Explorations of Multi-object Tracking in Videos

Jiaqi Wang, Yang Shi, Sili Wang

***Abstract*—This paper serves as the course project report of the CSCI 8000, Advanced topics in machine learning class by Dr. Sheng Li. In this project, a framework of multi-object tracking in video is proposed and implemented from base models, the RetinaNet [1] and the MemTrack network [2]. The base models are integrated under the constraints of our tracker selection strategies to perform tracking with not many excessive trackers, and decent tracking accuracy in multi-object tasks. Because of the nature of the framework, our method can also detect newly shown up objects even when it does not show up in the first frame. The Davis challenge dataset [3] is used for the testing purpose, and different training schedules are compared in the experiments. We also propose to further explore the mechanism of the image comparison for a better tracker management strategy and compare with the state-of-art multi-object tracking frameworks as a future direction of this work.**

***Index Terms*—Video, Multi-object tracking, RetinaNet, Image comparison, Memory mechanism, Attention.**

I. Introduction

Object tracking in videos is a classical computer vision problem. It consists of not only detecting the object in a scene but also recognizing the object in each and every frame, so as to distinguish it from other objects, both static and dynamic.

This research topic starts from image caption. Then it develops to single object tracking in videos. Tracking multiple objects in videos is an important problem in computer vision which has wide applications in various video analysis scenarios, such as visual surveillance, sports analysis, robot navigation and autonomous driving. However, there is not too much work about multi-object tracking.

Our project targets at multi-object tracking in videos. We propose a framework which combines RestinaNet as the detector and Memtrack as the tracker. When new objects appear in videos besides the initial components in the first frame in video, our framework can also capture them. RestinaNet is firstly proposed in paper [1]. They design a fully convolutional one-stage detector to evaluate the effectiveness of their experiment loss. The designed and trained simple dense detector is called RetinaNet. Trackers in this project are designed using Memtrack. A memory network can be considered as a combination of memory m and components named input feature map (I), generalization (G), output feature map (O) and response (R). The functions of each component are listed as follows:

1) I converts inputs to the internal feature representation.
2) G updates old memories given the new input.
3) O produces new outputs given the new input and the current memory state.
4) R converts the output into the response format desired

Given an input, the whole process of the model is shown in Fig. 1. The process happen in train and test time given a distinction between such phases, which means memories are also stored at test time, but the model parameters of I, G, O and R are not updated.

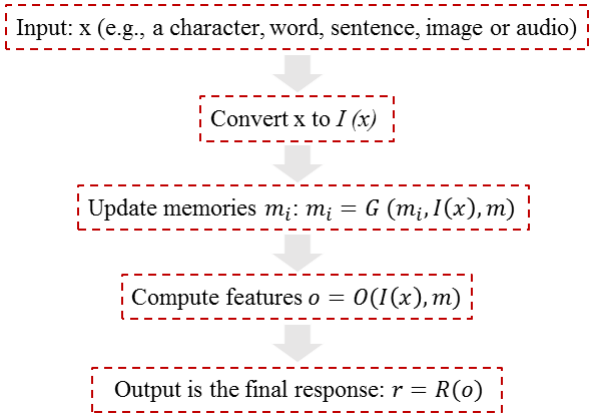

Fig. 1: The flow of the memory network model.

Main contributions of this work can be summarized as below:

1) Implement multi-object tracking in videos.
2) Capture the new objects showing up in videos.
3) Object tracking without initial localizations or labels.
4) Integration of tracker validation mechanism.

II. Related Work

A. Video Object Tracking

When we talk about computer vision, objective tracking has been actively studied for years. It started with single-objective tracking and has upgraded to multi-objective tracking with the increased requirement for complex conditions.

For several years, the most successful paradigm for this scenario has been to learn a model of the object's appearance in an online fashion using examples extracted from the video itself [4]. It owes in large part to the demonstrated ability of methods like TLD [5], Struck [6] and KCF [7]. The limitation using a pretrained deep convolutional networks is that the scarcity of supervised data and the constraint of real-time operation prevent the naive application of deep learning within this paradigm of learning a detector per video. To solve this, "shallow" methods are applied using the network's internal representation as features [8], [9]. Stochastic gradient descent(SGD) is also used in fine-tune multiple layers of the network [10]–[12].

From the other hand, we can follow the tracker types to review the previous work. The trackers can be classified as classification-based trackers, regression-based trackers and recurrent-neural-network trackers. Trackers for generic object

tracking often follows a tracking-by-classification methodology [13], [14], such as work in paper [12], [14]–[16]. Recent work in [17], [18] have attempted to treat tracking as a regression instead of classification problem. Also, work in [19], [20] attempted to treat tracking as a regression instead of classification problem.

B. Memory Mechanism

Memory networks are firstly proposed in paper [21]. It is used to solve the problem that most machine learning models lack an easy way to read and write to part of a (potentially very large) long-term memory component, and to combine this seamlessly with inference. Recent paper [22] implies that memory states help a lot about object template management. Based on NTM (Neural Turing Machine) proposed in paper [23], DNC (Different Neural Computer) was proposed to use different access mechanism to alleviate the memory overlap and interference problem in paper [24]. NTM is currently applied to one-shot learning with a new method for reading and writing memory in paper [25]. Later, a dynamic memory network is proposed in paper [2] to adapt the template to the targets appearance variations during tracking.

C. Attention Mechanism

Attention mechanisms in neural networks are about memory access. To make it easier, it can be described as something of a misnomer. Attention mechanisms has a wide application in machine translation [26], image caption [27], entailment Reasoning [28], speech recognition [29] and abstractive sentence summarization [30].

It is firstly proposed in paper [26]. The neural machine translation aims at building a single neural network that can be jointly tuned to maximize the translation performance. The model is shown in Fig.2. The graphical illustration of the model as below tries to generate *t*-th target word y_t given a source sentence $(x_1, x_2, ..., x_T)$. More details can be found in the paper mentioned above.

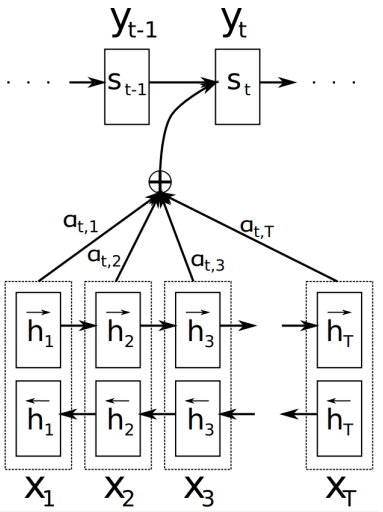

Fig. 2: Attention mechanism model.

Based on the traditional attention model, there are a lot of work studying deeper about its improvements and application. In paper [27], authors introduce an attention based model that automatically learns to describe the content of images, which is the first time that attention mechanism is applied to image caption. classified as soft attention and hard attention. Furthermore, researchers try to combine attention mechanism with other methods. In paper [31], authors propose a hierarchical attention network for document classification. In paper [32], authors present a novel model called attention-over-attention reader for the Cloze-style reading comprehension task. They aims to place another attention mechanism over the document-level attention, and induces "attended attention" for final predictions. In paper [33], an architecture based entirely on convolutional neural networks is proposed, where equip each decoder layer with a separate attention module is used.

III. Framework

In this project, our framework is based on two deep neural networks: RetinaNet [1] and MemTrack [2]. The object detection part uses the RetinaNet backbone and the object tracking part uses MemTrack. In this section, we first introduce structure of the two deep neural networks, and then show our framework and explain how we combine this two network to realize multiple object tracking.

A. RetinaNet

RetinaNet is a single, unified network composed of a backbone network and two task-specific subnetworks. The backbone is responsible for computing a convolutional feature map over an entire input image and is an off-the-self convolutional network. RetinaNet can be built on different backbone network which have different structure and different number of nodes. Backbone can influence the efficiency and accuracy of the RetinaNet.

In this project, we take use of one-stage RetinaNet to detect new objects. Fig. 3 shows the one-stage RetinaNet network architecture. It uses a Feature Pyramid Network (FPN) [34] backbone on top of a feed forward ResNet architecture [35] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d).

RetinaNet introduce the focal loss starting from the cross entropy (CE) loss for binary classification:

$$CE(p, y) = \begin{cases} -log(p) & if y = 1 \\ -log(1-p) & otherwise \end{cases} \quad (1)$$

In the above y in $[-1, 1]$ specifies the ground-truth class and p in $[0, 1]$is the models estimated probability for the class with label $y = 1$. For notational convenience, we define p_t as:

$$p_t = \begin{cases} p & if y = 1 \\ 1-p & otherwise \end{cases} \quad (2)$$

and rewrite $CE(p, y) = CE(p_t) = log(p_t)$.

One notable property of this loss, which can be easily seen in its plot, is that even examples that are easily classified p_t

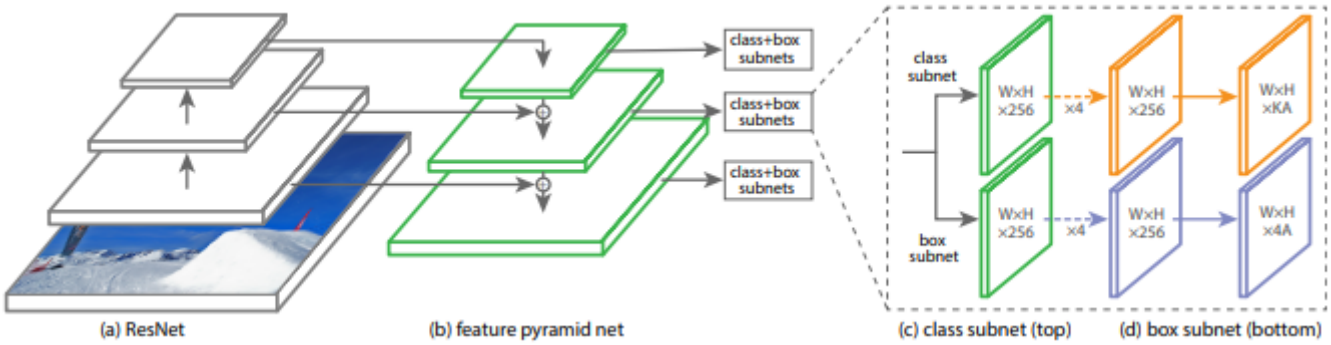

Fig. 3: One-stage RetinaNet network architecture

less than 0.5 incur a loss with non-trivial magnitude. When summed over a large number of easy examples, these small loss values can overwhelm the rare class.

A common method for addressing class imbalance is to introduce a weighting factor a, and then we have:

$$CE(pt) = -a_t log(pt) \tag{3}$$

A classification subnet and a box regression subnet are used together to get the object detection result. The classification subnet predicts the probability of object presence at each spatial position for each of the A anchors and K object classes. This subnet is a small FCN attached to each FPN level; parameters of this subnet are shared across all pyramid levels. The design of the box regression subnet is identical to the classification subnet except that it terminates in 4A linear outputs per spatial location. The output of RetinaNet should be a set of boxes which indicate the location of objects.

B. MemTrack

Dynamic memory network is a popular framework in object tracking. The main idea of dynamic memory network is to store some target information in an external memory and recall this information to maintain the variations of object appearance for template-matching. In the object tracking model based on detection where the targets information is stored in the weights of neural networks, the capacity of the model is fixed after offline training. However in a dynamic memory network, the capacity of the model can be changed by modified the size of external memory. Which is useful for memorizing long-term appearance variations. In Memtrack network present by Yang [2], they defined an initial template as reference. Since aggressive template updating is prone to overfit recent frames and the initial template is the most reliable one, they use the initial template as a conservative reference of the object and a residual template, obtained from retrieved memory, to adapt to the appearance variations. The image features are input into an attentional LSTM, which controls the memory reading and writing.

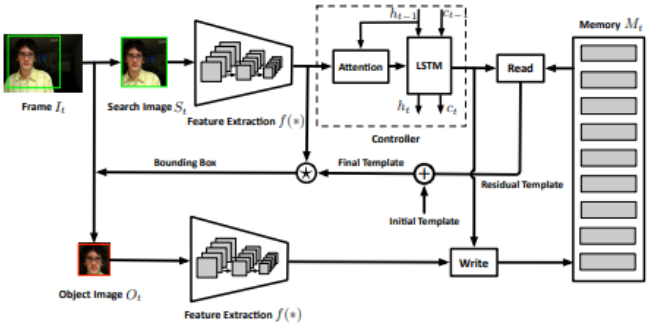

Fig. 4: The structure of Memtrack network.

Their algorithm including six parts: Feature Extraction, Attention Scheme, LSTM Memory Controller, Memory Reading, Residual Template Learning, and Memory Writing. The structure of Memtrack network is shown in fig.4. The green rectangle are the candidate region for target searching. The Feature Extractions for object image and search image share the same architecture and parameters. An attentional LSTM extracts the targets information on the search feature map, which guides the memory reading process to retrieve a matching template. The residual template is combined with the initial template, to obtain a final template for generating the response score. The newly predicted bounding box is then used to crop the objects image patch for memory writing.

In the feature extraction part, an Alex-like CNN is adopted. The searching space is larger than the template image. Attention scheme aim to make the input of LSTM concentrate more on the target object. To reduce the computational cost, an average pooling layer is applied on data before feeding it into LSTM. The LSTM control the length of memory, which can be write as:

$$a_t = \sum_{i=1}^{L} a_{t,i} f^*_{t,i} \tag{4}$$

Fig.5 shows the diagram of memory access mechanism. In the reading part, memory is retrieved by computing a weighted summation of all memory slots with a read weight vector, which is determined by the cosine similarity between a read key and the memory keys. In the writing part, there are three cases for memory writing: 1) when the new object template is not reliable (e.g. contains a lot of background), there is

no need to write new information into memory; 2) when the new object appearance does not change much compared with the previous frame, the memory slot that was previously read should be updated; 3) when the new target has a large appearance change, a new memory slot should be overwritten.

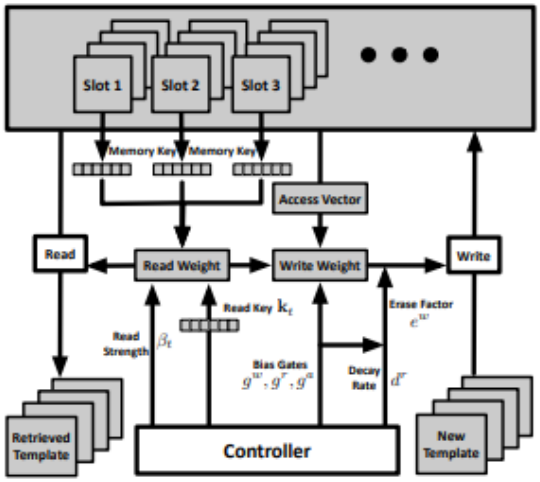

Fig. 5: Diagram of memory access mechanism.

C. Our current framework

Our current framework is shown in fig.6. In the object detection part, we use RPN feature to detect object, and comparing the feature from detected objects and the feature from template bank to give each object a label. This process will be applied on each single frames and thus we can know if there is a new object occurred. Then we use Memtrack structure to tracking each single object with independent boxes. In this way, we can tracking multiple object in same time. We also have some constrains to filter the new objects to avoid overlap or false alarm.

D. Image comparison

To detect the new object, we introduce image comparison algorithm. After feature extraction part, we compare the new detected objects with the objects templates in our database to decide whether if this object is a new object. We use a very simple built-in python algorithm to calculate the similarity of two pictures. The algorithm is simply read the RGB value of the picture and compare it point by point, the difference of the two pictures can be represent by quadratic sum of the two pictures. For the pictures with different size, we resize it first and then make the comparison. We use a threshold to define whether the new object is already existed on the database. If it is a new object, we will save that in the memory, if not we just discard it. With this image comparison algorithm, we can detect new object which is not occurred in the video at the beginning.

E. Location comparison

For multiple object tracking, one problem is that sometimes one single object might be detected more than one time. In this case, there will be a lot of boxes targeting the same object and the overlap between boxes will be large. To solve this problem we add some constrains to the boxes : i if the boundary of new boxes is too faraway from initial boxes, discard it; ii) if the center of new boxes have similar location with initial boxes, discard it; ii) if not, save these new boxes in memory. One hypothesis is that the initial boxes is reliable, so if the new boxes too far away from the initial boxes, we suppose they are false alarm, if the center location is too similar to the initial boxes we think they are overlapping boxes.

IV. Experiments

In this project, we finished the experiments for the three milestones of our project. We will first introduce the datasets that we use for the project first, and then discuss about the experiments we designed for the milestones.

A. Datasets

The Davis 2018 [3] is the testing dataset that we use for the project. In this dataset, 90 videos of motions of objects are provided, and some of them are for multi-object tracking. We employ the videos with quality of 480p for the fast image processing purpose. Besides, every video sequence come with the annotations of the objects for detection. It should be noticed that the Davis dataset originally does not consider the objects not shown up in the first frame, because of the nature of the Davis challenge. Fig. 7, 8 are from the same frame in a video from the Davis dataset. We observe a bench in the frame, while the bench is not labeled as an object since it does not show up in the first frame of this video. We take the annotation images, and use the smallest box that can contain all pixels in this image as the annotation box, and use the annotation box as the ground truth for the evaluation part.

Fig. 7: One frame from a video in Davis dataset.

Fig. 8: The annotation of the frame shown in Fig. 7.

In the evaluation part, as mentioned, some objects are not annotated. However, if the tracker recognizes the object, and the tracked areas obviously are associated with some certain objects, the tracked area is counted as an object. We summarize the evaluation metrics in Table I.

TABLE I: Metrics of the evaluation in the proposed framework

Metric	Definition
Total Objects (O)	The objects in the video sequence.
Correct Trackings (C)	The tracking result that covers the major part of the object.
Missed Objects (M)	Objects that major body is not tracked.
False Positives (FP)	Trackings that does not cover any object.
Mean Detection Time (T)	Mean tracking time for each frame.

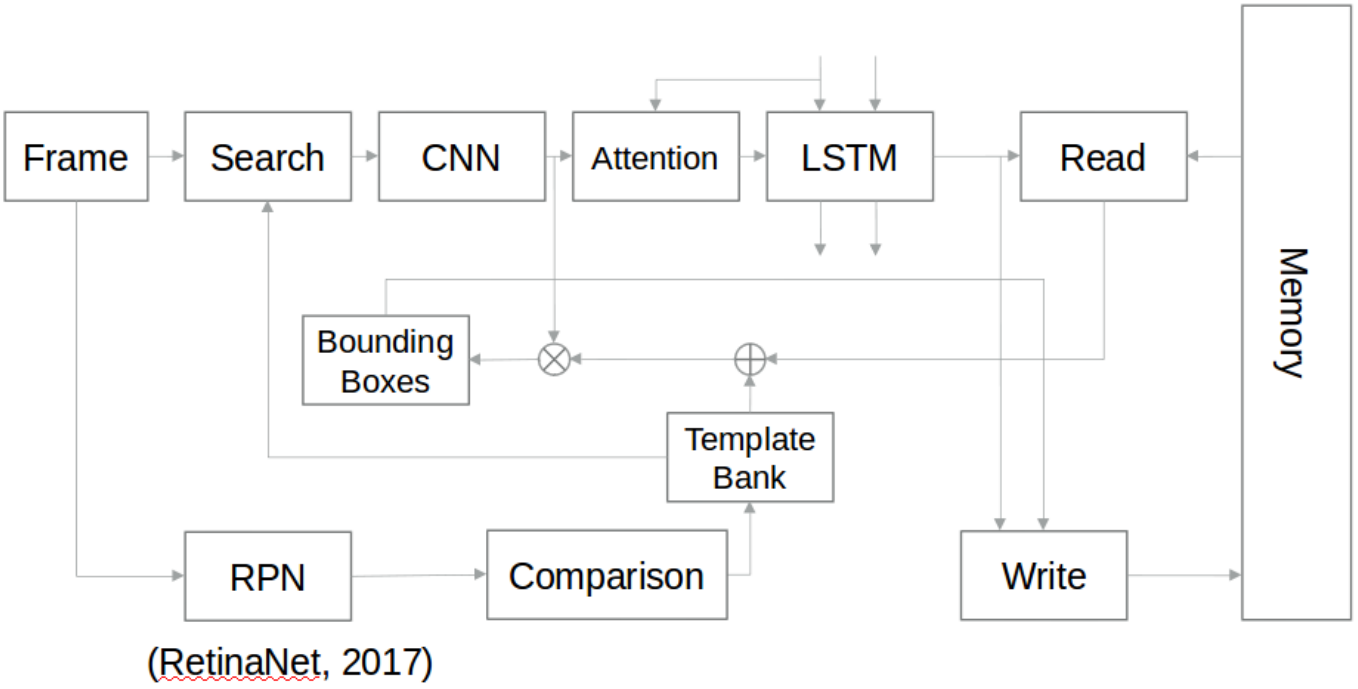

Fig. 6: One-stage RetinaNet network architecture

In the following experiments, pretrained models are used in RetinaNet. We leverage the default settings and backbones for RetinaNet in the pretrained models to get the best out of the model in object detection tasks in this framework. The model contains a backbone of ResNet , and pretrained with MS COCO [36]. Because of the nature of the MemTrack network, we first evaluate the performance of the proposed framework that the pretrained RetinaNet only detect the objects on the first frame, and only use this as the annotations for the multi-object tracking in MemTrack. Then we add the constraints proposed in Section Framework and test the performance of the framework in multi-object tracking tasks. Finally, we also retrained the RetinaNet with the Davis dataset, to compress some false positives and compare the performance with the pretrained model. Since the pretrained model is trained from MS COCO, dataset, the distribution should be different from Davis dataset. We randomly selected 10 videos for testing through the three experiments to keep a better consistency and provide a better comparison on the performances.

B. Tracking with the First Frame Detected with Pretrained RetinaNet

We start from using the pretrained RetinaNet only in the first frame of the videos. The evaluation result is shown in Table II.

This result shows that our integration of the framework is successful, yet for the multi-object tracking task, the framework does not fulfill the job directly. We can see on the video #2, 6, 7, 10 since there are not a lot of objects, the tracker perform well with a good detection rate and low false positives or missed objects in these detections. Only in some situations when the object is not shown up in the first frame, the objects are not tracked.

TABLE II: Experiment result of tracking with only the first frame detected with pretrained RetinaNet.

Video Index	O	C	M	FP	T
1	3	3	2	2	0.94s
2	2	1	1	0	0.40s
3	5	3	3	0	0.99s
4	3	2	3	2	0.65s
5	5	3	4	4	1.47s
6	2	2	1	1	0.66s
7	1	1	0	1	0.39s
8	4	2	2	0	0.68s
9	6	0	-	-	-
10	2	2	0	0	0.67s
Total	33	19	22	10	-

However, this also lead to missing objects in more complicated situations. For example, in video #8, only 2 out of 4 objects are tracked. The reason is that these objects do not show up in the first frames. In the more complicated multi-object tracking tasks in video #1, 3, 4, 5, this framework cannot track all objects and with a high missing or false positive rate. In these situations, the tracker can initially track the objects, but later when the objects deform, or shift rapidly, the tracker cannot follow the objects in this situation, and especially when objects overlap, the trackers have a high chance of lose the object. In video #9, the framework even fails to track since it cannot detect any objects in the first frame. This result has put us to the work in using RetinaNet to detect all frames in a video sequence. We use this information to compare with the trackers to decide if a new tracker is necessary, or if the current tracker is already not valid.

C. Tracking with All Frame Detected with Pretrained RetinaNet

The evaluation of the experiment that we use RetinaNet for all frames in the tracking task is shown in Table III.

TABLE III: Experiment result of tracking with all frames detected with pretrained RetinaNet.

Video Index	O	C	M	FP	T
1	3	3	3	2	5.89s
2	2	2	1	1	5.37s
3	5	5	1	2	5.76s
4	3	3	1	7	5.21s
5	5	4	2	7	5.71s
6	2	2	0	0	5.20s
7	1	1	0	0	4.65s
8	4	4	0	0	6.01s
9	6	3	3	1	4.76s
10	2	2	0	0	5.16s
Total	33	29	11	22	5.31s

From this evaluation result, it should be noticed that the detection rate of objects obviously increased. Out of 33 objects, 29 are tracked. The reason behind this rise is that our framework is able to detect the objects not shown up in the first frame, and track them. Especially in the video #1, 2, 3, 8, the trackers outperform the previous version.

We also see some drawbacks in this version. First of all, the false positives increased. in this framework, the number of untracked objects decreased, but since the RetinaNet is pretrained in another dataset, the false positives increased a lot. Another issue of this framework is that the averaged tracking time in each frame is too long for the video object tracking tasks. For this issue, it is because of the excessive trackers that generated by the framework. Since the more trackers the more computation is needed for a frame, reducing the excessive trackers is needed. This issue is still under our investigation currently.

To reveal the extreme performance of our proposed framework, we retrain the RetinaNet with the Davis dataset to see if the false positive would drop, because that our assumption is based on that the pretrained model detects some objects that are not the real objects, and a smaller training dataset can reduce false positives lead by the pretrained model.

D. Tracking with All Frame Detected with Retrained RetinaNet

We retrained RetinaNet with the same backbone network of the pretrained model, which is ResNet. The default settings of the training is employed. The batch size is 1, and the epochs is 50. To accelerate the training process, we used NVIDIA GeForce GTX 1080 Ti GPU for the training process. The retrained model is then applied in our proposed framework and the evaluation result is shown in Table IV

TABLE IV: Experiment result of tracking with all frames detected with retrained RetinaNet.

Video Index	O	C	M	FP	T
1	3	3	0	0	2.07s
2	2	1	1	0	1.61s
3	5	3	2	2	1.89s
4	3	2	1	3	1.11s
5	5	1	4	0	0.45s
6	2	2	1	0	0.90s
7	1	1	0	0	0.72s
8	4	2	2	0	1.39s
9	6	5	2	0	2.31s
10	2	1	1	0	0.46s
Total	33	21	14	5	1.25s

The experiment result shows that with the retrained model, the false positive dropped obviously. This confirms with our assumption that the false positives are mostly generated from the pretrained model that some complicated non-objects are considered as objects by the pretrained model. This does not happen in the retrained model, and the false positive dropped to only 5 throughout the videos. But since the annotation is not complete, as some of the objects are not labeled in our training data, some objects are missing. Even though, the missing objects are still a lot better than the first experiment, since it does not have the issue of losing trackers. The performance difference in the video #9 shows especially the improvement of our models. The content of this video is that several goldfishes In the second experiment, since the pretrained model does not have the samples of goldfish, the detection of the goldfish failed, and led to the low performance in this video.

The evaluations of our experiments for the framework that we propose in this course project show that our framework performs multi-object tracking without annotations with a decent detection rate, and it also detects the objects shown up in later frames of the videos. However, there are also some current limitations from our framework. We will introduce the limitations in the next section.

V. Conclusion and Future Plan

In this project, a multi-object tracking framework for videos that tracks the objects show up in any frame is proposed, implemented and evaluated. The framework integrates the RetinaNet as the detector, and the MemTrack network as the tracker. The comparison of trackers and location constraints are also introduced in the integration for a better performance of our framework.

From our evaluation, we find that our framework is able to produce accurate tracking for the objects in videos. However, there are still some issue with our framework, and we plan to polish the framework to a better performance.

The current issues of the framework includes three components. The computational speed is slow, and cannot meet the requirement of fast tracking in videos; the current image comparison mechanism is simple yet effective, while an automated thresholding technique can potentially benefit this process; the current dataset is not a major dataset for the multi-object tracking task, as it does not label the objects shown up in the later frames in videos.

To address the mentioned issues, we have plans for the future work. We plan to explore more on the object detection network, especially for the lightweight ones. The current RetinaNet is accurate, but not fast enough for the task. We will also compare with popular networks such as YOLO to find more to learn from. To address the issue in image comparison, we plan to use deep neural networks to extract features in the images, and compare the extracted features. The detailed fashion is yet to be decided, but we will transfer the current one into an automated and more accurate one. We will also look into the state-of-art multi-object tracking datasets to provide better evaluations for our proposed framework.

Multi-object tracking is a popular and interesting topic in computer vision field due to the vast application areas. We believe that with more development our proposed framework will be able to achieve a more accurate and faster tracking in this task.

VI. My work and contribution

Firstly, I am grateful for all the efforts of everyone in the team. Yang is a great fellow to work with and I we enjoyed a nice time together. My work and contribution are required to list as below:

1) I took charge of the third and fourth sections.
2) I proposed to solve a multi-objective tracking problem. The main targets consist of multi detection, new objective detection and better image comparison mechanism.
3) I led the team to propose such a framework using RetinaNet and MemTrack.
4) I chose the dataset after comparison. Then I finished most parts of the algorithm implement.
5) Yang and I run the experiments and analyze the results together.
6) Yang and I discuss the future work and target at ICCV 2019.

References

[1] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," *IEEE transactions on pattern analysis and machine intelligence*, 2018.

[2] T. Yang and A. B. Chan, "Learning dynamic memory networks for object tracking," *arXiv preprint arXiv:1803.07268*, 2018.

[3] "The 2018 DAVIS challenge on video object segmentation," Mar. 2018. [Online]. Available: http://arxiv.org/abs/1803.00557

[4] A. W. M. Smeulders, D. M. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah, "Visual tracking: An experimental survey," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 7, pp. 1442–1468, 2014.

[5] Z. Wu, "Multi-object tracking," in *Human Re-Identification*. Springer, 2016, pp. 23–26.

[6] S. Hare, S. Golodetz, A. Saffari, V. Vineet, M.-M. Cheng, S. L. Hicks, and P. H. S. Torr, "Struck: Structured output tracking with kernels," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 10, pp. 2096–2109, 2016.

[7] J. F. Henriques, R. Caseiro, P. Martins, and J. Batista, "High-speed tracking with kernelized correlation filters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 3, pp. 583–596, 2015.

[8] C. Ma, J.-B. Huang, X. Yang, and M.-H. Yang, "Hierarchical convolutional features for visual tracking," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3074–3082.

[9] M. Danelljan, G. Hager, F. Shahbaz Khan, and M. Felsberg, "Convolutional features for correlation filter based visual tracking," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 58–66.

[10] N. Wang, S. Li, A. Gupta, and D.-Y. Yeung, "Transferring rich feature hierarchies for robust visual tracking," *arXiv preprint arXiv:1501.04587*, 2015.

[11] L. Wang, W. Ouyang, X. Wang, and H. Lu, "Visual tracking with fully convolutional networks," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3119–3127.

[12] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 4293–4302.

[13] Z. Kalal, K. Mikolajczyk, J. Matas, and Others, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, p. 1409, 2012.

[14] N. Wang, J. Shi, D.-Y. Yeung, and J. Jia, "Understanding and diagnosing visual tracking systems," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3101–3109.

[15] S. Hong, T. You, S. Kwak, and B. Han, "Online tracking by learning discriminative saliency map with convolutional neural network," in *International Conference on Machine Learning*, 2015, pp. 597–606.

[16] K. Zhang, Q. Liu, Y. Wu, and M.-H. Yang, "Robust visual tracking via convolutional networks without training," *IEEE Transactions on Image Processing*, vol. 25, no. 4, pp. 1779–1792, 2016.

[17] D. Held, S. Thrun, and S. Savarese, "Learning to track at 100 fps with deep regression networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 749–765.

[18] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr, "Fully-convolutional siamese networks for object tracking," in *European conference on computer vision*. Springer, 2016, pp. 850–865.

[19] S. E. Kahou, V. Michalski, and R. Memisevic, "RATM: recurrent attentive tracking model," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. Workshops*, 2015, pp. 1613–1622.

[20] Q. Gan, Q. Guo, Z. Zhang, and K. Cho, "First step toward model-free, anonymous object tracking with recurrent neural networks," *arXiv preprint arXiv:1511.06425*, 2015.

[21] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *CoRR*, vol. abs/1410.3916, 2014. [Online]. Available: http://arxiv.org/abs/1410.3916

[22] T. Yang and A. B. Chan, "Recurrent filter learning for visual tracking," *arXiv preprint arXiv:1708.03874*, 2017.

[23] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[24] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou, and Others, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, p. 471, 2016.

[25] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "One-shot learning with memory-augmented neural networks," *arXiv preprint arXiv:1605.06065*, 2016.

[26] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[27] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[28] T. Rocktäschel, E. Grefenstette, K. M. Hermann, T. Kočiskỳ, and P. Blunsom, "Reasoning about entailment with neural attention," *arXiv preprint arXiv:1509.06664*, 2015.

[29] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, "Attention-based models for speech recognition," in *Advances in neural information processing systems*, 2015, pp. 577–585.

[30] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.

[31] Z. Yang, D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy, "Hierarchical attention networks for document classification," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 1480–1489.

[32] Y. Cui, Z. Chen, S. Wei, S. Wang, T. Liu, and G. Hu, "Attention-over-attention neural networks for reading comprehension," *arXiv preprint arXiv:1607.04423*, 2016.

[33] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y. N. Dauphin, "Convolutional sequence to sequence learning," *arXiv preprint arXiv:1705.03122*, 2017.

[34] T.-Y. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie, "Feature pyramid networks for object detection." in *CVPR*, vol. 1, no. 2, 2017, p. 4.

[35] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[36] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, *Microsoft COCO: Common Objects in Context*. Cham: Springer International Publishing, 2014, vol. 8693, ch. 48, pp. 740–755. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10602-1˙48