DLSS Project - Social Distancing during COVID

Maximilian Weiland, Nina Geyer, Adrian Ruhe, Raphaela Keßler

August 27, 2024

1 Overview

This project explores the application of Convolutional Neural Networks (CNNs) to evaluate social distancing measures using surveillance and crowd imagery. The aim is to use CNN architectures to recognise and evaluate compliance with social distancing guidelines in public spaces. This approach is intended to help health authorities and policy makers manage and enforce social distancing during pandemics. We approached the topic with three approaches: we built a YOLO model from scratch, which could not be trained due to limited computer power, we assembled our own CNN model and we fine-tuned a pre-trained yolo v8 model. We decided to focus on our own, lightweight CNN model. The code for this project is available in this repository: https://github.com/Raphaaella/CNN-to-Evaluate-Social-Distancing-Measures.

2 Literature Review

To assess the effectiveness of social distancing, it's important to automatically detect individuals in images. In computer vision, this falls under object detection, with crowd counting focusing specifically on estimating the number of people without predicting their exact locations.
Crowd counting methods can be based on object detection, regression-based approaches, or techniques that require generating a density map from the image and then estimating the count from this map [Patwal et al., 2023]. Object detection usually operates by applying a sliding window that moves over the image and detects the exact position of people [Felzenszwalb et al., 2009]. Advances in deep learning have led to significant improvements in recognition accuracy by harnessing the potential of CNNs, with the most innovative method being You Only Look Once (YOLO) [Girshick et al., 2014, Girshick, 2015, Redmon et al., 2016]. Object detection-based approaches aim at predicting peoples' exact location which allows determining the exact distance between them [Parikh et al., 2023]. While these methods perform well for sparse crowds, they struggle in challenging conditions like occluded or low-resolution images of dense crowds [Gao et al., 2020].
Regression-based methods aim to overcome these issues by extracting global [Kong et al., 2005] or local [Chan et al., 2008] features, such as gradient, texture, or edge pixels. The regression model then tries to learn a direct mapping between these low-level features and the total people count [Chen et al., 2012]. This approach disregards any spatial information. Counting methods based on the estimation of density maps have been proposed to address this problem [Lempitsky and Zisserman, 2010]. The estimated density maps should contain enough spatial information to accurately predict the number of people even in very dense crowds. Recent studies have suggested to automate the feature extraction by applying CNNs [Zhang et al., 2015, Walach and Wolf, 2016].
The rapid progress in deep learning has led to the development of numerous new methods such as encoder-decoder structures [Yi et al., 2023]. Moreover, the success of the Transformer model in Natural Language Processing (NLP) initiated researchers to apply this architecture also to crowd counting [Deng et al., 2024]. However, the Transformer architecture still does not reveal the same promising results in this field as it does in NLP [Khan et al., 2023].
The COVID-19 pandemic has created an urgent need for new methods to evaluate the effectiveness of social distancing measures. Rezaei and Azarmi published a paper on tracking social distancing methods with CNNs [Rezaei and Azarmi, 2020]. Further research set the focus on developing lightweight CNN structures that allow respective systems to operate in real-time [Vivekanandam, 2021, Mokayed et al., 2023]. Other scientists developed a general computer vision system capable of performing face mask recognition, face-hand interaction and social distance measurement [Eyiokur et al., 2023]. Measuring the effectiveness of social distancing has also been recognized as a prime example of the link between computer vision and social science research

[Bernasco et al., 2023].

3 Data Preprocessing

3.1 The Data

We used data from MS COCO, Oxford Town Centre and EarthCam, which includes crowds on the streets as well as people in other contexts.
The COCO dataset is a large-scale resource for object detection, segmentation, and captioning, widely used in visual recognition research. It includes over 330 000 images, 1.5 million object instances, and rich annotations for tasks like object detection, segmentation, human pose estimation, and image captions [Lin et al., 2014].
The Oxford Town Centre dataset is a surveillance dataset that captures pedestrian activity on a London street. It was used, for example, in the paper by Rezaei and Azarmi [Rezaei and Azarmi, 2020]. It contains annotations for detecting and tracking pedestrians, including bounding boxes around detected individuals and tracking IDs to track their movements across frames [Benfold and Reid, 2011]. As the original Oxford Town dataset is no longer available, we used a reduced copy of it with 3 090 annotations.
We combined the Oxford and COCO dataset for training to ensure that the model learns how to detect people in a diverse set of images. While the Oxford dataset contains only images of one street-like setting, the COCO dataset also includes various other scenes that show people in many different settings and perspectives.
Surveillance camera data to measure the effectiveness of social distancing measures comes from the Earthcam webpage, a global provider of live streaming of public spaces [EarthCam, Inc., 2024]. To explore whether a CNN model can assess social distancing, we scraped Earthcam data from 2019, 2020, and 2022 in New York, New Orleans, and London. These cities were selected due to more permissive surveillance policies in the US and UK, ensuring data availability and stable camera positions for consistent image comparison.
We scraped two weeks of data per year, starting with one week in 2020 during lockdown and one week after restrictions were eased. We assumed fewer people in public during lockdown and slightly more afterwards. We filtered images from the same weeks in 2019 and 2022 to track changes and compare pre- and post-COVID data. We chose 2022 instead of 2021 since COVID measures were still in place in 2021. For London in 2019, missing data required shifting the timeframe by two days, resulting in 9 days of lockdown and 5 days of eased measures. Plot 6 in the appendix shows the detailed time periods.

3.2 Data Cleaning

The COCO images are downloaded using the fiftyone API and filtered for having object detection annotations. Since the task is to detect people in public places such as streets, the images are further filtered to create a representative training sample. This was done by linking the COCO images to their captions and filtering for stemmed keywords like "street", "road", "city", "crosswalk". An unfiltered dataset has also remained to test how dependent the model is on suitable training images. Due to memory shortage, 35 000 images could be downloaded initially from which ∼18 000 remained without and ∼9 000 remained with caption filtering.
The Oxford dataset consists of a 5-minute video and 3 090 corresponding annotations. The video was split by frame into 3 090 individual images that were then be mapped to the annotations. Both datasets were then combined to create a single representative dataset that can be used to train a model capable of counting the number of people in a diverse range of images.
The surveillance data for the analysis was scraped from the Earthcam webpage and initially all images for the respective time periods were downloaded. The data was then manually filtered to select ten images per day (five per day for New York) that had a consistent frame (since cameras occasionally zoomed in, altering the frame) and were taken during daylight.
All images were resized to a resolution of 128x128 because of limited computational power on publicly available GPUs and improved batch processing efficiency. Instead of cropping, zero-padding was applied to ensure that the original content of the image was preserved without distortion. Squaring and padding ensure consistent feature map sizes across network layers, preserving spatial relationships in the image. This consistency is essential for CNNs to effectively learn and recognize patterns. Bounding box annotations had to be recalculated to ensure that they keep their relative position and after resizing. The training images were transformed using data augmentation techniques like flipping, rotating or color jitter. Images could not be cropped during transformation because it would have distorted the people count labels. While the

transformations are only applied to the training data to help create a more diverse data set that performs better on unseen data, both training and validation/test datasets were also normalised to help the model train more effectively. For normalisation, we used standard values applied in many other computer vision data sets. In order to get the count of people in an image, we counted the number of bounding boxes in the annotations.

3.3 Preliminary Data Analysis

To get a better idea of the data used to train our model, we visualized a sample of the COCO and Oxford Town Centre datasets (figure 1). People are highlighted in green bounding boxes, illustrating the clear differences between the two datasets. The COCO dataset features diverse images, including street scenes similar to surveillance footage as well as close-up portraits. In contrast, the Oxford Town Centre dataset contains only CCTV images from a fixed-perspective camera.

Figure 1: Example Images with Bounding Boxes from Oxford and COCO Dataset

To gain further insight into the training data, we created a heat map based on the Oxford Town Centre data (figure 7). The graph shows which areas of the street were highly frequented and which were not. To do this, we split the images into individual grid cells and counted the number of annotations that the data contained for each individual cell. This aggregation was possible because all images were taken by a camera that does not change its perspective. The visualization shows that most people were walking on the sidewalks, while very few were found to be directly on the street.

4 Model Architectures and Hyperparameters

We have worked intensively on the structure of the YOLO model and tried to re-code it. The code can be viewed in the repository. Rebuilding the complex YOLO model, as well as the limited computational power, brought us to limits. Therefore, we decided to use a simpler CNN model which is just able to count the number of people on an image without estimating bounding boxes around them. By counting people with the same camera setting, we can also estimate how empty or crowded a public area is and therefore how well social distancing measures are applied. As a third approach, we used a pre-trained v8 YOLO model and fine-tuned it with the Oxford dataset in order to draw bounding boxes around the people in the image. You can find the results of the third approach in the Appendix B.

4.1 The CNN Model

We developed a CNN model with a residual architecture using PyTorch, incorporating layers suited for image detection and segmentation tasks. The architecture includes convolutional, pooling, and fully connected layers optimized for identifying individuals. Figure 2 is a detailed breakdown of the architecture.

The architecture starts with an initial convolutional layer that processes the input image, which has three color channels (RGB). Afterwards, batch normalization is applied to normalize the activations, followed by a ReLU activation function that introduces non-linearity. The core of the model consists of four main layers, each composed of several instances of the "Basic Block". Each basic building block has two convolutional layers, followed by batch normalisation and a ReLU activation function. Residual learning is enabled in the Basic Block through a skip connection that adds the input of the block directly to its output. This skip

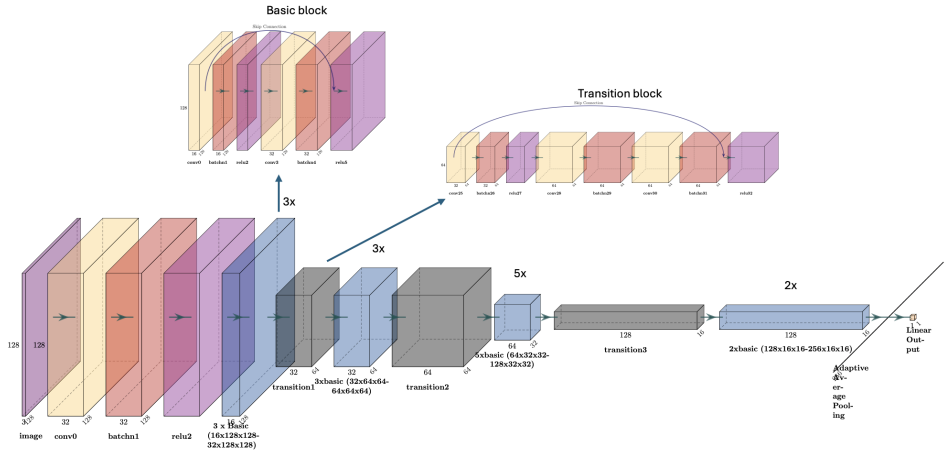

Figure 2: CNN architecture - Reduced to the main basic blocks and transitions

connection addresses the problem of vanishing gradients and helps to train deeper networks, by potentially allowing the input to bypass one or more layers.

The four main layers of the basic blocks progressively increase the number of filters (32, 64, 128, 256) allowing to capture more complex features of the images when moving deeper into the CNN. Each layer has a varying number of stacked basic blocks (3, 4, 6, 3) balancing the trade-off between capturing complex features and computational efficiency. As the network transitions through the layers, the spatial dimensions of the feature maps are reduced to manage computational complexity and focus on higher-level features. This reduction is achieved by using a stride of 2 in the first convolutional layer of each new block. Stride is the parameter that defines the number of pixel steps of the kernel movement across the input data.

After the last Basic Block, an adaptive average pooling layer is applied, reducing the spatial dimensions of the feature maps, resulting in a single feature vector per channel. This output is flattened and passed through the following fully connected layer to reduce the output dimensionality to match the number of final outputs, in this case the final count. For a detailed representation of the model, see the Appendix C.

4.2 Hyperparameters

The model was trained for a total of 100 epochs. We implemented a learning rate scheduler in the training function that adapts the learning rate based on the training progress. This approach allows for faster convergence at the start with a higher learning rate, while also ensuring a stable training process in later epochs by reducing the learning rate for more precise adjustments of the model parameters. Therefore, we chose a relatively high learning rate of 0.0001 in the beginning. We further apply regularisation to prevent overfitting and enhance generalization of the model. We experimented with the regularisation parameter, ultimately setting it also to 0.0001 as it yielded the best results. To add those optimization measures to the model, we used PyTorch's Adam Optimizer that uses L2 regularization. As a criterion to measure how well the model performs, we used the mean squared error. A further crucial measure to avoid overfitting is early stopping. The training is stopped when the validation loss is not improved anymore keeping the model from learning the noise in the data. We set the patience for the early stopping mechanism in the model to 25, meaning that the training process stops after 25 epochs without improvement on the validation loss.

5 Model Evaluation

Although we set the number of epochs to 100, our model stopped the training process after 74 epochs because the early stopping mechanism was triggered. As the graph below shows, the loss curve for the training set decreased steadily, while the loss of the validation set fluctuated much more in the beginning, but also decreased significantly in the long run. After around 35 epochs, the validation loss remained at a fairly constant level, which ultimately led to the early stopping of the training. Over the course of the entire training, the validation loss was higher than the training loss, which is to be expected as the model was specifically tailored to the training data. Clear evidence for over- or underfitting was not visible. During

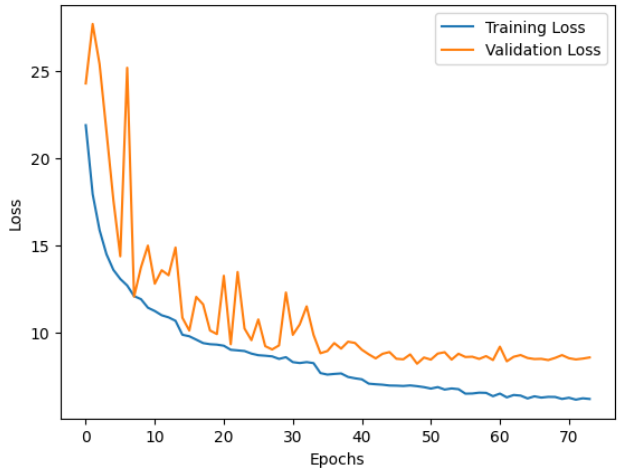

Figure 3: Loss Curves for the Training and Validation Set

training, it turned out that the model performs best when trained with the most data available, even if additional training data might not be representative for the tested data anymore. That is why we finally trained the model with all the images available that included any kind of person annotation.

We evaluated the performance of our model using the test set, the results of which are presented below. The model achieved a mean absolute error (MAE) of 1.79 and an R-squared of approximately 0.79 (Table 1). The MAE describes the error that the model makes on average when counting people in a picture without considering the direction of the error. It is important to note that we found significant differences in performance on the images from the COCO and Oxford datasets. The model was much more successful in learning the features from the Oxford dataset, achieving an R-squared of 0.96 on these images alone. The model has much more difficulty when dealing with images from the COCO dataset, resulting in an R-squared of only 0.51. This is probably due to the very different contexts of the COCO images, whereas the Oxford dataset has a fixed camera perspective and therefore shows very similar and comparable images. Due to the fact that we are dealing with surveillance footage in our analysis, we are confident that the model has successfully learned to count the number of people if it can successfully process the CCTV images from the Oxford dataset.

Metric	**Value**
Mean Absolute Error (MAE)	1.7972
Mean Squared Error (MSE)	7.3055
Root Mean Squared Error (RMSE)	2.7029
R-squared (R2)	0.7877

Table 1: Evaluation Metrics on the Test Set

To illustrate the model's internal workings, we visualize four feature maps from the final convolutional layer, just before the data enters the linear regression. Using images from the Oxford Street dataset, lighter colors in the maps indicate higher pixel values. The model recognizes the diagonal street and detects people through higher pixel values. However, these are just 4 of the 256 feature maps, so the model likely captures many more details across the others.

To get an impression of how well the model actually performs on unseen EarthCam images, we annotated 30 images manually and compared them to the model prediction. The predictions and performance metrics are available in the appendix (E). Even though this small sample cannot be used to evaluate the performance of the model, the examples still give an impression of how difficult it is to identify the individuals. It shows that the model is not able to perfectly count people in unseen camera sequences. While the predictions for smaller amounts of people are partly reasonable, it faces significant challenges when dealing with crowds of more than 20 individuals. Additionally, it occasionally has difficulties distinguishing between people and other objects.

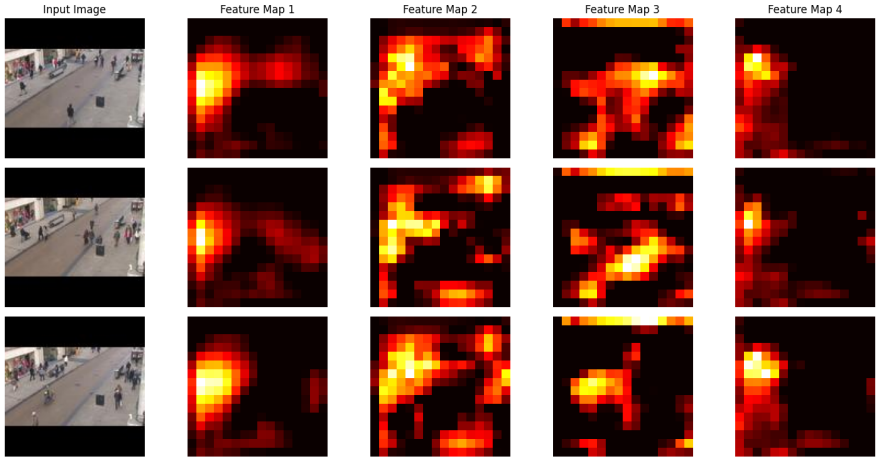

Figure 4: Feature Maps of the CNN Model on Images From Oxford Dataset

6 Results

The following chapter describes the insights gained by applying our model to real-world data. It also highlights the challenges, limits, and ethical considerations of this project.

6.1 Public Health Impact Analysis

Figure 5 illustrates the outcomes of applying the CNN model to Earthcam data, showing the average number of people detected across 5-10 selected CCTV images for each city. The red bars indicate the lockdown periods and the green bars the period when the lockdown was eased for each city in 2020.
All three cities have a different average number of people, which is due to the fundamental difference in vibrancy in the particular public places. all three cities show significant fluctuations in the number of people detected before the lockdown. These fluctuations can be partly attributed to the days of the week. We have not yet found an explanation for the peak on 24 May 2019 in New Orleans for example; research has not revealed any particular event or extreme weather on this day. In New Orleans and New York, the average number of people detected decreased in 2020 compared to 2019. This trend shows that our model could work, recognizing the reduced density of people and thus the increased distancing or hesitation to enter a public place. Nevertheless, in all three cities, the number of people detected by our model did not decrease as drastically during the lockdown as we had expected. In London, only the fluctuations in 2020 are lower than those in 2019 and 2022. Assuming our model is effective, this could be due to difficulties of the city in maintaining strict social distancing. There might be various reasons why the expected differences during, before, and after the lockdown are not apparent. Technical problems that we faced, which influenced the results, are described in the following paragraph.

6.2 Challenges and Limits

Due to limited computational resources, we were unable to train the YOLO model ourselves. With our second approach of counting people, we were capable of capturing people and measuring the density, but we could not perform accurate distance measurements. In a follow-up project, it would be interesting to see if the distance between people has actually changed and whether there are differences in the discipline of compliance with COVID measures in the various cities.

Furthermore, our memory capacity was also limited which resulted in a limited amount of images available for training purposes. Since the images could not be filtered for people detection labels before downloading, we had to discard around one third of the images post-download leading to an inefficient use of the already limited storage space.

We struggled to find CCTV images that relate to the correct time period before and after COVID. In addition, the camera setting should be reasonably comparable so that the density of people makes sense within a certain frame of reference. For the street in New York, for example, we realised that in some years, awnings and cafés took up the space, in other years, parked cars obscured the people. In London, the street

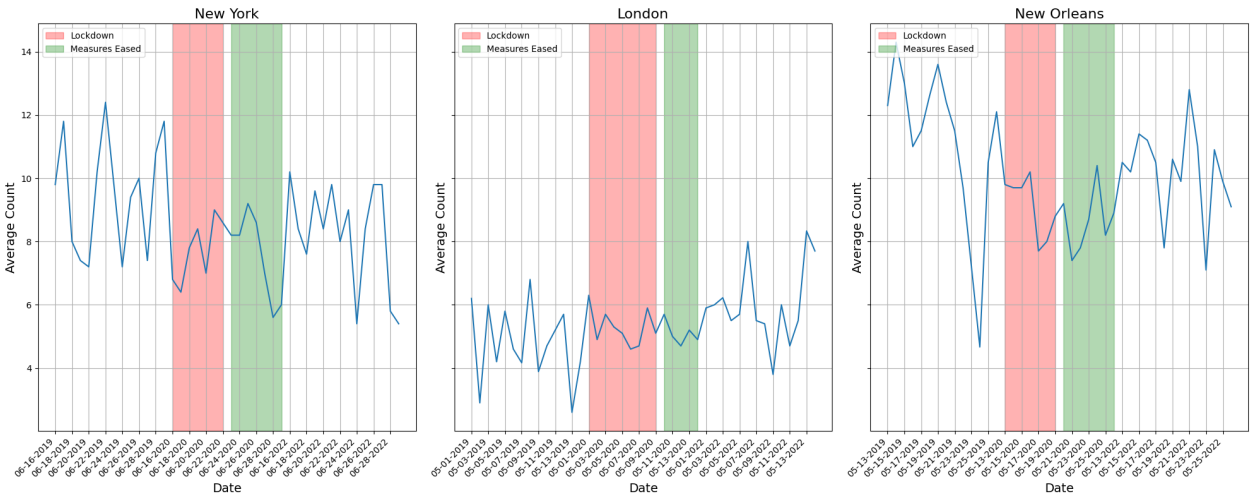

Figure 5: Estimated People Count with CNN Model

is no ordinary street; it shows the famous Abbey Road where the Beatles made their album cover. Many tourists come to this crosswalk for a photo, but when the traffic lights are red, there are no people on the road. Furthermore, the images in the archive of EarthCam do not represent the daily amount of people crossing that specific place because images seem to be saved arbitrarily and not in an equal frequency. This is why we chose to manually select images from the scraped dataset, trying to make it more representative. We would have preferred to compile more EarthCam data for analysis, for example to compare COVID measures in different phases of the pandemic and in more countries. However, the problem here was that only a limited number of countries share camera recordings of public places. In addition, for the counting approach to work effectively, it was necessary for the cameras to maintain consistent angles and settings. Unfortunately, EarthCam rarely offer cameras with such stability.

Another challenge that occurred was the annotations of the training images being very detailed. We could sometimes barely recognize people that were annotated in the picture by ourselves and therefore it is difficult for the model as well. Due to the limited computational power, we had to resize the images and reduce the resolution, resulting in an even less detailed view on the actual amount of people in the scene.

6.3 Ethical Considerations

The protection of privacy is important, as people can be recorded in public spaces without their explicit consent when surveillance and crowd images are captured and used. There is a risk that recognition algorithms could be misused for applications beyond the public health sector, for example for extensive state surveillance networks targeting ethnic minorities or political dissidents [Mozur, 2019]. This potential for abuse emphasises the need for robust ethical guidelines and controls to prevent AI from being used in ways that harm individual rights.

7 Summary

While the CNN model shows potential in detecting the number of people in public spaces, it cannot be reliably used by policymakers to monitor social distancing compliance due to several limitations.
Firstly, the model exhibits significant variability in its detection accuracy across different cities and time periods, as shown by the inconsistent results between New York, London, and New Orleans. Secondly, the model's performance during the lockdown phase does not clearly indicate a reduction in people counts, which would be expected if social distancing guidelines were effectively followed. This suggests that the model might not accurately capture the true number of people in all scenarios, especially in varying lighting conditions, perspectives, or when individuals are partially obscured. Additionally, the model's inability to measure the exact distances between people limits its usefulness for assessing adherence to social distancing rules.
Consequently, while the model provides valuable insights into general trends in crowds, it does not reveal insights that are necessary for policy decisions on adherence to or adaptation of social distancing rules.

References

[Benfold and Reid, 2011] Benfold, B. and Reid, I. (2011). Stable multi-target tracking in real-time surveillance video. *CVPR 2011*, pages 3457–3464.

[Bernasco et al., 2023] Bernasco, W., M. Hoeben, E., Koelma, D., Liebst, L. S., Thomas, J., Appelman, J., Snoek, C. G., and Lindegaard, M. R. (2023). Promise into practice: Application of computer vision in empirical research on social distancing. *Sociological Methods & Research*, 52(3):1239–1287.

[Chan et al., 2008] Chan, A. B., Liang, Z.-S. J., and Vasconcelos, N. (2008). Privacy preserving crowd monitoring: Counting people without people models or tracking. In *2008 IEEE conference on computer vision and pattern recognition*, pages 1–7. IEEE.

[Chen et al., 2012] Chen, K., Loy, C. C., Gong, S., and Xiang, T. (2012). Feature mining for localised crowd counting. In *Bmvc*, volume 1, page 3.

[Deng et al., 2024] Deng, M., Zhao, H., and Gao, M. (2024). Clformer: a unified transformer-based framework for weakly supervised crowd counting and localization. *The Visual Computer*, 40(2):1053–1067.

[EarthCam, Inc., 2024] EarthCam, Inc. (2024). Earthcam: Live streaming video and time-lapse cameras. Accessed: 2024-07-22.

[Eyiokur et al., 2023] Eyiokur, F. I., Ekenel, H. K., and Waibel, A. (2023). Unconstrained face mask and face-hand interaction datasets: building a computer vision system to help prevent the transmission of covid-19. *Signal, Image and Video Processing*, 17(4):1027–1034.

[Felzenszwalb et al., 2009] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2009). Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645.

[Gao et al., 2020] Gao, G., Gao, J., Liu, Q., Wang, Q., and Wang, Y. (2020). Cnn-based density estimation and crowd counting: A survey. *arXiv preprint arXiv:2003.12783*.

[Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

[Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[Khan et al., 2023] Khan, M. A., Menouar, H., and Hamila, R. (2023). Revisiting crowd counting: State-of-the-art, trends, and future perspectives. *Image and Vision Computing*, 129:104597.

[Kong et al., 2005] Kong, D., Gray, D., and Tao, H. (2005). Counting pedestrians in crowds using viewpoint invariant training. In *BMVC*, volume 1, page 2. Citeseer.

[Lempitsky and Zisserman, 2010] Lempitsky, V. and Zisserman, A. (2010). Learning to count objects in images. *Advances in neural information processing systems*, 23.

[Lin et al., 2014] Lin, T.-Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context.

[Mokayed et al., 2023] Mokayed, H., Quan, T. Z., Alkhaled, L., and Sivakumar, V. (2023). Real-time human detection and counting system using deep learning computer vision techniques. In *Artificial Intelligence and Applications*, volume 1, pages 221–229.

[Mozur, 2019] Mozur, P. (2019). One month, 500,000 face scans: How china is using a.i. to profile a minority. *The New York Times*.

[Parikh et al., 2023] Parikh, V., Chelani, Y., and Oza, P. (2023). Social distance detection using customized yolov4 (sddyv4) model. *International Journal of Computing and Digital Systems*, 14(1):10481–10489.

[Patwal et al., 2023] Patwal, A., Diwakar, M., Tripathi, V., and Singh, P. (2023). Crowd counting analysis using deep learning: A critical review. *Procedia Computer Science*, 218:2448–2458.

[Redmon et al., 2016] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection.

[Rezaei and Azarmi, 2020] Rezaei, M. and Azarmi, M. (2020). Deepsocial: Social distancing monitoring and infection risk assessment in covid-19 pandemic. *Applied Sciences*, 10(21).

[Terven et al., 2023] Terven, J., Córdova-Esparza, D.-M., and Romero-González, J.-A. (2023). A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas. *Machine Learning and Knowledge Extraction*, 5(4):1680–1716.

[Ultralytics, 2024] Ultralytics (2024). Yolo: A brief history. https://docs.ultralytics.com/#yolo-a-brief-history. Accessed: 2024-08-25.

[Vivekanandam, 2021] Vivekanandam, B. (2021). Speedy image crowd counting by light weight convolutional neural network. *Journal of Innovative Image Processing*, 3(3):208–222.

[Walach and Wolf, 2016] Walach, E. and Wolf, L. (2016). Learning to count with cnn boosting. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part II 14*, pages 660–676. Springer.

[Yi et al., 2023] Yi, J., Chen, F., Shen, Z., Xiang, Y., Xiao, S., and Zhou, W. (2023). An effective lightweight crowd counting method based on an encoder-decoder network for the internet of video things. *IEEE Internet of Things Journal*.

[Zhang et al., 2015] Zhang, C., Li, H., Wang, X., and Yang, X. (2015). Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 833–841.

Appendix

A Additional Plots

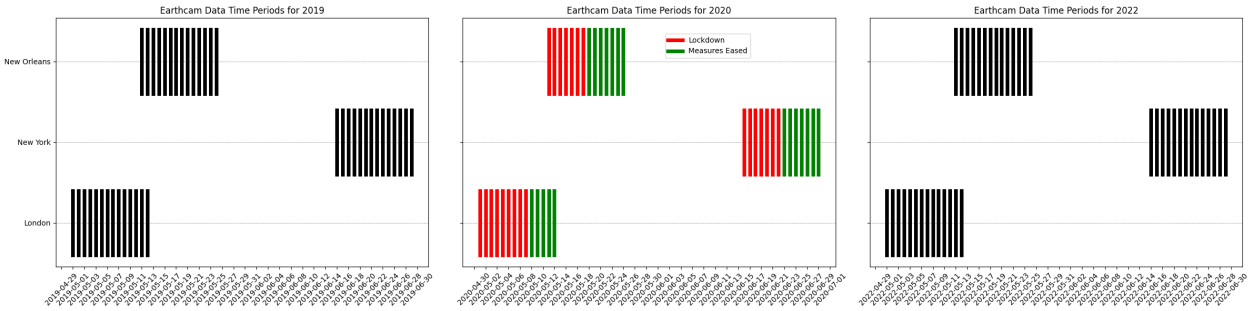

Figure 6: EarthCam Data Timeline

Figure 7: Heatmap of Most Frequently Visited Regions in the Oxford Dataset

B The Pre-Trained YOLO model

The YOLO model is an advanced real-time object recognition system known for its speed and accuracy. Using a pre-trained YOLO model to evaluate social distance measures can significantly improve the effectiveness and efficiency of the task. YOLO's architecture allows for the detection of multiple objects within an image through a single forward pass, making it exceptionally fast and suitable for real-time applications.
According to "You Only Look Once: Unified, Real-Time Object Detection," YOLO divides the input image into an S x S grid, with each grid cell predicting B bounding boxes and corresponding confidence scores [Redmon et al., 2016]. Each bounding box includes coordinates, dimensions, and a confidence score, while each grid cell also predicts a class probability map, indicating the likelihood of the presence of a particular class. This unified detection approach allows YOLO to achieve both high speed and accuracy by consolidating object detection tasks into a single neural network. YOLOv8 published by Ultralytics in 2023 offers an even more efficient architecture, with an optimized backbone and neck network that balances speed and accuracy more effectively than its predecessors [Ultralytics, 2024]. The model benefits from enhanced transfer learning capabilities, allowing it to adapt more effectively to domain-specific challenges [Terven et al., 2023].
By leveraging a pre-trained YOLO model, we can capitalize on its robust feature extraction capabilities, which have been fine-tuned on large datasets such as COCO, to accurately identify individuals in crowded

environments. The pre-trained model has been further refined using Oxford Town Centre Data to improve its accuracy and adaptability to different surveillance situations.

C Detailed CNN Model Architecture

Table 2: Model Summary Table

Layer	**Output Shape**	**Param #**
Conv2d-1	[-1, 32, 128, 128]	864
BatchNorm2d-2	[-1, 32, 128, 128]	64
ReLU-3	[-1, 32, 128, 128]	0
Conv2d-4	[-1, 16, 128, 128]	4,608
BatchNorm2d-5	[-1, 16, 128, 128]	32
ReLU-6	[-1, 16, 128, 128]	0
Conv2d-7	[-1, 32, 128, 128]	4,608
BatchNorm2d-8	[-1, 32, 128, 128]	64
ReLU-9	[-1, 32, 128, 128]	0
BasicBlock-10	[-1, 32, 128, 128]	0
Conv2d-11	[-1, 16, 128, 128]	4,608
BatchNorm2d-12	[-1, 16, 128, 128]	32
ReLU-13	[-1, 16, 128, 128]	0
Conv2d-14	[-1, 32, 128, 128]	4,608
BatchNorm2d-15	[-1, 32, 128, 128]	64
ReLU-16	[-1, 32, 128, 128]	0
BasicBlock-17	[-1, 32, 128, 128]	0
Conv2d-18	[-1, 16, 128, 128]	4,608
BatchNorm2d-19	[-1, 16, 128, 128]	32
ReLU-20	[-1, 16, 128, 128]	0
Conv2d-21	[-1, 32, 128, 128]	4,608
BatchNorm2d-22	[-1, 32, 128, 128]	64
ReLU-23	[-1, 32, 128, 128]	0
BasicBlock-24	[-1, 32, 128, 128]	0
Conv2d-25	[-1, 32, 64, 64]	9,216
BatchNorm2d-26	[-1, 32, 64, 64]	64
ReLU-27	[-1, 32, 64, 64]	0
Conv2d-28	[-1, 64, 64, 64]	18,432
BatchNorm2d-29	[-1, 64, 64, 64]	128
Conv2d-30	[-1, 64, 64, 64]	2,048
BatchNorm2d-31	[-1, 64, 64, 64]	128
ReLU-32	[-1, 64, 64, 64]	0
BasicBlock-33	[-1, 64, 64, 64]	0
Conv2d-34	[-1, 32, 64, 64]	18,432
BatchNorm2d-35	[-1, 32, 64, 64]	64
ReLU-36	[-1, 32, 64, 64]	0
Conv2d-37	[-1, 64, 64, 64]	18,432
BatchNorm2d-38	[-1, 64, 64, 64]	128
ReLU-39	[-1, 64, 64, 64]	0
BasicBlock-40	[-1, 64, 64, 64]	0
Conv2d-41	[-1, 32, 64, 64]	18,432
BatchNorm2d-42	[-1, 32, 64, 64]	64
ReLU-43	[-1, 32, 64, 64]	0
Conv2d-44	[-1, 64, 64, 64]	18,432
BatchNorm2d-45	[-1, 64, 64, 64]	128
ReLU-46	[-1, 64, 64, 64]	0

Layer	Output Shape	Param #
BasicBlock-47	[-1, 64, 64, 64]	0
Conv2d-48	[-1, 32, 64, 64]	18,432
BatchNorm2d-49	[-1, 32, 64, 64]	64
ReLU-50	[-1, 32, 64, 64]	0
Conv2d-51	[-1, 64, 64, 64]	18,432
BatchNorm2d-52	[-1, 64, 64, 64]	128
ReLU-53	[-1, 64, 64, 64]	0
BasicBlock-54	[-1, 64, 64, 64]	0
Conv2d-55	[-1, 64, 32, 32]	36,864
BatchNorm2d-56	[-1, 64, 32, 32]	128
ReLU-57	[-1, 64, 32, 32]	0
Conv2d-58	[-1, 128, 32, 32]	73,728
BatchNorm2d-59	[-1, 128, 32, 32]	256
Conv2d-60	[-1, 128, 32, 32]	8,192
BatchNorm2d-61	[-1, 128, 32, 32]	256
ReLU-62	[-1, 128, 32, 32]	0
BasicBlock-63	[-1, 128, 32, 32]	0
Conv2d-64	[-1, 64, 32, 32]	73,728
BatchNorm2d-65	[-1, 64, 32, 32]	128
ReLU-66	[-1, 64, 32, 32]	0
Conv2d-67	[-1, 128, 32, 32]	73,728
BatchNorm2d-68	[-1, 128, 32, 32]	256
ReLU-69	[-1, 128, 32, 32]	0
BasicBlock-70	[-1, 128, 32, 32]	0
Conv2d-71	[-1, 64, 32, 32]	73,728
BatchNorm2d-72	[-1, 64, 32, 32]	128
ReLU-73	[-1, 64, 32, 32]	0
Conv2d-74	[-1, 128, 32, 32]	73,728
BatchNorm2d-75	[-1, 128, 32, 32]	256
ReLU-76	[-1, 128, 32, 32]	0
BasicBlock-77	[-1, 128, 32, 32]	0
Conv2d-78	[-1, 64, 32, 32]	73,728
BatchNorm2d-79	[-1, 64, 32, 32]	128
ReLU-80	[-1, 64, 32, 32]	0
Conv2d-81	[-1, 128, 32, 32]	73,728
BatchNorm2d-82	[-1, 128, 32, 32]	256
ReLU-83	[-1, 128, 32, 32]	0
BasicBlock-84	[-1, 128, 32, 32]	0
Conv2d-85	[-1, 64, 32, 32]	73,728
BatchNorm2d-86	[-1, 64, 32, 32]	128
ReLU-87	[-1, 64, 32, 32]	0
Conv2d-88	[-1, 128, 32, 32]	73,728
BatchNorm2d-89	[-1, 128, 32, 32]	256
ReLU-90	[-1, 128, 32, 32]	0
BasicBlock-91	[-1, 128, 32, 32]	0
Conv2d-92	[-1, 64, 32, 32]	73,728
BatchNorm2d-93	[-1, 64, 32, 32]	128
ReLU-94	[-1, 64, 32, 32]	0
Conv2d-95	[-1, 128, 32, 32]	73,728
BatchNorm2d-96	[-1, 128, 32, 32]	256
ReLU-97	[-1, 128, 32, 32]	0
BasicBlock-98	[-1, 128, 32, 32]	0
Conv2d-99	[-1, 128, 16, 16]	147,456
BatchNorm2d-100	[-1, 128, 16, 16]	256

Layer	Output Shape	Param #
ReLU-101	[-1, 256, 16, 16]	0
Conv2d-102	[-1, 256, 16, 16]	294,912
BatchNorm2d-103	[-1, 256, 16, 16]	512
Conv2d-104	[-1, 256, 16, 16]	32,768
BatchNorm2d-105	[-1, 256, 16, 16]	512
ReLU-106	[-1, 256, 16, 16]	0
BasicBlock-107	[-1, 256, 16, 16]	0
Conv2d-108	[-1, 128, 16, 16]	294,912
BatchNorm2d-109	[-1, 128, 16, 16]	256
ReLU-110	[-1, 128, 16, 16]	0
Conv2d-111	[-1, 256, 16, 16]	294,912
BatchNorm2d-112	[-1, 256, 16, 16]	512
ReLU-113	[-1, 256, 16, 16]	0
BasicBlock-114	[-1, 256, 16, 16]	0
Conv2d-115	[-1, 128, 16, 16]	294,912
BatchNorm2d-116	[-1, 128, 16, 16]	256
ReLU-117	[-1, 128, 16, 16]	0
Conv2d-118	[-1, 256, 16, 16]	294,912
BatchNorm2d-119	[-1, 256, 16, 16]	512
ReLU-120	[-1, 256, 16, 16]	0
BasicBlock-121	[-1, 256, 16, 16]	0
AdaptiveAvgPool2d-122	[-1, 256, 1, 1]	0
Linear-123	[-1, 1,,]	257
Total params	**2,686,529**	
Trainable params	**2,686,529**	
Non-trainable params	**0**	

D Results of YOLO Model

The estimated average number of people in the YOLO model is shown in the graph below. The number of people varies considerably for New York. No consistent trend can be recognised here, as the average value rises and falls sharply and the peak is reached even during the lockdown. In London, the average number remains at roughly the same level over the entire period for all three observation periods. It does not fall significantly either during the lockdown or during the period in which the measures were relaxed. In New Orleans, the average number of people captured by the YOLO model is also fairly consistent, with some sudden sharp increases, for example on May 26, 2019. During the lockdown period, the average number is roughly at its lowest level. However, we would have expected these numbers to be even lower, as they also do not represent a significant decrease compared to the numbers before and after the lockdown.
Comparing the results with those of the CNN counting model, it is noticeable that the average values for all three cities are quite similar, but the curve peaks at different dates. It can therefore be concluded that even the YOLO model in its current form cannot be used reliably to assess social distancing measures. The fact that in all three cities the average number did not decrease significantly during the lockdown indicates that the model is not working optimally. The YOLO model was fine tuned with the Oxford Street dataset to teach the model how to deal with CCTV footage. One suggestion for improvement would be to significantly increase the number of such training images so that the model can learn more effectively how to recognize people in such images. Nevertheless, if the YOLO model works as intended, it offers the invaluable advantage of being able to calculate the distances between people, which may be an even more effective way of measuring social distance than simply counting the number of people. To do this correctly, we would need to incorporate a method that measures the distances between people regardless of their relative position in the image.

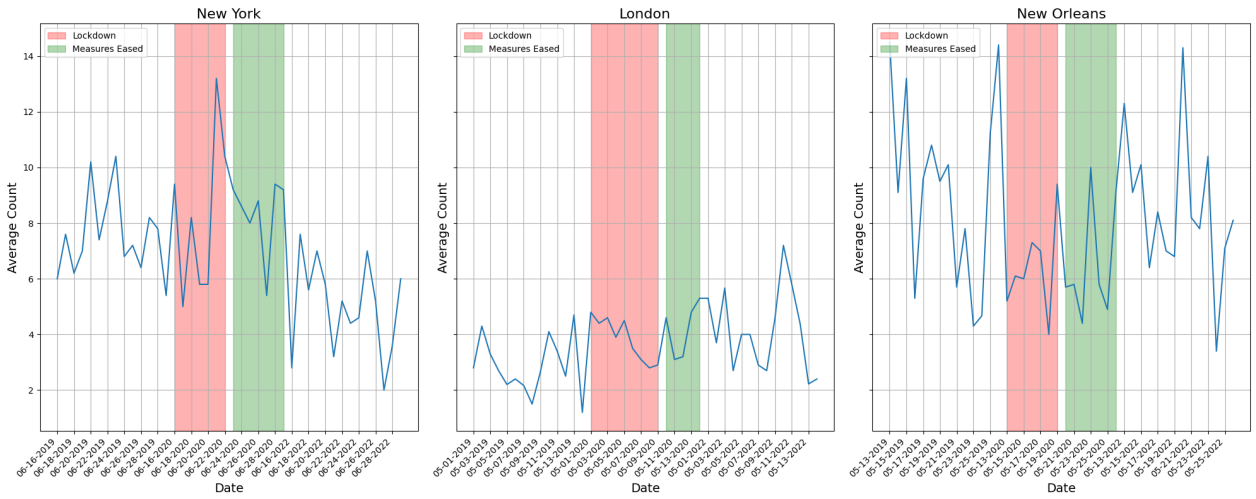

Figure 8: Estimated People Count with YOLO Model

E Prediction - Ground Truth Comparison

The Mean Absolute Error (MAE) of 6.43 indicates that, on average, the model's predictions deviate from the actual values by approximately 6.4 units. The Mean Squared Error (MSE) is significantly higher at 71.17, suggesting the presence of larger errors that are magnified by the squaring of error terms, which points to the poor predictions of outlier instances with more than 20 people in the image. The Root Mean Squared Error (RMSE) of 8.44 highlights that typical prediction errors have a substantial magnitude.

Metric	**Value**
Mean Absolute Error (MAE)	6.43
Mean Squared Error (MSE)	71.17
Root Mean Squared Error (RMSE)	8.44

Table 3: Performance Metrics of EartCam Analysis

Figure 9: London

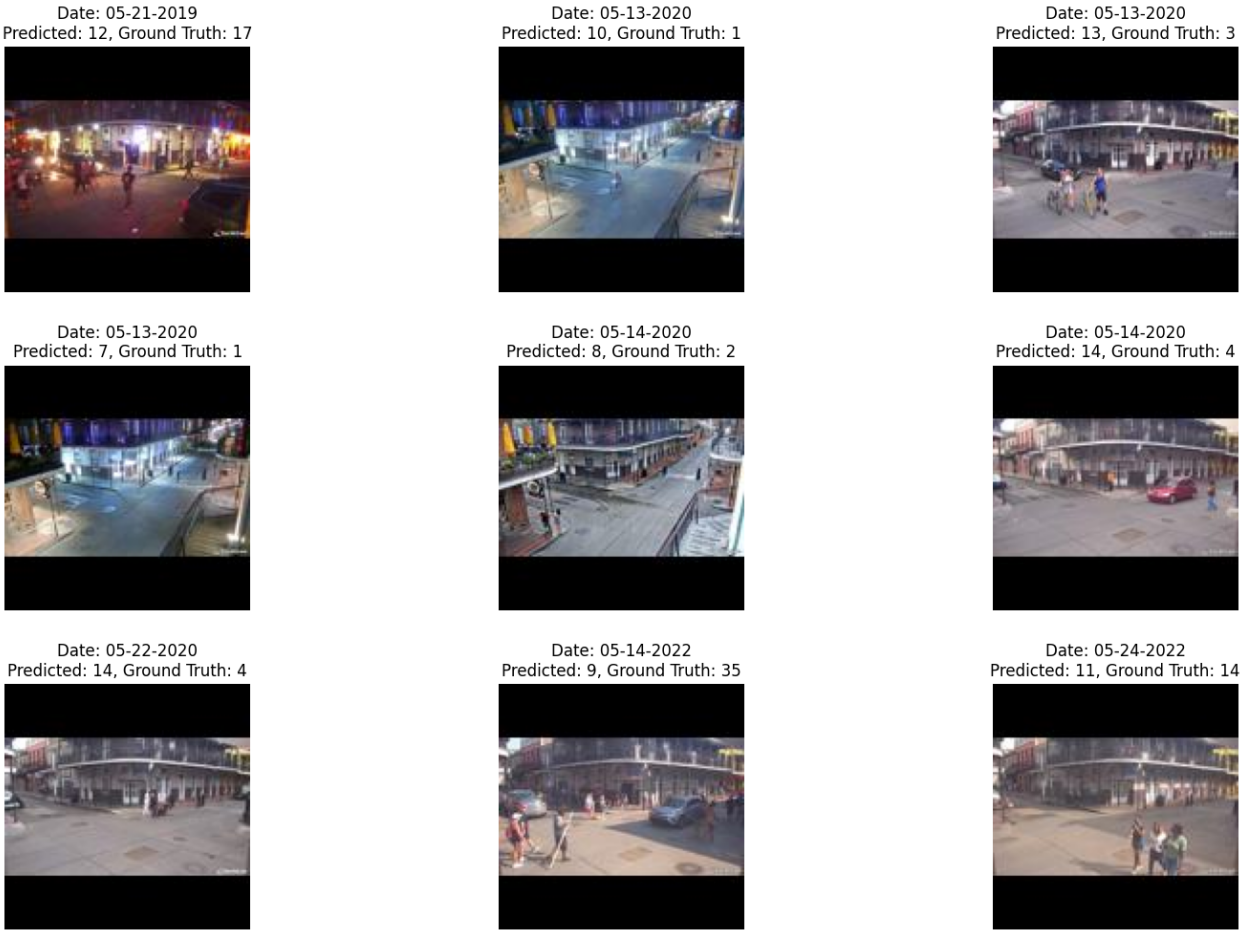

Figure 10: New Orleans

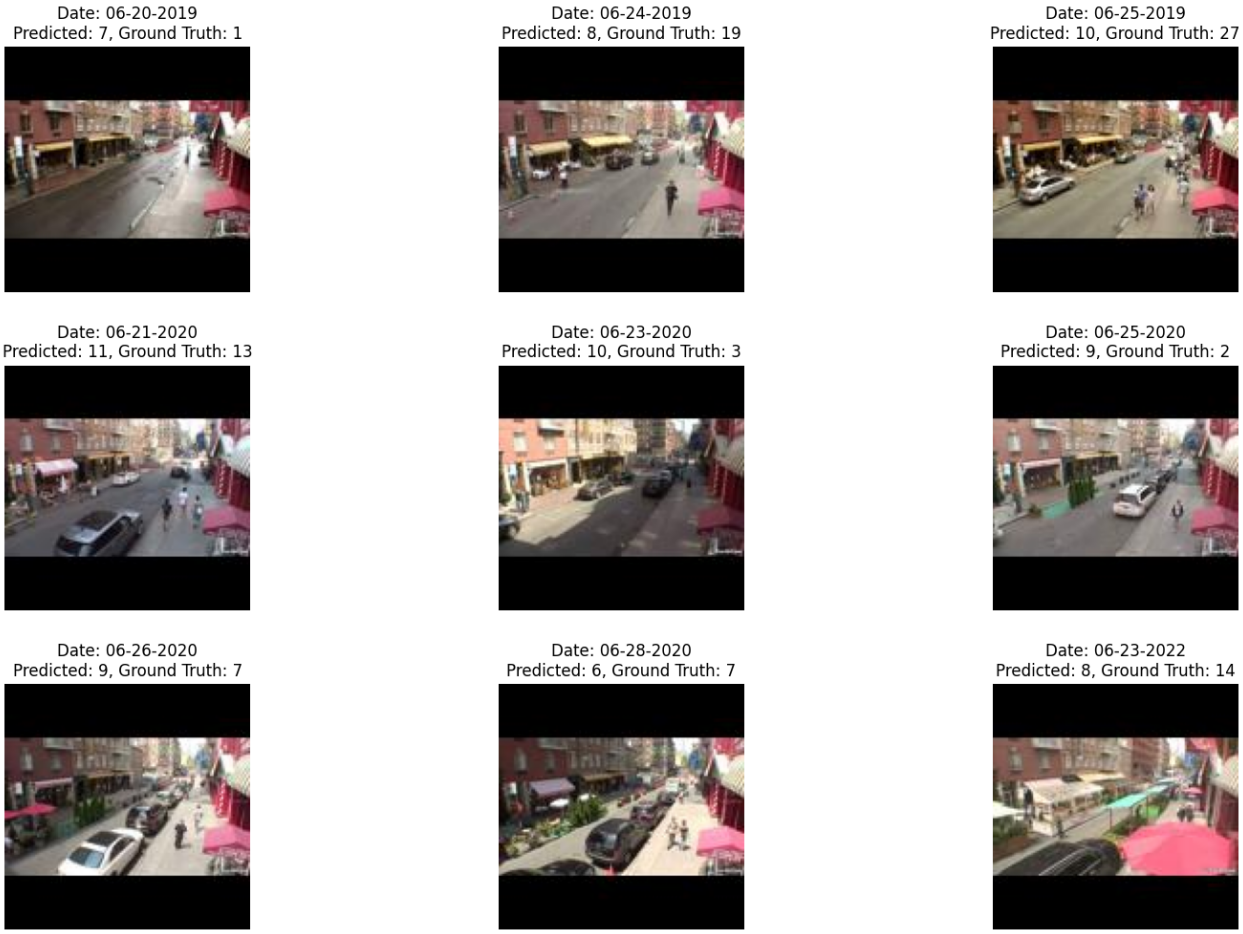

Figure 11: New York