Pedestrian Tracking with Computer Vision

1st Andrew Cvetko
School of Comp. Sci. & Eng.
University of New South Wales
Sydney, Australia
z333071@student.unsw.edu.au

2nd Benjamin McCoy
School of Comp. Sci. & Eng.
University of New South Wales
Sydney, Australia
z3464555@student.unsw.edu.au

3rd Chukwuemeka Eze
School of Comp. Sci. & Eng.
University of New South Wales
Sydney, Australia
z5185318@student.unsw.edu.au

4th George Maksour
School of Mathematics & Statistics
University of New South Wales
Sydney, Australia
z5254685@student.unsw.ed.au

5th Justin Clarke
School of Mathematics & Statistics
University of New South Wales
Sydney, Australia
z5259616@student.unsw.ed.au

***Abstract*—Computer vision has over the years grown to function effectively in a range of tasks including motion detection, identification and pattern analysis. One major improvement in this field is the tracking of pedestrians using deep learning techniques, in particular the convolutional neural network (CNN). CNNs have improved the performance of remote sensing image scene classification due to the powerful perspective of feature learning and reasoning. This paper applies computer-vision techniques across four videos from the segmenting and tracking every pixel data-set (STEP) to further analyse the behaviour of pedestrians. The paper focuses on the use of pre-processing techniques and human detection with the YOLO and DeepSORT models. These predictions are used across multiple frames to analyse the movement, numbers and behaviours of pedestrians. The results are tested against human-level performance to benchmark the developed methods. The tests yield an indication that the methods implemented within this paper are successful.**

***Index Terms*—Convolution, Computer Vision, Real-time tracking, pedestrians, pre-processing, YOLO, DeepSORT**

I. Introduction

Computer vision is an interdisciplinary field that creates techniques to derive information from images and videos. It involves the application of mathematical and computer science techniques in the generation of knowledge of this medium. Since the development of the perceptron [33] the information of images has been able to be quantified and approximated with custom functions. The field has had an exponentially growing number of applications. These include facial recognition [3], pedestrian tracking [42], traffic flow analysis [29] and medical resonance imaging (MRI) [17]. The field has grown with these applications as models and techniques have improved throughout time. One key model that has improved the range and applicability of computer vision are the advances in deep learning.

The deep learning (DL) computing paradigm has been deemed the gold standard in machine learning (ML) [1] and the advent of large convolutional neural networks have generated impressive results within the field of computer vision. Convolutional neural networks (CNN) are primarily used within literature to generate information on images. This is due to the inherent structure of these models that allow the input of multidimensional data. The CNN is a biologically inspired model that has 'neurons' that fire sequentially. These sequential neuron movements imitate the human brain's synapse patterns which create pattern recognition. Over the course of time with better processing technology and improvements to model learning, CNNs have grown in size with the number of 'neurons' in the billions. These large CNN models have progressively improved classification [21] on a variety of different data sets. These classification problems involve, wildlife classification [8], social network identification [2] and medical classification such as mammography [36]. Within the field of tracking videos, there has been much research into the classification of humans within an environment and analysis of their behaviours. These human tracking models aim to predict trajectories [49] of pedestrians which will improve safety for automated vehicles and other processes.

With these processes' the aim of this paper is to combine deep learning and pre-processing methods in order to create a useful predictive and tracking system for pedestrians. The rest of the paper is organised as follows. Section II focuses on the related works to this paper. Section III focuses on the materials and methods for the application of convolutional neural networks to track pedestrians. Section IV presents the results. Section V provides a discussion and section VI concludes the paper.

II. Literature Review

A. Importance of detection and tracking

The field of computer vision has evolved greatly through research and academia. In its early stages, facial recognition was simply done by identifying the distances between facial features to identify a face which while simplistic was not very effective. Over the years, computer vision has found use cases in transportation for self-driving cars, medicine for identifying tumours in medical imaging [38] and surveillance for facial recognition [18]. Pedestrian tracking has become an

essential task in the development of various technologies such as intelligent video surveillance, and traffic control systems, along with various levels of autonomous vehicles that are completely self-driving or simply for obstacle avoidance or automatic braking systems.

In the past, the field generally struggled with challenges such as occlusion, geometric and illumination variance, and other external noise. With the development of new algorithms and the rise of deep learning, computer vision has become increasingly accurate and fast to run with improvements in computing technology. Some issues that still remain are the need for human interpretation for labelling data which is time intensive and error-prone[4].

B. Object Detection

1) Early detection methods: The earliest proposed object detection method was the 2001 Viola Jones method [43] which was traditionally used for detecting faces. The model was efficient for its time but had a range of restrictions such as being unable to handle occlusion and only detected frontal upright faces. However, it did pave the way for future methods in the object detection field with technologies such as the 'integral image' [41].

Namely, the Histogram of Oriented Gradients (HOG) feature descriptors proposed by Dalal and Triggs [10] in 2005 built upon this technology to solve the problem of pedestrian detection. By counting gradient orientations in localised portions of an image, HOG was found to be particularly powerful in human detection as the method was invariant to geometric and photometric transformations. As HOG is a feature descriptor method, this could be used with any machine learning algorithm.

Felzenszwalb et al. [14] extended on this by proposing the Deformable Parts Model (DPM) in 2008 which could break down an object such as a pedestrian into multiple segments. By using HOG, objects could be broken down into a root filter which loosely approximated the entire object and multiple part filters at a higher resolution would cover smaller parts of the object. Then associated deformation models would score the locations and deformations of the parts relative to the root giving an overall confidence score for the entire object.

In the initial stages of the task, we began by creating an initial prototype using a HOG descriptor. Due to its simplicity and ease of use, it was a great starting point to develop our code and other image processing techniques. However, we found that the HOG method was somewhat inconsistent with the results which led us to test deep learning methods which are widely regarded to be more accurate and faster but require more computational power.

C. Deep learning algorithms

Deep learning[23] algorithms are computational models that are composed of multiple processing layers to learn representations of data with multiple levels of abstraction. Deep learning algorithms play an important role in object detection. However, the problem of tracking is more complex compared to the object detection problem. In Single Object Tracking (SOT) the appearance of the target is known as a prior, compared to Multiple Object Tracking (MOT), a detection step is necessary to identify the targets that can leave or enter the scene. Applying SOT models directly to solve MOT leads to poor results, with a high tendency to drift and create numerous ID switch errors, as such models usually struggle in distinguishing between similar-looking intra-class objects[9]. A series of algorithms specifically tuned to multi-target tracking have been developed in recent years to address these issues, together with a number of benchmark data sets and competitions to ease the comparisons between the different methods. CNN [9], [22] applied to the ImageNet[12] classification data set results to a remarkable performance and very popular in spatial pattern extraction. Recurrent Neural Networks (RNN)[35] like the Long Short-Term Memory (LSTM)[35] are used to process sequential data. Luo et al.[27] presented a comprehensive review on MOT for pedestrian tracking. The formulation unified the formulation of the MOT problem and described the main techniques used in the key steps of an MOT system. They presented deep learning as one of the premier future research directions since at the time it had only been employed by very few algorithms. Luo[27] presented a survey on Multiple Pedestrian Tracking, but they focused on RGB-D data.

However, the issue with these early deep learning algorithms was that they required two stages and multiple convolutions to perform. Although they were highly accurate, there required an immense amount of time to train and run. This led to the development of faster single-stage models such as You Only Look Once (YOLO)[30], SSD[26] and RetinaNet[24] which revolutionised fast and highly accurate deep learning for computer vision which is extremely important to pedestrian detection which usually has a high number of data points to test on.

D. Concept of object tracking in video frames

Segmentation and tracking, in particular, are among the most challenging problems in computer vision. Inferring the next direction of an object in a series of sequentially aligned images requires a deep understanding of motion estimation or optical flow[34] in a three-dimension (i.e., image/space plus the time between frames). In this paper, we try to simplify the definition of ideas built into the methods, which could be evaluated objectively. As direction and velocity are simply defined by the relationship between the points of an object at multiple points in time, tracking an object in a video, therefore, involves the analysis of a sequence of image frames for the purpose of establishing the location of a target object(s) over a series of image frames starting from the bounding box in the entry frame.

One of the problems that arise in the attempt to track an object over the course of a video is that a model must be able to re-identify an object throughout the course of multiple frames in a video. To solve this problem, most existing methods utilise a detection model which places all objects into a bounding box and an association model extracts re-

identification features from the image for future frames [50]. However, with two models, scalability issues arise and as such, some one-shot models have been produced which detect and re-identify objects. Voigtlaender et al. [44] added a branch to a Mask R-CNN to extract a re-ID feature for this purpose.

For our implementation, we decided to use a DeepSORT for the association model as we were not too concerned about the scalability of our code. Since the train and test only made up 2000 frames in total and using YOLO had already significantly improved our runtime, we stuck with a two model approach.

III. Methods

A. YOLO Model

We decided to implement the YOLO model for object detection. The YOLO model is a one-stage model allowing it to detect bounding boxes and class probabilities by looking at a frame once rather than separating out the bounding box and classification problems as two separate tasks as previous methods (see RCNN model family [16], [15], [32]). In doing so it reduces the complexity of the model and is computationally faster. As YOLO looks at the whole image rather than specific regions, it is able to pull in more contextual information reducing false positive classifications compared to other methods. This also aids in the models ability to generalise well and performance on out-of-sample testing [30]. Compared to other one-stage models, YOLO achieves similar performance results but with a material speed advantage[40], which is why it was chosen. In particular, we have implemented the YOLOv5 model with code sourced from [7].

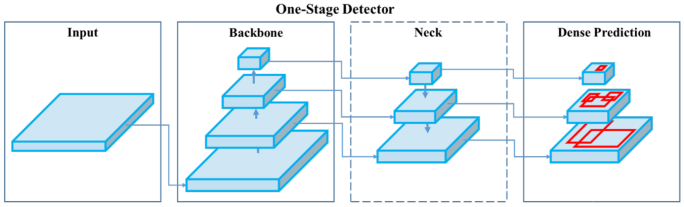

Fig. 1. High-level YOLOv5 architecture [6].

The model architecture consists of three main components [39]. A high-level and detailed summary of the architecture can be found in Figures 1 and 3 respectively.

a) Backbone: The backbone takes images and outputs images features using a convolutional neural network. The underlying architecture behind YOLOv5 is CSPDarkNet53 [31]. To improve model detection SiLU activation functions, skip connections, and batch normalisation was added to the model. The model uses cross-staged partial (CSP) networks which reduces computation time by 20% [46]. This is achieved by concatenating convolutional layers with copies of the previous layer reducing the number of duplicate gradients. The last layer includes fast spatial pyramid pooling (SPPF) which is applied using multiple pooled layers to produce a fixed length vector regardless of the input size which is robust to image deformations [19].

b) Neck: The neck is a series of layers which combines images for features for prediction. In our framework, a path augmentation network (PANet) is used [6]. The PAN layer integrates the features forming a feature pyramid at each convolutional layer. These are subsequently passed through a reversed DarkNet53 residual layers without shortcut connections [45]. This is used to enhance the process of image segmentation within the model by preserving spatial information which is achieved through bottom-up path augmentation and ROI align pooling. There is an inverse relationship between feature complexity and spatial resolution as we move through a network. Therefore, to combine semantically rich features with localisation information, PANet applies bottom-up path augmentation through the use of shortcut connections. PANet applies adaptive feature pooling using ROI align pooling on each feature map across all layers in the network using a concatenation of the layers [28].

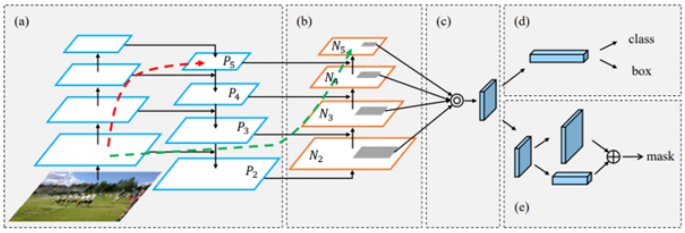

Fig. 2. PANet architecture. (a) Feature pyramid backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected [25].

c) Head: The head, or the dense prediction layer, takes the features from the neck and outputs the box and class predictions. Under the YOLO model, the input image is divided into an $S \times S$ grid where each cell in the grid is responsible for detecting the object if the centre of the object is present in the cell. Next, the cells predict B bounding boxes along with how confident they are that the object has been identified. In doing so it stores the following information: (x, y) being the coordinates of the midpoint of the identified bounding boxes, and (w, h) which are the relative width and height of the bounding box in the cell. The confidence for each bounding box is calculated as $Pr(Object) \times IOU^{Pred}_{Truth}$. The intersection over union is equal to 1 if the truth box corresponds to an instance in B. Finally, non-maximal suppression is applied to the remaining bounding boxes that meet the threshold requirement to produce a final bounding box for the object [30].

We used pre-trained YOLOv5 weights which were provided by [11] which were pre-trained on the COCO dataset of images focusing only on person and head classes. Images were pre-processed through a number of methods including mosiac, copy-paste, random affine transformations such as rotation, scale, translation and shear, mixup and HSV augmentation.

The model was optimised based on the following loss function:

$$Loss = \lambda_1 L_{Cls} + \lambda_2 L_{Obj} + \lambda_3 L_{Loc}$$

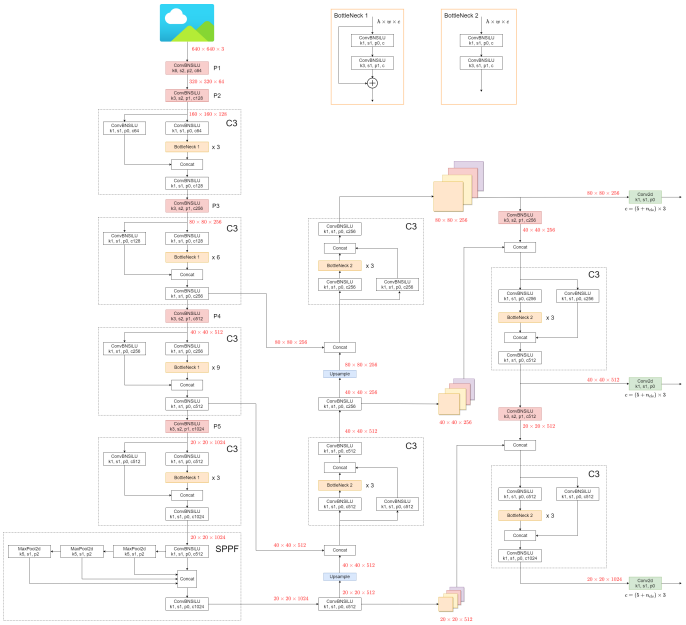

Fig. 3. Detailed YOLOv5 architecture [7].

where:

L_{Cls} is the classification loss measured through binary cross entropy.

L_{Obj} is the confidence of the object present measured through binary cross entropy. This is weighted based on the prediction layers in the network as $L_{Obj} = 4.0L_{Obj}^{Small} + 1.0L_{Obj}^{Medium} + 0.4L_{Obj}^{Large}$.

L_{Loc} is the bounding box regression loss.

The model was able to achieve a mean average precision of 89.6% on the trained COCO dataset.

B. DeepSORT

To complement the YOLO object detection algorithm, we paired this with DeepSORT to track identified pedestrians across frames. The DeepSORT algorithm builds upon the simple online and real-time tracking (SORT) framework. SORT consists of 4 pillars [37]. Firstly, objects must be detected which was attained through the YOLOv5 model. Next, estimation of the detected object's velocity across frames was calculated through the use of a Kalman filter. Thirdly, object association across frames is done to determine the bounding box in the subsequent frame by minimising the IOU distance from the detected object and the predicted bounding boxes from the previous step. This is completed using the Hungarian algorithm. Finally, the identity life-cycle maintains the IDs of all detected objects, creating new IDs for newly detected objects and destroying IDs if they are untracked due to low IOU.

DeepSORT adds to this by not only tracking objects through velocity but also through appearance. A neural network is trained offline to obtain well-discriminated feature embeddings on appearance. It applies a weighted average of the Mahalanobis distance and cosine distance metrics for training. The Mahalanobis distance measures the distance between a point to the mean of a distribution and is useful for short-term predictions based on object motion. The cosine distance considers appearance information which is useful under long-term occlusion scenarios or where objects have less motion. Finally, a cascading matching algorithm is employed to match objects between frames to account for partial occlusions on static scene geometry improving the overall robustness of results[48].

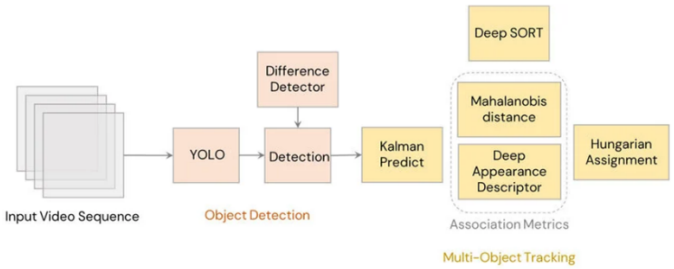

Fig. 4. DeepSORT architecture [37].

We choose the DeepSORT algorithm over other tracking algorithms such as Tracktor [5] or JDE [47] as it achieves similar accuracy results compared to other alternatives but at far lower computational cost [20]. Specifically, we used the StrongSort model outlined in [13] which provides some enhancements to the DeepSORT algorithm around the Kalman filter and matching logic.

The neural network used for person re-idenification is the OSNet proposed by [54], [53]. Typically, person re-identification faces two major challenges of intra and inter-class deviation. People are not uniform depending on the perspective they are viewed from (high intra-class variation) and secondly, different people have similar characteristics (e.g. similar clothing) making inter-class variation low. To solve this problem, [54], [53] implement an omni-scale network (OSNet) that extracts features at different scales and aggregates them together to perform the re-identification task.

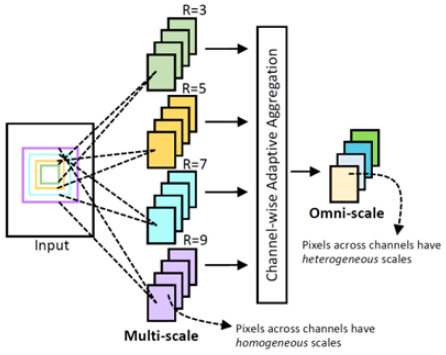

Fig. 5. Schematic of building block for OSNet [54].

OSNet uses a 'Lite $N \times N$' convolutional layer that performs a traditional 1×1 convolution followed by an $N \times N$ depth-wise convolution. Batch normalisation is then applied followed

by ReLU activation. These Lite convolutional layers form the basis of multi-scale feature learning. Different sized receptive fields pick up different feature sizes which are combined under a unified aggregation gate (AG). The AG is a mini-network comprised of a global average pooling layer and a multi-layer perceptron with ReLU activation at the first hidden layer followed by sigmoid activation. We used pre-trained weights on OSNet provided by the authors [52], [51] which achieved outstanding results across a number of different person re-identification datasets compared to other models [53].

stage	output	OSNet
conv1	128×64, 64 64×32, 64	7×7 conv, stride 2 3×3 max pool, stride 2
conv2	64×32, 256	bottleneck × 2
transition	64×32, 256 32×16, 256	1×1 conv 2×2 average pool, stride 2
conv3	32×16, 384	bottleneck × 2
transition	32×16, 384 16×8, 384	1×1 conv 2×2 average pool, stride 2
conv4	16×8, 512	bottleneck × 2
conv5	16×8, 512	1×1 conv
gap	1×1, 512	global average pool
fc	1×1, 512	fc
# params		2.2M
Mult-Adds		978.9M

Fig. 6. Architecture of OSNet [54].

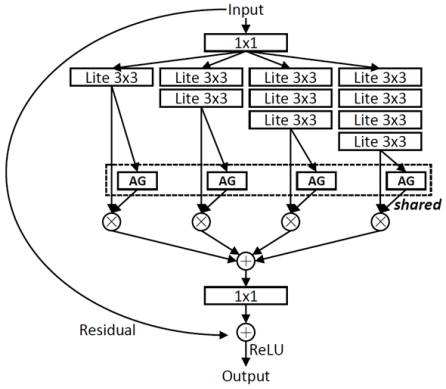

Fig. 7. OSNet bottleneck layer where AG is the aggregation gate. The first/last 1×1 layers are used to reduce/store feature dimension [54].

C. Process Analysis

There are three main sub-goals within this pedestrian detection task which are outlined below.

a) Track pedestrians: The first major sub-goal within this process is to track all pedestrians and calculate the bounding box for each frame within the videos. Using these bounding boxes the task is then extended to identifying the same pedestrians within the video and detecting the movement across frames. Then within multiple frames calculate the trajectory of the pedestrians and apply a unique label and colour to it.

b) Count pedestrians: The second goal is to count the number of unique pedestrians within the current frame of the video and then apply this to calculate the total number of unique pedestrians that are currently within that region. By extension, any user should be able to draw a box over any frame within the image and the number of humans within the video should be represented.

c) Analyse pedestrians: The final goal of this paper is to analyse some behaviours of the pedestrians. These behaviours include the count of how many walk within groups, when groups are formed or destroyed and when pedestrians are entering or leaving the scene.

These processes to achieve these goals will be represented throughout the methods section of this paper.

D. RankedList

Whilst the YOLO and DeepSORT models were an excellent start to achieve the project goals, an additional data structure was required to maintain information such as trajectories, groups and more, the solution was the RankedList class. The RankedList class contained a dataframe which was used to maintain a list of all detected objects in a video. Additionally, the RankedList class contained a dictionary of object trajectories and a list of groups. Finally the class maintained a list of rendered frames.

The Ranked List was incorporated into our project architecture by updating the ranked list each frame. For each frame our model would: 1. Read in the frame. 2. Get object information from YOLO Output. 3. Match objects in the frame to exsiting objects in RankedList. 4. Add any unmatched (new) objects to RankedList. 5. Remove any non-active objects from RankedList. 6. Update object trajectories. 7. Detect groups of pedestrians. 8. Update group information. 9. Render the frame with annotations.

Overall, including the RankedList allowed us to detect groups, plot object trajectories, detect objects entering or leaving the scene and aggregate object counts.

E. Data

The data used within this experiment is from the Segmenting and Tracking Every Pixel (STEP) benchmark which consists of two training videos and two test videos. The videos contain a similar theme of busy pedestrian places that are filmed at low/eye level. They contain a range of movements and interactions of humans. These include people walking in groups, busking, smoking and riding bicycles. The training videos provide values that represent categories including human, tree and shop. These videos are useful in providing a model to learn the features of some of the people within the video. The results of this process will investigate the detection, movement and grouping of humans within this video.

F. Analysis of pedestrians behaviours

A major subset of the task is to apply the models learnt information about a video and apply it to create more information about pedestrians, specifically groups of pedestrians. The major tasks involve identifying when a pedestrian joins or breaks apart from a group and when a pedestrian enters or leaves the video. The first problem will be solved using a set of decision equations to identify groups of people. These decision equations involve the distance of two pedestrians in an image, their relative trajectories and the comparison of the bounding

box size. These three tests will decide whether a pedestrian will become part of a group or leave a group depending on whether it is consistent with the decision process or not. The formulas for the decision equations are as follows:

a) Relative Gradients: Groups pedestrians with the same trajectories by brute force matching pedestrian ids that fit within a threshold.

$$\nabla g = |m_{p_2} - m_{p_1}|$$

$$d = \begin{cases} 1, & \text{if } \nabla g < G \\ 0, & \text{otherwise} \end{cases}$$

where G is a manually selected threshold and 1 represents inclusion into the set of grouped pedestrians

b) Distance of Centroids: Groups pedestrians within a specified distance ratio of the image frame.

$$D_p = \sqrt{(x_2^2 - x_1^2) + (y_2^2 - y_1^2)}$$

$$d = \begin{cases} 1, & \text{if } \nabla \frac{D_p}{D_{img}} < D \\ 0, & \text{otherwise} \end{cases}$$

Where D_{img} is the diagonal distance of the image. and 1 represents inclusion into the set of grouped pedestrians

c) Comparison of bounding boxes: Groups of pedestrians that have similar sized bounding boxes is used to represent whether objects are grouped in the foreground or background. As closer objects will inherently have larger bounding boxes.

$$A_i = W_{bb} * L_{bb}$$

For $A_1 \leq A_2$:

$$d = \begin{cases} 1, & \text{if } \nabla \frac{A_i}{A_{i+k}} > A \\ 0, & \text{otherwise} \end{cases}$$

where A is a threshold for the ratio of the two areas and if they're comparatively large enough the two pedestrians are further accepted in the set.

IV. Experimental Results

A. Experimental Setup

Our system, built on the original YOLOv5 model with mean average precision (mAP) of 89.6% on weight obtained by pre-learning on COCO (Common Object in Context) dataset, was experimented on CPU-based devices on datasets from Segmenting and Tracking Every Pixel (STEP) benchmark, using STEP-ICCV21-02 and STEP-ICCV21-01 for training and testing respectively. We then used visually identification to obtain counts of both correctly and incorrectly identified groups and individual pedestrians

B. Experimental Key Indicators

Our system is evaluated base on Precision, Recall and F1_score.

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{All\ Detections}$$

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{All\ Ground\ Truths}$$

$$F1\_score = \frac{2(Precision * Recall)}{(Precision + Recall)}$$

Precision refers to the percentage of all detection results that are correctly human while Recall indicates how well a positive prediction is made when a positive input is given. In other words, precision is the correct metric for evaluating our system. True Positive (TP) is a number of true/correctly detected objects (humanS). False Positive (FP) on the other hand is an incorrect prediction by the model which classes something that is not human as a human. False Negative (FN) means an object of another class that should have been detected but not detected as such.

C. Results

The two figures below show the results of the training and testing run of our system

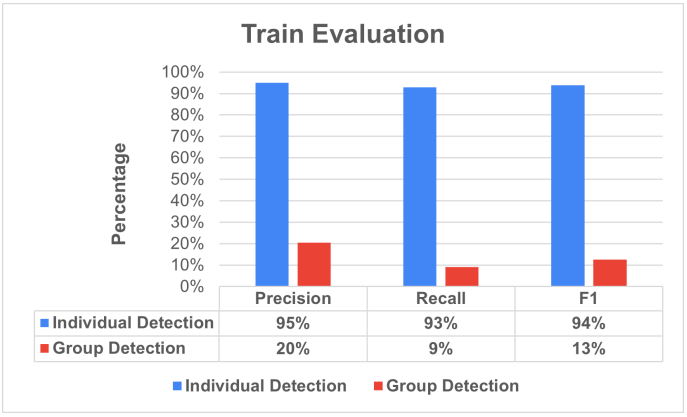

Fig. 8. Training Result

V. Discussion

1) Pedestrian Precision: The precision results on both train9 and test1 samples were extremely high. An example of this is shown in Figure 10 where all detections by the model were true pedestrians. The result was achieved by tuning the detection confidence threshold hyperparameter in the YOLO model. This value was increased from 20% to 50% meaning we were more confident in what was being detected was a true pedestrian.

There were very few cases of false positives. The main error case was the instances of the statutes near the building which were human-shaped. For example in Figure 11 ID

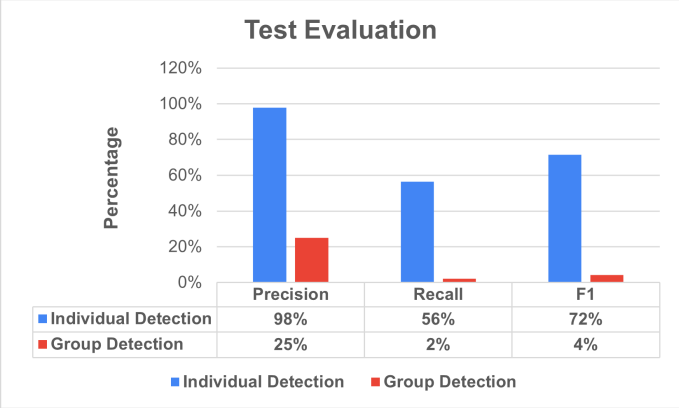

Fig. 9. Test Result

Fig. 10. High level of precision on frame 60 in test1.

26 was a statue which was identified as a pedestrian. The reason for the false positive is that the statue is human in shape and appearance making it difficult for the YOLO model to correctly differentiate given its similar feature profile. Potentially adding more cases of statues during the model training phase could have helped with this type of error.

Fig. 11. ID 26 - false positive on frame 119 in test1.

The second and smaller set of false positives were miscellaneous in nature. In Figure 12 ID 38 is a part of a building which has been misidentified as a pedestrian. This was only for a few frames throughout the video so its impact was minor. The result is likely due to general performance issues with the pre-trained YOLO model, which might have been fixed with more extensive training.

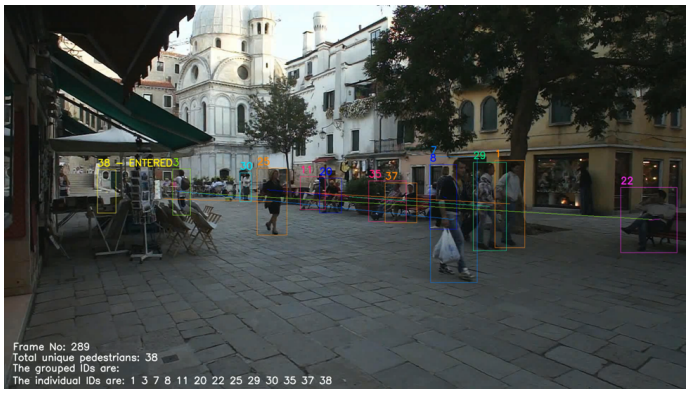

Fig. 12. ID 38 - false positive on frame 289 in test1.

The high precision score, however, came at the expense of lower recall as we were more discerning on correctly identifying pedestrians versus identifying all pedestrians. The recall was far lower on the test1 sample versus the train9 sample due to the number of pedestrians in the background versus the foreground. YOLO has a few limitations due to its model design mainly it doesn't detect small and/or grouped objects very well. The reason stems from breaking down the image into an $S \times S$ grid with each grid responsible for detecting a single object. If there are multiple objects in the grid this leads to poorer performance in these circumstances. With smaller grouped objects being more prevalent in test1, this was the main driver behind the lower recall scores when compared to the train9 sample.

2) Group Detection: We achieved decent group detection results on the train9 sample. In Figure 13 we picked up two groups 45, 47 and 48, 13, whilst in Figure 14 we missed detecting groups in the background of the image. The reason for the better results on train9 was that the groups were moving as well as in the foreground. Background objects were harder to detect due to the YOLO model limitations explained above. Similarly, stationary objects were also an issue as they had larger differences in the trajectory angle as their centroids were very stable and didn't trend in a particular direction resulting in instability in the trajectory angle.

3) Pedestrian Tracking: Our model did a great job of tracking pedestrians between frames. This is evident by plotting the number of times a pedestrian is seen in a frame throughout the video and is shown in Figure 15. There is a large number of pedestrians which have been successfully tracked over the majority of the video such as pedestrians 1, 7, 8, 25, and 29. This set of pedestrians is unique in that they are mostly in the foreground and are clearly visible throughout the whole video.

On the other hand, there is a long tail to this distribution where the number of pedestrians have been only tracked for a few frames. The majority of these are false positives which

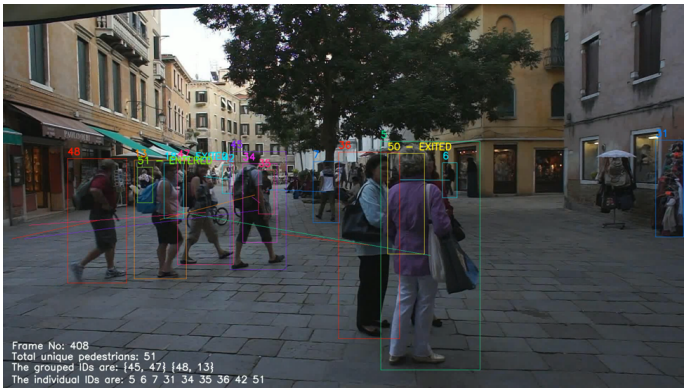

Fig. 13. Two groups detected in frame 408 on train9.

Fig. 14. Background groups missed. One at the table to the left of ID 7. Two sitting on the bench next to ID 20.

we have explained in the previous section. However, there is another group of pedestrians that have been re-identified under a new ID. This set is typically for pedestrians that have been occluded and in motion, which the DeepSORT algorithm found difficult.

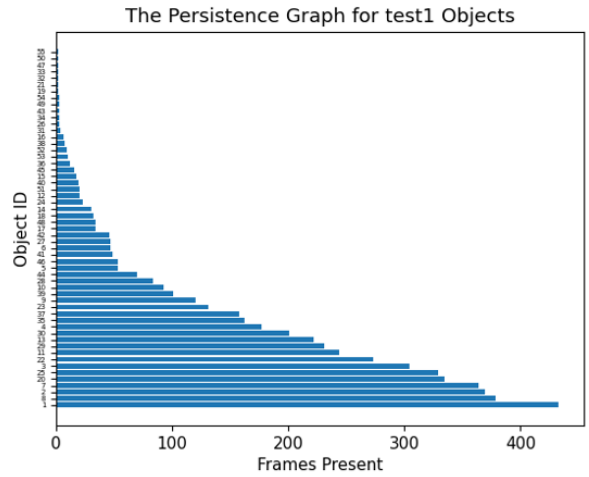

Fig. 15. Count of pedestrian IDs across all frames in test1.

In Figures 16 and 17 you can see the boy on the bike has been identified under different IDs as he passes behind the pedestrians in front of him as well as the tree. The DeepSORT algorithm would have struggled in these situations as the motion was fast making the distribution on the Kalman filter wider. Furthermore, the OSNet responsible for looking at appearance features would have been poor in this scenario as this area of the frame had poor illumination making it difficult to pick up uniquely identifying features.

Fig. 16. ID 36 - boy on bike identified on frame 244 in test1.

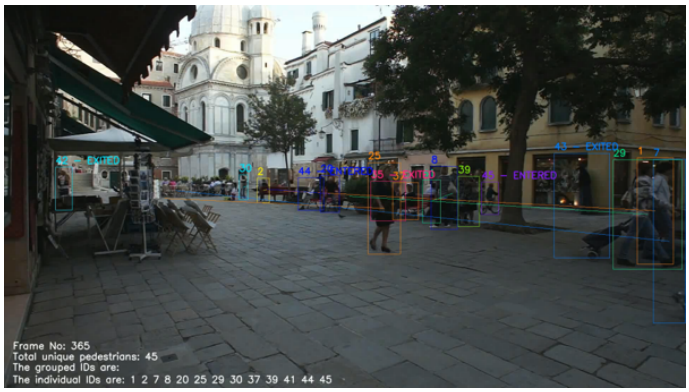

Fig. 17. ID 45 - boy on bike re-identified under a different ID on frame 244 in test1.

VI. Conclusion

Given the timeframe of only four weeks, it would have been hard to make a perfect product and as such, there is plenty of room for improvement and future work. The main issue encountered by this implementation was the tradeoff between precision and recall. This tradeoff is highly dependent on the task the pedestrian detection would be used for. The system worked extremely well on the foreground with a very high precision which would make it excellent for some sort of jaywalking detection / fining system where we want a minimal amount of false positives. On the other hand, if we were doing some sort of crowd counting task or self-driving task, it is absolutely imperative that we do not get any false negatives.

Another thing to also update would be our classification of groups. There was no real given definition of a group so we

had to define our own. The problem with our definition was that it struggled with stationary groups that were sitting as their trajectories and angles were highly inconsistent. Furthermore, if the footage was from a moving camera, this added even more variability to the movement of potential groups and as a result, the detection of groups suffered greatly.

The final improvement we would make would be to improve the matching system. Our implementation was decent in terms of matching objects across frames but whenever an object was occluded or had some extra noise like riding a bike, our model usually gave it a new ID. It was also very hard to evaluate these errors and labelling each frame one by one was a very time-consuming process. In the future, it might be better to only subsample one in every 5 frames as that was the frame criteria we determined was necessary for forming a group.

References

[1] Alzubaidi, L., Zhang, J., Humaidi, A.J., Al-Dujaili, A., Duan, Y., Al-Shamma, O., Santamaría, J., Fadhel, M.A., Al-Amidie, M., Farhan, L., 2021. Review of deep learning: Concepts, cnn architectures, challenges, applications, future directions. Journal of big Data 8, 1–74.

[2] Amerini, I., Li, C.T., Caldelli, R., 2019. Social network identification through image classification with cnn. IEEE access 7, 35264–35273.

[3] Balaban, S., 2015. Deep learning and face recognition: the state of the art. Biometric and surveillance technology for human and activity identification XII 9457, 68–75.

[4] Bernasco, W., Hoeben, E.M., Koelma, D., Liebst, L.S., Thomas, J., Appelman, J., Snoek, C.G.M., Lindegaard, M.R., 0. Promise into practice: Application of computer vision in empirical research on social distancing. Sociological Methods & Research 0, 00491241221099554. URL: https://doi.org/10.1177/00491241221099554, doi:10.1177/00491241221099554, arXiv:https://doi.org/10.1177/00491241221099554.

[5] Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B., 2016. Simple online and realtime tracking. CoRR abs/1602.00763. URL: http://arxiv.org/abs/1602.00763, arXiv:1602.00763.

[6] Bochkovskiy, A., Wang, C., Liao, H.M., 2020. Yolov4: Optimal speed and accuracy of object detection. CoRR abs/2004.10934. URL: https://arxiv.org/abs/2004.10934, arXiv:2004.10934.

[7] Broström, M., 2022. Real-time multi-camera multi-object tracker using yolov5 and strongsort with osnet. https://github.com/mikel-brostrom/Yolov5_StrongSORT_OSNet.

[8] Chen, R., Little, R., Mihaylova, L., Delahay, R., Cox, R., 2019. Wildlife surveillance using deep learning methods. Ecology and evolution 9, 9453–9466.

[9] Ciaparrone, G., Sánchez, F.L., Tabik, S., Troiano, L., Tagliaferri, R., Herrera, F., 2020. Deep learning in video multi-object tracking: A survey. Neurocomputing 381, 61–88.

[10] Dalal, N., Triggs, B., 2005. Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), pp. 886–893 vol. 1. doi:10.1109/CVPR.2005.177.

[11] deepakcrk, 2021. Head person detection model. https://github.com/deepakcrk/yolov5-crowdhuman.

[12] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L., 2009. Imagenet: A large-scale hierarchical image database, in: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255. doi:10.1109/CVPR.2009.5206848.

[13] Du, Y., Song, Y., Yang, B., Zhao, Y., 2022. Strongsort: Make deepsort great again .

[14] Felzenszwalb, P., McAllester, D., Ramanan, D., 2008. A discriminatively trained, multiscale, deformable part model, in: 2008 IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. doi:10.1109/CVPR.2008.4587597.

[15] Girshick, R., 2015. Fast r-cnn, in: 2015 IEEE International Conference on Computer Vision (ICCV), pp. 1440–1448. doi:10.1109/ICCV.2015.169.

[16] Girshick, R., Donahue, J., Darrell, T., Malik, J., 2014. Rich feature hierarchies for accurate object detection and semantic segmentation, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, pp. 580–587. doi:10.1109/CVPR.2014.81.

[17] Gozes, O., Frid-Adar, M., Sagie, N., Zhang, H., Ji, W., Greenspan, H., 2020. Coronavirus detection and analysis on chest ct with deep learning. arXiv preprint arXiv:2004.02640 .

[18] Hasan, I., Liao, S., Li, J., Akram, S.U., Shao, L., 2021. Generalizable pedestrian detection: The elephant in the room, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11328–11337.

[19] He, K., Zhang, X., Ren, S., Sun, J., 2014. Spatial pyramid pooling in deep convolutional networks for visual recognition. CoRR abs/1406.4729. URL: http://arxiv.org/abs/1406.4729, arXiv:1406.4729.

[20] Kanjee, R., 2020. Deepsort — deep learning applied to object tracking. URL: https://medium.com/augmented-startups/deepsort-deep-learning-applied-to-object-tracking-924f59f99104.

[21] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L., 2014. Large-scale video classification with convolutional neural networks, in: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pp. 1725–1732.

[22] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks, in: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (Eds.), Advances in Neural Information Processing Systems, Curran Associates, Inc. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[23] LeCun Y, B.Y., G.E., H., 2015. Deep learning. Nature 521, 436–444.

[24] Lin, T., Goyal, P., Girshick, R.B., He, K., Dollár, P., 2017. Focal loss for dense object detection. CoRR abs/1708.02002. URL: http://arxiv.org/abs/1708.02002, arXiv:1708.02002.

[25] Liu, S., Qi, L., Qin, H., Shi, J., Jia, J., 2018. Path aggregation network for instance segmentation. CoRR abs/1803.01534. URL: http://arxiv.org/abs/1803.01534, arXiv:1803.01534.

[26] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C., Berg, A.C., 2015. SSD: single shot multibox detector. CoRR abs/1512.02325. URL: http://arxiv.org/abs/1512.02325, arXiv:1512.02325.

[27] Luo, W., Zhao, X., Kim, T., 2014. Multiple object tracking: A review. CoRR abs/1409.7618. URL: http://arxiv.org/abs/1409.7618, arXiv:1409.7618.

[28] Miracle, R., 2020. Panet: Path aggregation network in yolov4. URL: https://medium.com/clique-org/panet-path-aggregation-network-in-yolov4-b1a6dd09d158.

[29] Peppa, M., Bell, D., Komar, T., Xiao, W., 2018. Urban traffic flow analysis based on deep learning car detection from cctv image series, in: SPRS TC IV Mid-term Symposium "3D Spatial Information Science–The Engine of Change", Newcastle University.

[30] Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A., 2015. You only look once: Unified, real-time object detection. CoRR abs/1506.02640. URL: http://arxiv.org/abs/1506.02640, arXiv:1506.02640.

[31] Redmon, J., Farhadi, A., 2018. Yolov3: An incremental improvement. CoRR abs/1804.02767. URL: http://arxiv.org/abs/1804.02767, arXiv:1804.02767.

[32] Ren, S., He, K., Girshick, R., Sun, J., 2017. Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 39, 1137–1149. doi:10.1109/TPAMI.2016.2577031.

[33] Rosenblatt, F., 1958. The perceptron: a probabilistic model for information storage and organization in the brain. Psychological review 65, 386.

[34] Royden, C., Moore, K., 2012. Use of speed cues in the detection of moving objects by moving observers. Vision Research 59, 17–24.

[35] Sak H., Senior A., B.F., 2014. Long short-term memory recurrent neural network architectures for large scale acoustic modeling , 248–255.

[36] Salama, W.M., Aly, M.H., 2021. Deep learning in mammography images segmentation and classification: Automated cnn approach. Alexandria Engineering Journal 60, 4701–4709.

[37] Sanyam, 2022. Understanding multiple object tracking using deepsort. URL: https://learnopencv.com/understanding-multiple-object-tracking-using-deepsort.

[38] Shia, W.C., Chen, D.R., 2021. Classification of malignant tumors in breast ultrasound using a pretrained deep residual network model and support vector machine. Computerized Medical Imaging and Graphics 87, 101829. URL: https://www.sciencedirect.com/science/article/pii/S0895611120301245, doi:https://doi.org/10.1016/j.compmedimag.2020.101829.

[39] Solawetz, J., 2020. Yolov5 new version - improvements and evaluation. URL: https://blog.roboflow.com/yolov5-improvements-and-evaluation/.

[40] Tan, L., Huangfu, T., Wu, L., Chen, W., 2021. Comparison of retinanet, ssd, and yolo v3 for real-time pill identification. BMC Medical Informatics and Decision Making 21, 324.

[41] Tian, D., Han, Y., Wang, B., Guan, T., Wei, W., 2021. A review of intelligent driving pedestrian detection based on deep learning. Computational Intelligence and Neuroscience 2021.

[42] Truong, M.T.N., Kim, S., 2019. A tracking-by-detection system for pedestrian tracking using deep learning technique and color information. Journal of Information Processing Systems 15, 1017–1028.

[43] Viola, P., Jones, M., 2001. Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, pp. I–I. doi:10.1109/CVPR.2001.990517.

[44] Voigtlaender, P., Krause, M., Osep, A., Luiten, J., Sekar, B.B.G., Geiger, A., Leibe, B., 2019. Mots: Multi-object tracking and segmentation URL: https://arxiv.org/abs/1902.03604, doi:10.48550/ARXIV.1902.03604.

[45] Wang, C., Bochkovskiy, A., Liao, H.M., 2020. Scaled-yolov4: Scaling cross stage partial network. CoRR abs/2011.08036. URL: https://arxiv.org/abs/2011.08036, arXiv:2011.08036.

[46] Wang, C., Liao, H.M., Yeh, I., Wu, Y., Chen, P., Hsieh, J., 2019a. Cspnet: A new backbone that can enhance learning capability of CNN. CoRR abs/1911.11929. URL: http://arxiv.org/abs/1911.11929, arXiv:1911.11929.

[47] Wang, Z., Zheng, L., Liu, Y., Wang, S., 2019b. Towards real-time multi-object tracking. CoRR abs/1909.12605. URL: http://arxiv.org/abs/1909.12605, arXiv:1909.12605.

[48] Wojke, N., Bewley, A., Paulus, D., 2017. Simple online and realtime tracking with a deep association metric. CoRR abs/1703.07402. URL: http://arxiv.org/abs/1703.07402, arXiv:1703.07402.

[49] Xue, H., Huynh, D.Q., Reynolds, M., 2018. Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction, in: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV), IEEE. pp. 1186–1194.

[50] Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W., 2021. FairMOT: On the fairness of detection and re-identification in multiple object tracking. International Journal of Computer Vision 129, 3069–3087. URL: https://doi.org/10.1007%2Fs11263-021-01513-4, doi:10.1007/s11263-021-01513-4.

[51] Zhou, 2022. Torchreidt. https://github.com/KaiyangZhou/deep-person-reid.

[52] Zhou, K., Xiang, T., 2019. Torchreid: A library for deep learning person re-identification in pytorch. arXiv preprint arXiv:1910.10093 .

[53] Zhou, K., Yang, Y., Cavallaro, A., Xiang, T., 2019a. Learning generalisable omni-scale representations for person re-identification. CoRR abs/1910.06827. URL: http://arxiv.org/abs/1910.06827, arXiv:1910.06827.

[54] Zhou, K., Yang, Y., Cavallaro, A., Xiang, T., 2019b. Omni-scale feature learning for person re-identification. CoRR abs/1905.00953. URL: http://arxiv.org/abs/1905.00953, arXiv:1905.00953.