point-based methods. For each model, we trained on high-density data and tested it on low-density and unseen data. Our experiments show that MPCount [9] achieves the best balance between accuracy and robustness, exhibiting strong generalization capabilities from dense to sparse scenarios. These processes provide insight values for the deployment of crowd monitoring systems in small-scale urban cities.

1 Introduction

Crowd counting contributes to improving public safety, optimizing resource allocation, and facilitating urban management. With the rapid development of smart city systems, the demand for accurate and scalable crowd counting technologies is increasing accordingly. These technologies are gaining attention from researchers and government agencies who are looking for data-driven tools to enable effective decision making.

In response to these growing needs, crowd counting models are mainly applicable in the following aspects. By accurately monitoring the crowd density, managers can identify potential risks and take preventive actions, reducing the possibility of accidents such as stampedes. In addition, real-time crowd distribution enables staff to dynamically allocate resources and guide visitors away from congested areas, ultimately improving operational efficiency and enhancing the overall visitor experience.

Existing deep learning crowd counting models have achieved excellent performance on benchmark datasets such as the ShanghaiTech Dataset [17] and the UCF-QNRF [4]. However, these data represent highly congested or idealized scenes. However, crowd images in real word often show greater diversity, such as significant differences in the number of people, diverse camera perspectives, and scale differences from near to far objects. Although these models perform well on benchmark datasets, their generalization to real-world deployment environments remains underexplored.

To address this gap, we collaborated with the City of Moose Jaw to investigate the performance and deployability of several crowd-counting models in real-world surveillance conditions. The models are mainly trained on the high-density ShanghaiTech Part A dataset and evaluated on the low-density ShanghaiTech Part B dataset and Moose Jaw datasets. Our evaluation focuses on the models single-

domain generalization ability, accuracy, and consistency under different crowd densities and visual conditions.

(a) Source domain (b) Target image A (c) Target images B

Figure 1: **Domain generalization for crowd counting.** (a) A sample image from the ShanghaiTech Part A dataset with narrow high-density distribution. (b) A target image from the ShanghaiTech Part B dataset with relatively low-density distribution. (c) A target image from the MooseJaw dataset.

We summarize our contributions as follows:

- We introduce one of the crowd-counting evaluations conducted in collaboration with a small Canadian city using real surveillance footage.
- We benchmark multiple state-of-the-art models across datasets, highlighting their strengths and weaknesses in single-domain generalization.
- We identify MPCount [9] as a practical and highly effective solution for deployment in real-world urban environments.

2 Related Work

2.1 Mainstream Approaches for Crowd Counting

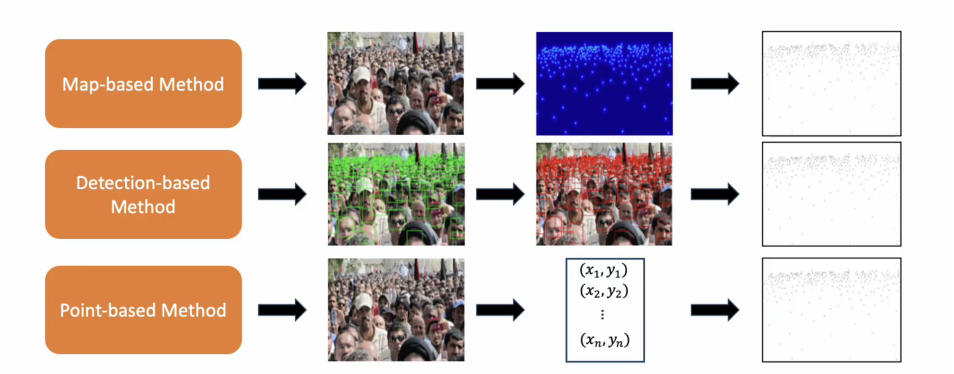


Figure 2: **Overview of the three mainstream crowd counting methods.**
Image Source: `https://www.youtube.com/watch?v=b_ltwfD9dLI&t=20s`

The core idea of the **Map-based Method** is to transform the crowd-counting task into a pixel-level density estimation problem. The model outputs a density map of the same or similar size as the input image. Each pixel represents the "probability density of the head", and the final number of people is calculated by integration. The advantage of this method is that it can handle scenes with high-density scenarios and the training process is relatively stable, but the model depends heavily on the quality of density map generation, which may introduce noise and have high errors in extremely sparse or long-distance images.

The Detection-based Method transforms crowd counting into an object detection task, directly detects each person in the image, and calculates the number of detection boxes as the number of people. This method is very accurate in positioning in sparse crowd environments, but performs poorly in dense crowds, is prone to missed detection, and is seriously affected by occlusion and scale changes.

The Point-based Method directly regresses the position of each person, that is, converts counting into a set prediction. Unlike the map-based pixel density regression, the point-based method directly predicts a set of (x_i, y_i) coordinates. Its advantage is that it is lighter, directly predicts the position,

and is simple to label with only point coordinates, but it is easy to miss dense areas and is difficult to train.

2.2 Domain Adaptation (DA) and Single Domain Generalization (SDG) for Crowd Counting

Deep learning-based methods have been widely used in crowd-counting problems. However, many of these methods tend to have varying degrees of performance degradation when testing data from unknown scenarios, which is the so-called "domain shift" problem. There are currently two ideas to solve this problem. **Domain adaptation (DA)** has been widely studied in the field of crowd counting, aiming to solve the domain shift problem by adjusting the source domain information to a specific target domain. Despite its significant progress, DA methods usually require data from the target domain, which may not be easy to obtain in practice. Therefore, models with strong generalization capabilities become very important in real-world applications, which is also the focus of this study. **Domain generalization (DG)** methods aim to train networks with generalization capabilities using only source domain data, while **single domain generalization (SDG)** is a special case when only one source domain is available. This study simulates this SDG situation and explores its performance in unknown target domains by training the model only in a narrowly distributed source domain.

3 Methodology

This section introduces the crowd counting models we explored and evaluated, covering various architectures and frameworks such as Convolutional Neural Networks, Vision Transformer, etc. MPCount is our main focus. We introduce some adjustments aimed at improving its performance and deployment efficiency.

3.1 UNet

UNet is a popular fully convolutional neural network (FCNN) architecture originally conceived for biomedical images segmentation [11]. It is characterized by a encoder-decoder structure. The encoder uses multiple layers to down-sample the image, extracting features at each layer. The output is constructed in the expanding decoder layers, leveraging skip connections from encoder layers to contextualize the decoder layer outputs. UNet's creator characterizes it as being very data efficient, and able to take great advantage of data augmentations. UNet is trained to output a density map guided by focal loss, and structural similarity index measure (SSIM).

3.2 DGCCUS

Domain Generalization for Crowd Counting in Unseen Scenarios (DGCCUS) [2] is the first method to introduce domain generalization into crowd counting. It combined dynamic subdomain division scheme with a meta-learning framework. A dual-branch architecture re-encodes image features into domain-invariant and domain-specific representations via memory modules. Feature disentanglement is guided by reconstruction and orthogonal losses.

3.3 YOLO

YOLO (You Only Look Once) is a real-time object detection model known for its speed and accuracy. YOLO directly predicts bounding boxes and class probabilities from an input image in a single forward pass through a Convolutional Neural Network (CNN). The image is divided into grids, with each grid cell responsible for detecting objects within its region by predicting bounding boxes and associated confidence scores [10].

In high-density crowd scenarios, YOLO often struggles due to its use of Non-Maximum Suppression (NMS). When multiple overlapping bounding boxes of the same class are predicted, NMS retains only the one with the highest confidence. As a result, closely grouped individuals may be merged into a single detection, leading to undercounting [1]. Additionally, if an object appears too small in the image, YOLO's accuracy may decline due to the lack of sufficient visual details and features. As a result, the model may fail to detect the object.

Aware of YOLO's limitations, we curated the training set from the NWPU crowd dataset to enhance detection. We selected images where individuals appeared larger, excluded densely crowded scenes, and avoided tightly grouped people to reduce overlap issues. Since original labels only marked heads, we relabeled the data with full-body annotations to provide richer spatial features for YOLO.

3.4 MPN

ModernPointNet (MPN) builds on P2PNet [12] by refining its point-based crowd counting approach. While P2PNet directly predicts individual coordinates instead of using density maps, MPN incorporates several architectural modifications aimed at enhancing performance.

The backbone was updated from VGG16_bn to ResNet101 [3], which improves feature extraction through residual connections. This update enables the model to capture complex patterns in crowded scenes, especially under varying densities and occlusions. A notable innovation in MPN is the integration of attention mechanisms through the Convolutional Block Attention Module (CBAM) [16]. By applying both channel and spatial attention, CBAM helps the model prioritize informative regions and reduce the impact of irrelevant background details. It is strategically placed after the feature pyramid network to refine the final feature representation. To address variations in crowd density and person size, a custom MultiScaleAttentionFusion module was developed. This module fuses features from multiple scales in the feature pyramid [6], assigning attention weights based on their relevance to the prediction task.

Additionally, the feature pyramid network was enhanced with residual connections to improve gradient flow during training, and its upsampling paths were refined to better preserve spatial information. Both the point detection and classification heads were streamlined to reduce computational complexity while maintaining their functions, and the Hungarian algorithm [5] for bipartite matching was retained from P2PNet. The training approach remains unchanged, jointly optimizing classification accuracy for point confidence and regression precision for point coordinates.

3.5 ED

In this approach, we propose a novel approach to crowd counting that leverages a state-of-the-art Vision Transformer and dynamic exemplar [15] extraction. Our model, built on the `vit_base_patch16_384` backbone, processes the input image through a transformer encoder to generate high-dimensional feature representations. In contrast to traditional methods that solely rely on the feature extraction power of Visual Transformers, our method brings the Visual Transformer more information about the image by running a YOLO model to detect candidate regions—specifically, regions with a detection confidence of 0.5 or above, during training. These detected regions are then cropped, resized to a fixed size (64×64), and embedded via a dedicated patch embedding module. The exemplar features are aggregated with the main image features, and a decoder network reconstructs a density map from which the final crowd count is inferred. The exemplar features act like a guide to the Visual Transformer, telling it what it should be detecting and regressing from the image.

3.6 SWIN

In this approach, we leverage the strong representation learning capabilities of Swin Transformers [7] and the EBC (Enhanced Block-wise Classification) [8] method to address the challenges of crowd counting. Specifically, we adopt a pretrained Swin Transformer backbone and adapt it for our task by modifying both the input resolution and output processing. For the baseline model, we resize each input image to 224×224, which is fed into the Swin Transformer. The backbone produces a feature map (of approximately 7×7 tokens) which is then upsampled via a learnable deconvolution (ConvTranspose2d) to a 28×28 grid. Each token in this grid is classified into one of five classes (corresponding to counts $0, 1, 2, 3$, or 4) using a linear prediction head.

To mitigate overfitting given the small dataset (approximately 300 images), we incorporate dropout layers both within the backbone (by setting appropriate values for `drop_rate` and `attn_drop_rate`) and before the final classification head. In addition, we employ weight decay in the optimizer and a cosine annealing warm restarts learning rate scheduler. These modifications help regularize the network, ensuring that the model learns robust features that generalize well to unseen data despite the limited training set.

3.7 MPCount

MPCount addresses the single domain generalization (SDG) for crowd counting, which is designed to overcome performance degradation in unseen scenarios caused by domain shift and label ambiguity. MPCount introduces an Attention Memory Bank (AMB) to store domain-invariant features through attention-based feature reconstruction while avoiding sub-domain partitioning for compatibility with narrow source distributions. An Attention Consistency Loss (ACL) stabilizes learning by aligning attention scores. A Content Error Mask (CEM) filters domain-specific content by comparing instance-normalized features from original and augmented images. To mitigate label ambiguity, Patch-wise Classification (PC) divides images into patches (e.g. 16×16) and classifies them as containing heads or not, thus using more reliable labels to refine density predictions.

3.7.1 Adjustments

To improve the alignment between validation and test-time performance, as well as to enhance the robustness and efficiency of the evaluation process, we made the following modifications:

- **Loss function adjustment.** We replaced the commonly used Mean Absolute Error (MAE) with the Mean Absolute Percentage Error (MAPE) as the validation loss. This change better reflects relative prediction errors and is more consistent with the focus of our study.
- **Validation set reconstruction.** We restructured the validation set to match the distribution of crowd counts in our test sets. Specifically, we selected 358 images from UCF-QNRF [4] dataset to form a validation set with comparable density ranges, focusing on low to medium crowd scenes.
- **Extended data augmentation.** In addition to standard augmentations such as *ColorJitter*, *GaussianBlur*, and *RandomAdjustSharpness*, we introduced additional transformations including *RandomRain, RandomFog, motion blur*, and both *horizontal and vertical flipping*. These augmentations are intended to simulate realistic adverse conditions and improve generalization.
- **Inference memory optimization.** The original inference pipeline preloaded all inference images into the GPU, which led to memory overflow when processing high-resolution batches. We rewrote the *inference.py* script to load, infer, and release memory sequentially on a per-image basis, significantly reducing GPU usage and enabling stable inference on 4GB memory GPUs.

4 Illustrative Experimental Results

4.1 Dataset

We introduce the datasets used in the experiments. Except for the YOLO model, which is trained on the specific NWPU dataset due to its limitations, all other models are trained on SHA and tested on SHB and MosseJaw datasets.

- **ShanghaiTech** [17] dataset is divided into two subsets: ShanghaiTech part A (SHA) and ShanghaiTech part B (SHB). SHA contains 300 training images and 182 testing images with an average of 500 people per image, while SHB contains 400 training images and 316 testing images with an average of 120 people per image.
- **The Moose Jaw** dataset, provided by the City of Moose Jaw, consists of two unlabelled surveillance videos showing indoor scenes of the Moose Jaw City recreation facility. To facilitate evaluation, we extracted each video frame as an individual image. After preprocessing, Video 1 (V1) yielded 222 images, and Video 2 (V2) yielded 920 images. Since the dataset lacks ground-truth annotations, we manually labelled every tenth frame for testing purposes (denoting LV1 and LV2, respectively). The first video contains about 14 people per frame, and the second video contains about 40 people per frame.
- **Northwestern Polytechnical University-Crowd (NWPU)** [14] is another dataset that we used. It contains more than 3000 labelled images. The number of people per image ranges from 0 to more than 500. From this dataset, we manually selected 600 images with fewer than 60 people. Among these, 400 images were used for training and 200 for validation.

4.2 Evaluation Metrics

We evaluate our method with mean absolute error (MAE), mean squared error (MSE), and Root Mean Squared Error (RMSE), defined as follows:

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \qquad \text{MAPE} = \frac{100\%}{N}\sum_{i=1}^{N}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \qquad \text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$$

where N is the number of testing images, y_i is the ground truth count of the i-th image and $\hat{y}_i$ is the predicted count. Lower values of all metrics indicate better performance.

In addition to using MAE and RMSE in line with standard industry practice, we also used MAPE as a performance metric because it is a percentage that does not depend on the unit of the predicted value and can directly compare model performance on data of different sizes, which is in line with the focus of our study.

4.3 Data Augmentation

To enhance the generalization and robustness of the model during training, we incorporated various data augmentation techniques, including random horizontal and vertical flipping, Colour Jittering, Gaussian blur, Motion blur, Pepper-Salt Noise, etc. Given the limited size of the training dataset, these augmentations were also expected to mitigate potential overfitting. Nevertheless, experimental results indicated that most data augmentation strategies did not yield performance improvements on the test set.

Taking the DGCCUS model as an example, we introduced additional Colour Jittering and Motion blur augmentations to the baseline horizontal flipping strategy, aiming to improve robustness under varying illumination conditions and dynamic blurring scenarios. Despite these efforts, no performance enhancements were observed. Another example is the MPCount model, for which the specific data augmentation methods are detailed in Section 3.7.1, and corresponding results are presented in Section 4.5, and both performance metrics (MAE, MAPE) showed declines to varying extents. UNet was the only model observed to benefit from the use of data augmentations. While the addition of data augmentations showed a consistent trend of improved performance on both LV1 and LV2 subsets, they led to a slight degradation on the SHB dataset, suggesting a trade-off between domain-specific tuning and generalization.

We hypothesize that the limited impact of these augmentation techniques on model performance arises from the relatively stable lighting conditions and consistent image quality present in the datasets used. Specifically, the majority of images in the SHB dataset were captured under consistent, sunny conditions, while the Moosejaw dataset predominantly consists of well-lit indoor scenes with minimal subject motion. Consequently, the introduced augmentations did not significantly alter the underlying data distribution.

Despite their limited effectiveness under current evaluation conditions, these augmentation methods may still provide substantial benefits in practical deployment scenarios. Real-world applications often involve more variable lighting conditions and dynamic motion-induced blurring due to camera movements or rapid object motion. Therefore, continued consideration of such augmentation techniques remains warranted for enhancing model robustness and generalization in actual operational environments.

4.4 Quantitative Results

Table 1 shows the quantitative comparison results of all the models we explored on the SHB dataset. Among all the methods, MPCount (MAE) has the best overall performance with an MAE of 11.19 and an RMSE of 19.00, while MPCount (MAPE) has the lowest MAPE of only 8.78%. UNet also performs very competitively and has great potential for improvement, with its MAE (11.43) and MAPE (9.89%) close to our best results. YOLO performs poorly in dense crowd scenes and has difficulty handling small and overlapping objects even when trained in a separately designed NWPU, which is reflected in its large MAE (80) and MAPE (48.64%).

Method	MAE	MAPE (%)	RMSE
UNet	11.43	9.89	23.09
DGCCUS	13.5	11.86	12.84
YOLO	80	48.64	121.26
MPN	18.99	16.04	29.55
ED	19.87	16.28	31.27
SWIN	13.27	10.33	-
MPCount (MAPE)[1]	12.51	**8.78**	22.89
MPCount (MAE)[2]	**11.19**	9.62	**19.00**

Table 1: Comparison of model performance on SHB

To further evaluate the generalization ability, we also tested all models on two subsets of the Moose Jaw dataset, LV1 and LV2. As shown in Table 2, MPCount outperforms other models on both subsets. In LV1, MPCount (MAE) is the lowest among the three metrics, while in LV2, MPCount (MAPE) has the best overall performance. Meanwhile, YOLO's performance has improved significantly, especially in LV1, where it has approached the best-performing MPCount. This is because LV1 consists of sparse scenes with less occlusion, and this condition is more favourable for detection-based methods. However, its error rate rises significantly in LV2. UNet also shows a stable performance trend, but its MAE and MAPE are slightly higher than MPCount. Other models, such as MPN and ED, are still less competitive.

Method	MAE	MAPE (%)	RMSE
UNet	2.59	19.83	3.11
DGCCUS	4.45	32.52	5.03
YOLO	2.14	15.60	2.50
MPN	4.00	30.11	4.81
ED	4.28	31.22	5.01
SWIN	2.93	20.18	-
MPCount (MAPE)	1.63	12.42	2.03
MPCount (MAE)	**1.50**	**11.32**	**1.88**

(a) LV1 (approx. 14 people on average)

Method	MAE	MAPE (%)	RMSE
UNet	5.30	14.13	6.88
DGCCUS	12.83	34.09	13.42
YOLO	9.57	24.65	10.07
MPN	13.46	35.54	14.49
ED	16.93	40.11	18.79
SWIN	13.18	34.88	-
MPCount (MAPE)	**3.28**	**8.72**	**3.84**
MPCount (MAE)	3.84	10.10	4.48

(b) LV2 (approx. 40 people on average)

Table 2: Comparison of model performance on LV1 and LV2

4.5 Progressive Ablation Study for MPCount

To assess the effectiveness of each proposed refinement described in Section 3.7.1, we conduct a stepwise ablation study on the SHB dataset. Starting from the baseline model, we gradually apply each modification and evaluate the resulting performance. The results are reported in the table below:

Method Variant	MAPE (%)	RMSE	GPU memory (Mib)
MPCount (baseline)	11.36	31.56	5553
+ MAPE validation loss	9.29	26.45	5553
+ validation set reconstruction	8.78	22.89	5553
+ extended data augmentation	11.44	32.607	5553
+ inference memory optimization (final)	8.78	22.89	733

Table 3: Stepwise ablation study on SHB dataset.

As shown in Table 3, most individual modifications contribute to improved performance over the baseline. Replacing the validation set loss function with MAPE yields more informative feedback, and reconstructing the validation set to match the test distribution also improves generalization.

However, incorporating extended data augmentation (e.g., *RandomRain*, *RandomFog*, *MotionBlur*) resulted in a slight performance drop. We hypothesize that the added visual perturbations may have introduced excessive domain noise, which could reduce the model's confidence on relatively clean test data. As a result, we ultimately excluded these augmentations from the final model.

[1]Best MAPE MPCount model during training.

[2]Best MAE MPCount model during training.

Finally, optimizing the inference memory pipeline has no direct effect on model accuracy but significantly reduces GPU memory consumption, thus enabling stable inference on resource-constrained devices.

5 Conclusion

This paper addresses the problem of crowd counting under the challenging setting of single domain generalization (SDG), where the model is trained on a single source domain and evaluated on unseen target domains. We conduct a systematic comparison of various crowd counting approaches, including convolutional architectures, attention-based models, and detection frameworks, to analyze their performance across different densities, scales, and occlusions.

Experimental results show that MPCount achieves state-of-the-art performance on both subsets of the ShanghaiTech Part B dataset and the real-world Moose Jaw dataset, outperforming other models in sparse (LV1) and medium-density (LV2, SHB) scenarios. To enhance generalization, we introduce several refinements including validation set reconstruction, the use of MAPE as the validation loss, and inference memory optimization. Ablation studies confirm the effectiveness of these components, while also revealing trade-offs between validation strategies and data augmentation.

Other models such as UNet also demonstrate strong performance and remain promising with higher-resolution inputs. Detection-based models like YOLO perform well in sparse scenes but degrade significantly in dense settings, clearly illustrating their limitations under occlusion and scale variation. Transformer-based models such as SWIN exhibit stable performance, suggesting its potential as a good alternative.

5.1 Limitations

Although our models perform well on the ShanghaiTech dataset and the Moose Jaw dataset, the accuracy of the model drops significantly in some scenarios, such as people in mirrors, faces on posters, and models in store windows. These scenes are highly similar to real people in appearance, but are pseudo-targets in semantics.

It is difficult for the model to recognize these scenes with distractors. On the one hand, the training dataset is not perfect and lacks samples of such scenarios with visual distractors, which limits the generalization ability of the model. On the other hand, more importantly, the model often relies on appearance features to judge whether the target is a "person", but lacks higher-level semantic understanding and physical common sense reasoning capabilities.

5.2 Future Work

From the data level, obtaining more diverse training samples, including scenes with visual distractors, can further enhance the robustness in various real-world scenarios. Meanwhile, our current method applies a fixed-size Gaussian kernel to all annotations regardless of the object scale. Following Wan et al. [13], using adaptive scale-aware kernels that vary with head size or distance can provide more accurate supervision, especially for density-based models.

From the model level, we observe that even in video-based scenarios, the current model processes each frame independently. Incorporating temporal cues such as inter-frame motion or predictions based on previous time steps can introduce temporal consistency and improve the accuracy and stability of crowd estimation across consecutive frames.

Acknowledgements

We would like to extend our thanks to Professor Russ Greiner and TA Weijie Sun for their valuable guidance and inspiration during the project. We would also like to thank our domain expert Alex Wang for providing us with the data of the MooseJaw City.

References

[1] M. Ali, T. Aly, A. Raslan, M. Gheith, and E. Amin. Advancing crowd object detection: A review of yolo, cnn and vits hybrid approach. *Journal of Intelligent Learning Systems and Applications*, 16:175–221, 2024.

[2] Zhipeng Du, Jiankang Deng, and Miaojing Shi. Domain-General Crowd Counting in Unseen Scenarios. *arXiv*, 2023.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[4] Haroon Idrees, Mohsen Tayyab, Naveed Athrey, Dong Zhang, Somaya Al-Maadeed, Nasir Rajpoot, and Mubarak Shah. Composition loss for counting, density map estimation and localization in dense crowds. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–546, 2018.

[5] Harold W. Kuhn. *The Hungarian Method for the Assignment Problem*, pages 29–47. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection, 2017.

[7] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *International Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[8] Y. Ma, V. Sanchez, and T. Guha. CLIP-EBC: CLIP Can Count Accurately through Enhanced Blockwise Classification. *arXiv*, 2024.

[9] Zhuoxuan Peng and S.-H. Gary Chan. Single domain generalization for crowd counting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR 2024)*, 2024.

[10] Ashish Ranjan, Namrata Pathare, Sunita Dhavale, and Suresh Kumar. Performance analysis of yolo algorithms for real-time crowd counting. In *2022 2nd Asian Conference on Innovation in Technology (ASIANCON)*, pages 1–8, 2022.

[11] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.

[12] Qingyu Song, Changan Wang, Zhengkai Jiang, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yang Wu. Rethinking counting and localization in crowds:a purely point-based framework, 2021.

[13] Jia Wan and Antoni Chan. Adaptive density map generation for crowd counting. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1130–1139, 2019.

[14] Qi Wang, Junyu Gao, Wei Lin, and Xuelong Li. Nwpu-crowd: A large-scale benchmark for crowd counting and localization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[15] Zhicheng Wang, Liwen Xiao, Zhiguo Cao, and Hao Lu. Vision transformer off-the-shelf: A surprising baseline for few-shot class-agnostic counting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[16] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module, 2018.

[17] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. Single-image crowd counting via multi-column convolutional neural network. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 589–597, 2016.

A Appendix

A.1 GPU Memory Usage comparison

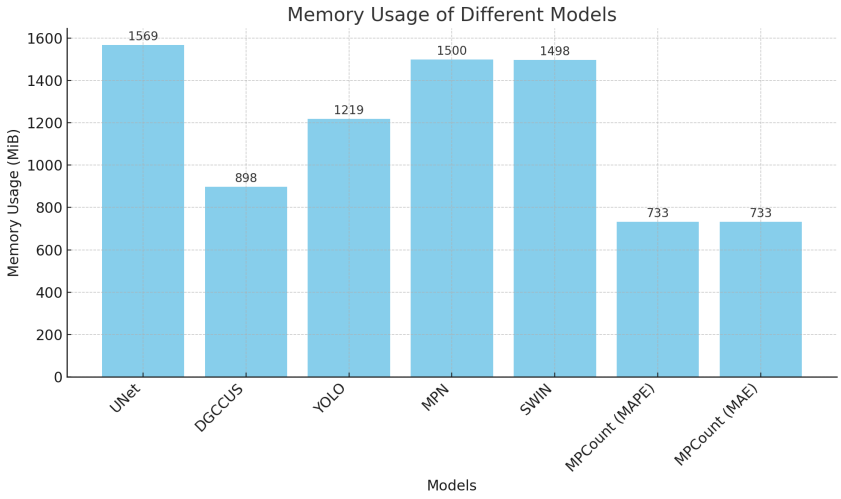
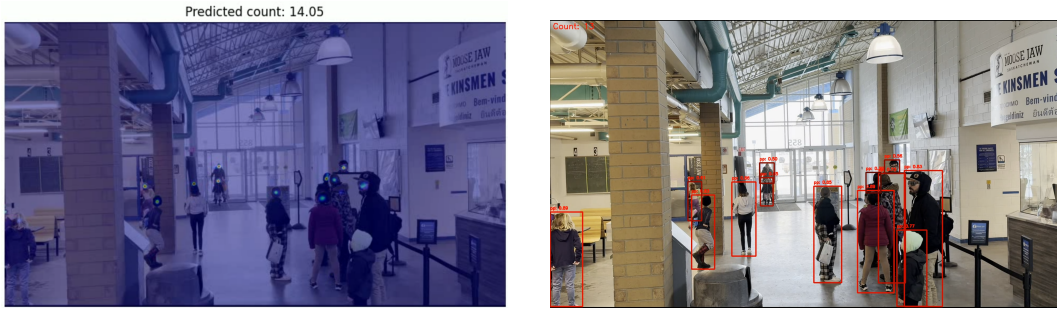


Figure 3: This bar chart illustrates the GPU memory usage of each model. GPU memory is a key consideration because the models must be deployable on devices with only 4 GB of GPU memory. For each model, the required input resolution and the image size used during inference are different. Therefore, we cannot directly compare memory efficiency across models. Instead, we focus on ensuring that each model runs successfully within the 4 GB memory limit.

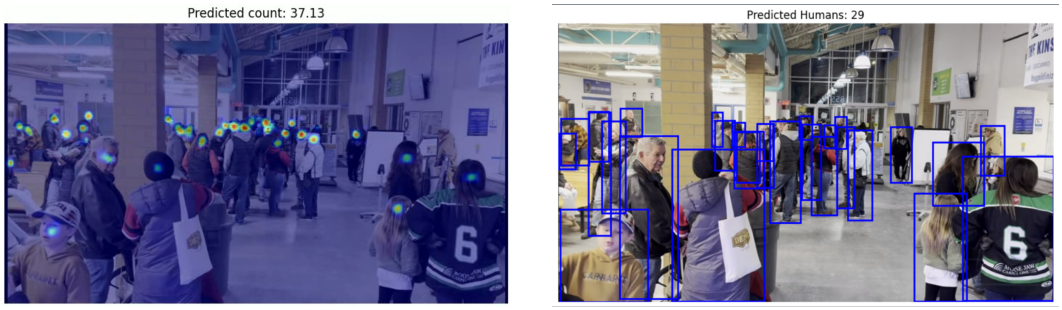
A.2 Demo images from the first Moose Jaw video, Comparing MPCount and YOLO



(a) MPCount (b) YOLO

Figure 4: The scene in the first Moose Jaw video, which contains an average of 11 people, is relatively simple. The two demo images above show how YOLO and MPCount perform on this scene. Note that the results from these two models are similar, and also consistent with the outputs of the other models. Therefore, testing on the first Moose Jaw video does not provide meaningful differentiation for evaluating model performance.

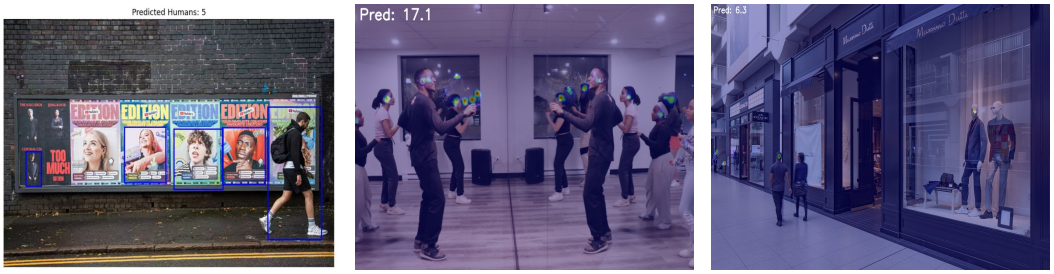
A.3 Demo images from the second Moose Jaw video, Comparing MPCount and YOLO



(a) MPCount (b) YOLO

Figure 5: The scene in the second Moose Jaw video, which contains an average of 40 people, is denser. The two demo images above illustrate how YOLO and MPCount perform in this scenario. The actual human count is 41, with MPCount estimating 37 and YOLO estimating 29. We found that several models, including YOLO, perform well on the first Moose Jaw video but struggle with the second. As a result, we chose to evaluate the models on the second video, as it is more complex, better suited for testing, and provides a more accurate representation of real-world scenarios.

A.4 Failure Scenarios



Poster Mirror Mannequin