Machine Learning with Applications 12 (2023) 100460

Contents lists available at ScienceDirect

Machine Learning with Applications

journal homepage: www.elsevier.com/locate/mlwa

ViCTer: A semi-supervised video character tracker

Zilinghan Li [a], Xiwei Wang [b], Zhenning Zhang [a], Volodymyr Kindratenko [a,b,c,*]

[a] Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana, 61801, IL, USA
[b] Department of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign, Urbana, 61801, IL, USA
[c] National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana, 61801, IL, USA

ARTICLE INFO

Dataset link: CHARACTER FACE IN VIDEO (Original data), Research Code (Original data)

Keywords:
Face recognition
Multiple object tracking
Semi-supervised learning

ABSTRACT

Video character tracking problem refers to tracking certain characters of interest in the video and returning the appearing time slots for those characters. Solutions to this problem can be applied in various video-analysis-related areas, such as movie analysis and automatic video clipping. However, there are very few researches investigating this problem and there are no existing relevant benchmark datasets available. In this paper, we design a novel model[1] to solve this problem by combining a semi-supervised face recognition network and a multi-human tracker. For the face recognition network, we propose a semi-supervised learning method to fully leverage the unlabeled images in the video, thus reducing the required number of labeled face images. Triplet loss is also used during the training to better distinguish among inter-class samples. However, a single face recognition network is insufficient for video character tracking since people do not always show their frontal faces, or sometimes their faces are blocked by some obstacles. Therefore, a multi-human tracker is integrated into the model to address those problems. Additionally, we collect a dataset for the video character tracking problem, Character Face in Video, which can support various experiments for evaluating video character tracker performance. Experiments show that the proposed semi-supervised face recognition model can achieve more than 98.5% recognition accuracy, and our video character tracker can track in near-real-time and achieve 70% ~ 80% average intersection-over-union tracking accuracy on the dataset.

1. Introduction

The development of deep neural networks leads to tremendous achievements in many computer vision tasks such as image classification (He et al., 2016; Krizhevsky et al., 2017), object detection (Girshick, 2015; Redmon et al., 2016), instance segmentation (He et al., 2017), action detection (Zhao et al., 2017), multiple object tracking (Bewley et al., 2016; Wang et al., 2020), and image generation (Goodfellow et al., 2020). In the paper, we focus on a computer vision problem with very few previous investigations, the video character tracking problem.

The video character tracking problem aims to return the appearing time slots for certain characters of interest in given videos by using as few labeled face images of those characters as possible. Video character tracking has many potential applications such as movie analysis and automatic movie clipping for certain characters. For example, knowing the total appearance time and the appearing time slots of a certain character in various movies can be useful in analyzing the acting career of the character, and the video character tracker can also help to easily find movie clips in which a certain character appears from long videos. However, this problem is challenging for several reasons such as the shortage of provided labeled face images, small or unclear faces in videos, the large number of appearances of non-interest characters, and non-frontal or blocked faces in videos. Specifically, the first two reasons mentioned above indicate the challenges of using face recognition techniques to recognize faces in videos. The third challenge requires the face recognition model to be able to distinguish faces that are not of interest. Even if an ideal face recognizer is developed by tackling all three challenges above, a single face recognizer is still insufficient for solving the video character tracking problem accurately since faces in videos are sometimes non-frontal or even totally blocked by obstacles.

In this paper, we propose a semi-supervised Video Character Tracker (ViCTer), which is a novel model to return the appearing time slots for certain characters by combining face recognition techniques and multi-human tracking techniques. To solve the problem of the shortage of labeled face images, ViCTer employs semi-supervised learning methods to fully leverage a large number of unlabeled face images from the video, thus boosting face recognition accuracy. To make the face recognition network better distinguish non-interest faces that are never seen by the network, we further tune the network using the triplet loss to enlarge face embedding distances among inter-class samples. To

* Corresponding author at: National Center for Supercomputing Applications, University of Illinois at Urbana-Champaign, Urbana, 61801, IL, USA.
E-mail addresses: zl52@illinois.edu (Z. Li), xiwei2@illinois.edu (X. Wang), zz45@illinois.edu (Z. Zhang), kindrtnk@illinois.edu (V. Kindratenko).

[1] The code for the project is available on https://github.com/zilinghan/victer.

https://doi.org/10.1016/j.mlwa.2023.100460
Received 3 January 2023; Received in revised form 3 March 2023; Accepted 4 March 2023
Available online 11 March 2023

deal with the scenarios when faces are small, unclear, non-frontal, or totally blocked, making it impossible for the recognizer to give accurate inference, a multi-human tracker is integrated into ViCTer to track characters. In this manner, for each tracked character, we can confirm his/her identity even if we only have a small number of clear faces for accurate face recognition. Compared with a single face recognition network, the aggregation of face recognition network and multi-human tracker can significantly increase the video character tracking accuracy.

Since there are very few previous investigations on the video character tracking problem and no existing suitable benchmark datasets for evaluating the character tracking performance, we collect a dataset, Character Face in Video (CFIV), to evaluate the model performance. CFIV contains information of ten movie clips that can support the evaluation of several metrics such as face recognition accuracy, the ability for distinguishing unseen faces, and overall character tracking accuracy. Based on the CFIV dataset, our empirical evaluations show good performance for both the proposed semi-supervised face recognition network and the ViCTer model. Specifically, by only using two labeled face images per character for training, the face recognition network can achieve more than 98.5% accuracy. The use of triplet loss in training makes the distances among face embeddings for different characters sufficiently large to distinguish faces. Finally, for the video character tracking problem, ViCTer can achieve around 70%~80% tracking accuracy on the ten video clips from the CFIV dataset.

In summary, the main contributions of this paper are:

1. A semi-supervised face recognition network with triplet loss, which can achieve high accuracy using very few training face images and distinguish unseen faces.
2. A novel model, ViCTer, which aggregates the face recognition network and the multi-human tracker for solving the video character tracking problem.
3. A benchmark dataset, CFIV, which supports various experiments to evaluate the performance of video character trackers.
4. A thorough evaluation of the proposed face recognition network and the tracking model ViCTer.

2. Related work

In this section, we review the related work in three main fields: (1) face recognition techniques, (2) semi-supervised learning methods, and (3) multiple object tracking methods. Finally, we discuss the research gaps between current related technologies and the video character tracking problem.

2.1. Face recognition

Face recognition is an important task in modern industry and society for many reasons such as security and productivity. Such a task is challenging due to uncontrollable facial appearance variations, and varying position, luminosity, size, and angle of a picture captured by the camera.

According to Wang and Deng (2021), the research on face recognition can be divided into four main stages. The holistic methods such as Eigenface (Turk & Pentland, 1991), Fisherface (Belhumeur et al., 1997), Laplacianface (He et al., 2005), and face recognition via sparse representation (Wright et al., 2008) derive various low-dimension face representations, and then use a classifier or some representation distance metrics to recognize faces. The holistic methods were dominant in the 1990s and early 2000s, but they failed to deal with the uncontrollable facial appearance variations.

In early 2000s, face recognition based on handcrafted local descriptors such as Gabor (Liu & Wechsler, 2002) and LBP (Ahonen et al., 2006) became the trend in the community. Those methods employ some filtering techniques to generate face features that are both discriminative and invariant to certain face appearance changes to increase the model's robustness. However, it is still challenging for handcrafted encoding methods to achieve the optimal tradeoff between the distinctiveness and the robustness of descriptors. Additionally, the uneven distribution of handcrafted codes results in a lack of compactness in the code histogram.

To ease the encoding process and increase the compactness of the face descriptors, learning-based face descriptors were introduced in early 2010's to learn more discriminative and compact face representations. Typical learning-based methods such as LE (Cao et al., 2010) and PCANet (Chan et al., 2015) can achieve around 85% accuracy on the Labeled Face in Wild (LFW) (Huang et al., 2008) face recognition benchmark.

Though many efforts were put into learning-based methods to learn the shallow face representations to increase the robustness to face variances, the face recognition techniques were still unreliable in many practical scenarios. However, everything changed since AlexNet (Krizhevsky et al., 2012) achieved a significant image classification accuracy increase on the ImageNet (Deng et al., 2009) via its proposed deep learning method in 2012. The deep learning method learns more discriminative and robust face features by using a cascade of processing layers. In 2014, DeepFace (Taigman et al., 2014) and DeepID (Sun et al., 2014) achieve more than 97% accuracy on the LFW benchmark, which is approaching the human performance. Since then, the research on face recognition started to shift to deep-learning-based methods, and more advanced and accurate models such as FaceNet (Schroff et al., 2015) and VGGFace (Parkhi et al., 2015) were proposed. These models are now widely used as the base for fine-tuning various practical applications.

2.2. Semi-supervised learning

Deep neural networks usually can achieve great performance and accuracy when training with a large amount of labeled data. However, labeling a large amount of data is expensive and time-consuming, and may lead to bad user experiences in practical applications. Therefore, it is highly demanded to develop training methods that can train the neural networks with few labeled data while still retaining a high model accuracy. Semi-supervised learning is one of the methods which alleviates the requirements for the number of labeled data by leveraging a large amount of unlabeled data. Semi-supervised learning uses techniques such as pseudo labeling (Lee, 2013; Xie, Luong et al., 2020), consistency regularization (Bachman et al., 2014; Laine & Aila, 2017; Sajjadi et al., 2016; Tarvainen & Valpola, 2017), and their combinations (Berthelot et al., 2020, 2019; Kim et al., 2021; Sohn et al., 2020; Xie, Dai et al., 2020; Yuan et al., 2022).

Pseudo labeling techniques generate pseudo labels for unlabeled data if a certain label corresponds to high predicted probabilities, and the pseudo labels are used during the training. Consistency regularization techniques generate various augmentations for each unlabeled data sample and minimize the embedding distances of those augmentations. Advanced methods such as MixMatch (Berthelot et al., 2019), FixMatch (Sohn et al., 2020), and ActiveMatch (Yuan et al., 2022) combine those two techniques together to obtain higher model accuracy using very few labeled data.

2.3. Multiple object tracking

Multiple Object Tracking (MOT) is a task to process a given video to locate multiple objects and determine their identities and trajectories. It specifically focuses on the varying number of objects and their identities in the video. Besides this, initialization and termination of tracks, occlusions, interactions, and similarity among different objects are also key challenges in MOT problems. According to Luo et al. (2021), MOT solutions can be categorized based on three criteria, initialization methods, processing modes, and types of output.

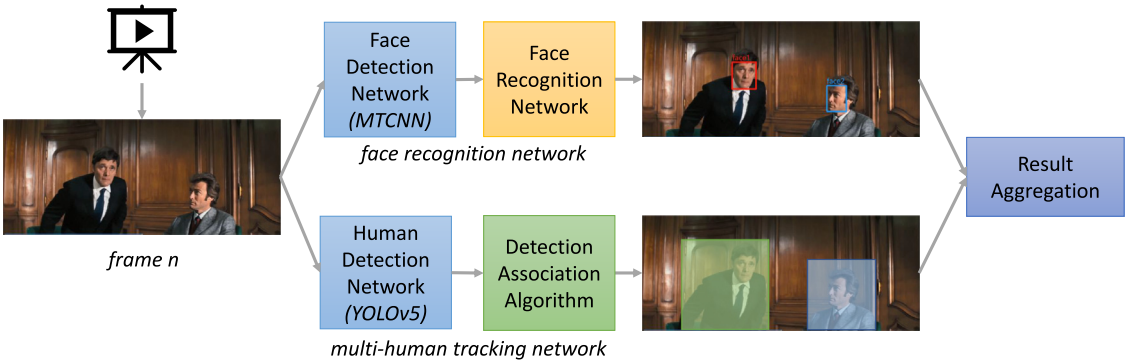

Fig. 1. Overview of the video character tracker. The model obtains the appearing frames for characters of interest by identifying the faces via the face recognition network, tracking the characters via the multi-human tracking network, and aggregating the results together.

Based on the initialization method, MOT solutions can be categorized into two kinds, detection-free tracking (DFT) and detection-based tracking (DBT). In detection-free tracking, objects are manually initialized in the first frame, and the trajectories of those objects are determined in the following frames (Zhang & van der Maaten, 2013). In detection-based tracking, a trained object detector is applied to each frame to detect objects, then the detections of objects are linked into trajectories (Song et al., 2010). For determining the trajectories of the manually labeled objects or detected objects, it is essentially a data association problem by using the appearance information (Kim et al., 2015; Rezatofighi et al., 2015) or motion information (Dicle et al., 2013; Yoon et al., 2015) of the objects, or the combinations of these two (Bewley et al., 2016).

Based on the processing methods, MOT solutions can be categorized into two kinds, online tracking and offline tracking. Online tracking such as SORT (Bewley et al., 2016) and its variants (Cao et al., 2022; Du et al., 2022; Wojke et al., 2017) is also called sequential tracking or causal tracking since it only uses information up to the current frame. Offline tracking employs a batch of frames, which could be the frames after the current frame (Song et al., 2010).

Based on the type of output, MOT solutions can be categorized into two kinds, stochastic tracking (Breitenstein et al., 2009; Rodriguez et al., 2011; Xing et al., 2009) and deterministic tracking (Butt & Collins, 2013; Pirsiavash et al., 2011; Yang & Nevatia, 2012). It means whether the outputs are identical in multiple runs on the same video.

2.4. Research gaps for video character tracking

As far as we know, there are no previous research papers investigating the video character tracking problem. A single multi-human tracker is insufficient since it cannot identify the tracked characters. As for the face recognition techniques, although they have been widely used in many fields for identification and authentication, using face recognition technology alone is still not the best solution to the video character tracking problem since people do not always show clear frontal faces in videos. Therefore, we propose this work to close the gaps by combining face recognition with multi-human tracking. Additionally, to achieve high recognition accuracy by using as few labeled faces as possible, semi-supervised learning methods are integrated into our proposed model to fully leverage large amounts of unlabeled faces from the videos.

3. Proposed model

Fig. 1 gives an overview of our proposed Video Character Tracker model. The model obtains the appearing frames for each character of interest by identifying the faces, tracking the characters, and aggregating the two sets of results together. A single face recognition network is insufficient for the video character tracking problem since people do not always show their frontal faces, or their faces are blocked by some obstacles, or sometimes people may only show the back of their heads. Integrating a human tracking network and aggregating the results together successfully address the problems. For each tracked character, there is no need to recognize the faces from every frame to determine the identity of the character. The details of the face recognition network, the multi-human tracking network, and the results aggregation algorithm are elaborated in the following subsections.

3.1. Semi-supervised face recognition network

As shown in Fig. 1, for face recognition, the multitask cascaded convolutional network (MTCNN) (Zhang et al., 2016) is used to first detect faces. Afterward, the face recognition network either matches each detected face to one character of interest or ignores it if dissimilar to any face. For the recognition network, we start from the FaceNet (Schroff et al., 2015) pretrained on the VGGFace2 dataset (Cao et al., 2018), and then fine-tuned the model using the video itself and the small labeled set of faces $\mathcal{L}$ provided by the user. From the video, the face detection network randomly collects a large unlabeled face set $\mathcal{U}$ with $|\mathcal{U}| \gg |\mathcal{L}|$. As shown in Fig. 2, the fine-tuning is composed of two stages. In stage one, the face recognition network is trained using the semi-supervised learning method adopted from FixMatch (Sohn et al., 2020). For a batch of labeled face images $\mathcal{B}_{\mathcal{L}} = \{\mathbf{x}_i, y_i\}_{i=1}^{B}$ with batch size B, each image is augmented to T images, the augmentation details are shown in Appendix A. The deep neural net and the fully connected layer compute the face logits $\mathbf{q}$ for each augmented image, where $\mathbf{q}$ is a vector with a length equal to the number of face classes in the labeled set. The training loss for the labeled faces is then defined as:

$$l_1(\mathcal{B}_{\mathcal{L}}) = \frac{1}{BT}\sum_{i=1}^{B}\sum_{t=1}^{T} H(y_i, \text{softmax}(\mathbf{q}_i^{(t)})), \tag{1}$$

where $\mathbf{q}_i^{(t)}$ is the face logit for the tth augmentation of image $\mathbf{x}_i$, and $H(\cdot)$ represents the cross-entropy function (Good, 1952). For the larger batch of unlabeled face images $\mathcal{B}_{\mathcal{U}} = \{\mathbf{x}_i\}_{i=1}^{\mu B}$, each image is augmented by one weak augmentation and one strong augmentation to obtain $\mathbf{x}_i^w$, $\mathbf{x}_i^s$ respectively. The details of the weak and strong augmentations are shown in Appendix A. The network computes the logits $\mathbf{q}_i^w$ and $\mathbf{q}_i^s$ for $\mathbf{x}_i^w$ and $\mathbf{x}_i^s$. If $\max(\text{softmax}(\mathbf{q}_i^w))$ exceeds a confidence threshold c, then $\hat{y}_i = \text{argmax}(\mathbf{q}_i^w)$ is considered as the pseudo label for the unlabeled face image $\mathbf{x}_i$. The training loss for the unlabeled faces is essentially the cross-entropy loss using the pseudo labels:

$$l_2(\mathcal{B}_{\mathcal{U}}) = \frac{1}{\mu B}\sum_{i=1}^{\mu B} \mathbb{1}(\max(\text{softmax}(\mathbf{q}_i^w)) > c) H(\hat{y}_i, \mathbf{q}_i^s), \tag{2}$$

where $\mathbb{1}(\cdot)$ is the indicator function that evaluates to 1 if and only if the condition inside is true. For stage one of the fine-tuning, the overall training loss is:

$$l = l_1(\mathcal{B}_{\mathcal{L}}) + \lambda l_2(\mathcal{B}_{\mathcal{U}}). \tag{3}$$

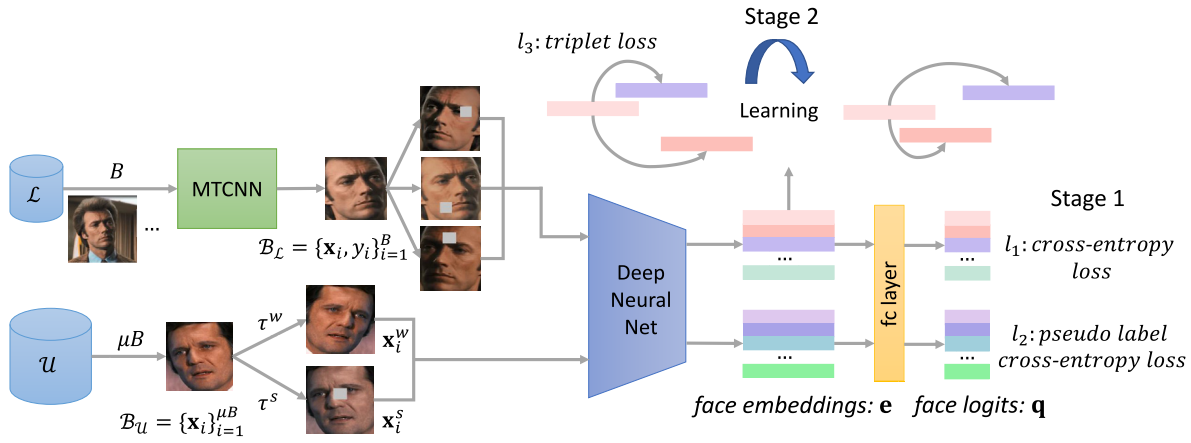

Fig. 2. Training process of the face recognition network. The model consists of a pretrained deep neural net backbone (FaceNet) and a randomly initialized fully connected layer. The training is composed of two stages: Stage one is based on semi-supervised learning methods using cross-entropy loss and pseudo label cross-entropy loss. Stage two is based on the supervised learning method using the triplet loss on the face embeddings.

For stage two of the fine-tuning, the last fully connected layer is disabled, and only the face embeddings from the deep neural net are used for training. This stage uses supervised learning with the triplet loss (Schroff et al., 2015) as the training loss. The underlying idea of triplet loss is to close the distances among face images of the same person and enlarge the distances among face images of different persons. For each face image $\mathbf{x}_i$, other face images from the same person are considered as positive samples $\mathbf{x}_i^p$, and face images from different persons are considered as negative samples $\mathbf{x}_i^n$. $\mathbf{e}_i$, $\mathbf{e}_i^p$, and $\mathbf{e}_i^n$ are their corresponding output face embeddings from the neural net. Mathematically, we want:

$$\|\mathbf{e}_i - \mathbf{e}_i^p\|_2^2 + \alpha < \|\mathbf{e}_i - \mathbf{e}_i^n\|_2^2, \tag{4}$$

where α is a forced margin between the positive sample embeddings and negative sample embeddings. Then the triplet training loss for stage two can be defined as:

$$l_3(\mathcal{B}_{\mathcal{L}}) = \sum_{i=1}^{B}\sum_{t=1}^{T}[\|\mathbf{e}_i^{(t)} - \mathbf{e}_i^{(t),p}\|_2^2 - \|\mathbf{e}_i^{(t)} - \mathbf{e}_i^{(t),n}\|_2^2 + \alpha]_+, \tag{5}$$

where $[x]_+ = \max(x, 0)$. When computing the triplet loss in our model, for each face image $\mathbf{x}_i$, we choose all face images of the same person as positive samples and only the hardest negative sample. The hardest negative sample is the face image of a different person with the smallest embedding distance to $\mathbf{e}_i^{(t)}$.

Since there are faces of non-interest persons in real settings, it is not suitable to enable the last fully connected layer and treat the face recognition network as a general classifier in practice. Therefore, we disable the last fully connected layer and first obtain the corresponding embedding set $E_k = \{\mathbf{e}_i | y_i = k\}_{i=1}^{N_k}$ for each character of interest k. During the inference, for each face image $\mathbf{x}$ and its corresponding embedding $\mathbf{e}$, we compute its embedding distance to each person k by averaging the distances to every embedding in E_k:

$$d(E_k, \mathbf{e}) = \frac{1}{N_k}\sum_{i=1}^{N_k}\|\mathbf{e}_i - \mathbf{e}\|, \text{ where } \mathbf{e}_i \in E_k \tag{6}$$

then the face class $\hat{y}$ predicted by the face recognition network is given by:

$$\hat{y} = \begin{cases} \underset{k}{\text{argmin}}(d(E_k, \mathbf{e})) & \text{if } \min(d(E_k, \mathbf{e})) < d, \\ -1 & \text{otherwise.} \end{cases} \tag{7}$$

If the minimum embedding distance from $\mathbf{e}$ to any embedding set is smaller than a distance threshold d, then the face class is predicted as the closest face. Otherwise, the network returns -1 to indicate that face image $\mathbf{x}$ does not match any character of interest. The value of the hyperparameter d is determined adaptively with details elaborated in Section 4.5.

3.2. Multi-human tracking network and results aggregation

Since people do not always show their frontal faces in the video or the faces are sometimes blocked by some obstacles, a single face recognition network is insufficient to accurately give the appearing frames for characters of interest. Therefore, ViCTer uses a multi-object tracking (MOT) network to track humans, namely, a multi-human tracking network. A multi-human tracking network returns several tracks for a video input, where a track is essentially a collection of images of the same person. Our model aggregates the recognition and tracking results together to obtain the appearing frames for characters of interest.

We use the detection-based tracking method for our multi-human tracking network, which first applies a YOLOv5 (Jocher, 2021) model pretrained on the COCO dataset (Lin et al., 2014) to detect humans, and the detected humans are then associated to tracks. For the detection association, ViCTer integrates two algorithms, StrongSORT algorithm (Du et al., 2022) and Observation-Centric SORT (OCSORT) algorithm (Cao et al., 2022). The StrongSORT algorithm associates the detected humans with the tracks according to the motion and appearance information of the humans. Specifically, the motion information is obtained by recursive Kalman filtering (Kalman, 1960) and the Hungarian method (Kalman, 1955). The appearance information is captured by an appearance descriptor generated by a convolutional neural network for each detection. As for the OCSORT algorithm, it achieves state-of-the-art tracking performance without using appearance information. It employs the "observation" to recover lost tracks and reduce the errors accumulated by linear motion models during the lost period.

For each video frame, let set $\mathcal{R} = \{(\mathbf{b}_i^{\text{face}}, y_i)\}$ be the results of the face recognition network and set $\mathcal{T} = \{(\mathbf{b}_j^{\text{human}}, p_j)\}$ be the results of the multi-human tracking network. Specifically, $\mathbf{b}_i^{\text{face}}$ and y_i are the bounding box position and the recognized face class for each face in the frame, $\mathbf{b}_j^{\text{human}}$ and p_j are the bounding box position and track index for each human in the frame. We aggregate $\mathcal{R}$ and $\mathcal{T}$ to get an appearing dictionary $\mathcal{D}$. The detailed data structure of $\mathcal{D}$ is shown in Fig. 3. Specifically, each entry of $\mathcal{D}$ is a sub-dictionary for each person track, and the sub-dictionary contains the information of several face classes. Since the tracking network employs the motion information during the detection-track association and the scene changes in some videos may result in small variance in the position of the detected person, it is possible that each person track may actually correspond to multiple faces as illustrated in Fig. 4. Therefore, we need a sub-dictionary to store the information of different face classes for each person track. The y^{new} and fr^{new} attributes of the sub-dictionary are used to detect new coming face classes for a given person track to deal with the issue of multiple character faces per person track in Fig. 4.

```
𝒟 = {
    Person Track 0 (𝒟[0]): {
        Face 0 (𝒟[0][0]): {
            face class (y): 0;
            face frame indices (fr): [0, 1, …, 92];
            new face class (y^new): 3;
            new face frame indices (fr^new): [93, 94, …];
        };
        Face 1 (𝒟[0][1]): {
            face class (y): 3;
            face frame indices (fr): [93, 94, …, 180];
            new face class (y^new): 2;
            new face frame indices (fr^new): [181, 182, …];
        };
        Face 2 (𝒟[0][2]): {
            face class (y): 2;
            face frame indices (fr): [181, 182, …, 240];
            new face class (y^new): None;
            new face frame indices (fr^new): [];
        };
    };
    Person Track 1 (𝒟[1]): {
        Face 0 (𝒟[1][0]): {
            face class (y): 1;
            face frame indices (fr): [0, 1, …, 150];
            new face class (y^new): None;
            new face frame indices (fr^new): [];
        };
    };
    Person Track 2 (𝒟[2]): {……};
}
```

Fig. 3. Data structure for appearing dictionary $\mathcal{D}$.

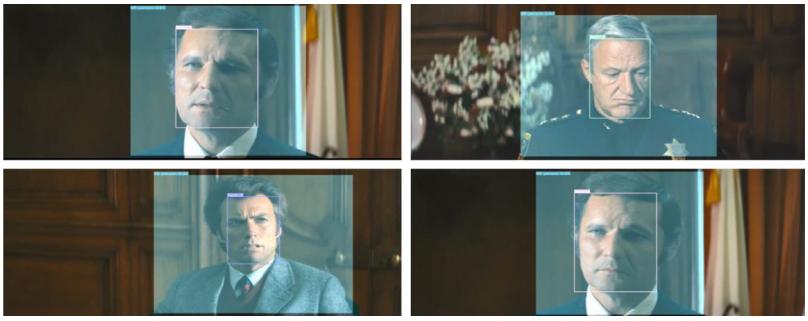

Fig. 4. Same track for multiple characters. Several characters are associated to the same person track due to the small variance in their locations.

Algorithm 1 gives the algorithm to obtain the appearing dictionary $\mathcal{D}$ from $\mathcal{R}$ and $\mathcal{T}$. The loop between lines 1 and 25 iterates over each recognized face. Line 2 associates the face with a tracked person by finding the person whose bounding box has the largest overlap with the face bounding box. The tracked person is then popped out from $\mathcal{T}$. If person track p_i is not in the appearing dictionary $\mathcal{D}$, a new record for it is added and initialized. As mentioned before, though each person track may correspond to multiple face classes, the current face class associated with person track p_i is always the face class of the last face of $\mathcal{D}[p_i]$, so we obtain its index $f_i = \texttt{len}(\mathcal{D}[p_i]) - 1$. If the face class of the detected face y_i is the same as the current face class associated with p_i ($\mathcal{D}[p_i][f_i][y]$), the current frame index fid is appended to the corresponding frame indices. Additionally, if the frame indices of the new face class ($\mathcal{D}[p_i][f_i][fr^{new}]$) are not empty, they are also appended to the frame indices of the current face class (we assume the previous new face class may come from some inaccurate recognition). If y_i is different from the current face class associated with p_i, we record the information of the new face class into y^{new} and fr^{new}. If there is no previous new face class yet, we initialize y^{new} and fr^{new} as shown in lines 12 and 13. If y_i is the same as the previous new face class, we append fid to the frame indices of the new face class. When the length of fr^{new} exceeds a threshold l_{max}, we create a new face instance for person track p_i. If y_i is different from the previous new face class, we append fr^{new} to fr, assuming the previous new face class is due to inaccurate recognition. Afterward, we set the information of the new face class according to y_i.

The loop between lines 26 and 34 iterates over all person tracks that are not associated with any face. This may happen because some

characters are not showing their face in the current frame, or the faces are not sufficiently clear for the face recognition network to give a confident inference. This loop adds fid either to the frame indices of the current face class fr if there is no new face class, or to the frame indices of the new face class fr^{new} otherwise.

After obtaining the appearing dictionary $\mathcal{D}$ by running Algorithm 1 for all video frames, Algorithm 2 gives the algorithm for converting $\mathcal{D}$ to the appearing frames for all characters of interest $\mathcal{A}$. $\mathcal{A}$ is also a dictionary with the face class for the character of interest as the key and the sorted list of appearing frame indices as the value.

Algorithm 1 Recognition and Tracking Results Aggregation

Input: face recognition output ($\mathcal{R}$), human tracking output ($\mathcal{T}$), current frame index (fid), maximum length of fr^{new} before creating a new face instance for corresponding person track (l_{max})
Output: appearing dictionary ($\mathcal{D}$)

for all $(\mathbf{b}_i^{\text{face}}, y_i) \in \mathcal{R}$ **do**
  $(\mathbf{b}_i^{\text{human}}, p_i) :=$ max bbox overlap$((\mathbf{b}_i^{\text{face}}, y_i), \mathcal{T})$;
  $\mathcal{T} := \mathcal{T}$ - $(\mathbf{b}_i^{\text{human}}, p_i)$;
  add p_i to $\mathcal{D}$ and initialize the entry **if** $p_i \notin \mathcal{D}$;
  $f_i :=$ `len(`$\mathcal{D}[p_i]$`)` - 1;
  if $y_i = \mathcal{D}[p_i][f_i][y]$ **then**
    $\mathcal{D}[p_i][f_i][fr]$.add($\mathcal{D}[p_i][f_i][fr^{new}]$);
    $\mathcal{D}[p_i][f_i][fr]$.add($fid$);
    $\mathcal{D}[p_i][f_i][fr^{new}] :=$ [];
    $\mathcal{D}[p_i][f_i][y^{new}] :=$ None;
  else if $\mathcal{D}[p_i][f_i][y^{new}] =$ None **then**
    $\mathcal{D}[p_i][f_i][y^{new}] := y_i$;
    $\mathcal{D}[p_i][f_i][fr^{new}] :=$ [fid];
  else if $\mathcal{D}[p_i][f_i][y^{new}] = y_i$ **then**
    $\mathcal{D}[p_i][f_i][fr^{new}]$.add($fid$);
    if len($\mathcal{D}[p_i][f_i][fr^{new}]$) $> l_{max}$ **then**
      $\mathcal{D}[p_i][f_i{+}1][y] := \mathcal{D}[p_i][f_i][y^{new}]$;
      $\mathcal{D}[p_i][f_i{+}1][fr] := \mathcal{D}[p_i][f_i][fr^{new}]$;
    end if
  else if $\mathcal{D}[p_i][f_i][y^{new}] \neq y_i$ **then**
    $\mathcal{D}[p_i][f_i][fr]$.add($\mathcal{D}[p_i][f_i][fr^{new}]$);
    $\mathcal{D}[p_i][f_i][fr^{new}] :=$ [fid];
    $\mathcal{D}[p_i][f_i][y^{new}] := y_i$;
  end if
end for
for $(\mathbf{b}_i^{\text{human}}, p_i) \in \mathcal{T}$ **do**
  add p_i to $\mathcal{D}$ and initialize the entry **if** $p_i \notin \mathcal{D}$;
  $f_i :=$ `len(`$\mathcal{D}[p_i]$`)` - 1;
  if $\mathcal{D}[p_i][f_i][y^{new}] \neq$ None **then**
    $\mathcal{D}[p_i][f_i][fr^{new}]$.add($fid$);
  else
    $\mathcal{D}[p_i][f_i][fr]$.add($fid$);
  end if
end for
return $\mathcal{D}$

Algorithm 2 Appearing Frame Dictionary Generation

Input: appearing dictionary ($\mathcal{D}$), number of characters of interest (n)
Output: appearing time slots for characters of interest ($\mathcal{A}$)

$\mathcal{A} :=$ {};
$\mathcal{A}[i] :=$ set( ) **for** $i \in$ range(n);
for $p \in \mathcal{D}$ **do**
  for $f \in \mathcal{D}[p]$ **do**
    if $\mathcal{D}[p][f][y] \notin$ {None, -1} **then**
      $\mathcal{A}[\mathcal{D}[p][f][y]]$.add($\mathcal{D}[p][f][fr]$);
    end if
  end for
end for
$\mathcal{A}[i] :=$ list($\mathcal{A}[i]$).sort();
return $\mathcal{A}$

4. Experiments

In this section, we first give an introduction to the dataset collected for the video character tracking problem. We then evaluate the performance of our proposed model on the collected dataset along several dimensions. In Section 4.2, we enable the last fully connected layer and use the face recognition network as a general classifier, and report on the improvements in the face classification accuracy of our proposed semi-supervised training method with triplet loss. However, since it is not sufficient to treat the recognition network as a general classifier due to many unseen faces in practice, we show the effectiveness of our face recognition network in distinguishing different faces by comparing the learned face embedding distances in Section 4.3. In Section 4.4, we show clustering plots of the face embeddings during the training process to visualize how faces are gradually clustered together. In Section 4.5, we show the performance of our whole model ViCTer in the video character tracking problem and compare it with the solution only using a single face recognition network.

4.1. Datasets

Since there are few research papers investigating the video character tracking problem and no existing suitable benchmark datasets for performance evaluation, we collect a dataset, Character Face in Video (CFIV) (Li et al., 2022), which can support various experiments to evaluate the performance of the video character tracker. CFIV contains the information of ten short videos downloaded from YouTube. For each video, the dataset includes the code for downloading the video, two labeled face images per character for training, and the appearing time slots for each character. Additionally, three videos in CFIV have a labeled testset that contains around 100 face images per character for testing the face recognition accuracy, together with faces of some unseen characters for testing the model's ability to distinguish unseen faces. A detailed description of the properties of each video in CFIV is shown in Appendix B.

4.2. Face recognition accuracy

Definition. For the video character tracking problem, the pretrained face recognition network is provided with a video V and l labeled faces for each character of interest appearing in V as the training data. In this experiment, the face recognition network is used as a general face classifier to match each testset face to one character.

Experiment Details. We evaluate our proposed model on three videos in our collected dataset using one or two labeled face images per character. In addition, we also show the face recognition accuracy of the fully supervised model trained only using the labeled faces. FaceNet (Schroff et al., 2015) pretrained on the VGGFace2 dataset (Cao et al., 2018) is used as the neural net backbone, and it should be noted that other state-of-the-art models such as DeepFace (Taigman et al., 2014) and VGGFace (Parkhi et al., 2015) lead to comparable performance. During the first stage of the training, the pretrained deep neural net backbone is trained with an initial learning rate lr_1, and the randomly-initialized fully connected layer is trained with a larger learning rate $\beta \cdot lr_1$. The initial learning rate for the triple loss training in the second stage is lr_2. For both stages, we employ cosine learning rate decay to mitigate overfitting (Loshchilov & Hutter, 2017), $lr = lr_i \cdot \cos(7\pi k/16K)$, where lr_i is the initial learning rate, K is the total number of training steps and k is the current training step. The model

Table 1
Face recognition accuracy. Recognition accuracy for different training methods with different numbers of labeled faces per person.

	Dirty Harry 2		*Scent of a Woman 2*		*Growing Pain II*	
Method	1 label	2 labels	1 label	2 labels	1 label	2 labels
Supervised	79.42	93.58	90.09	96.23	91.75	95.63
Supervised + triplet loss	86.95	95.35	93.40	97.88	94.66	97.57
Semi-supervised	95.58	98.89	96.70	97.88	94.42	98.54
Semi-supervised + triplet loss	96.90	99.12	98.35	98.58	97.82	99.27

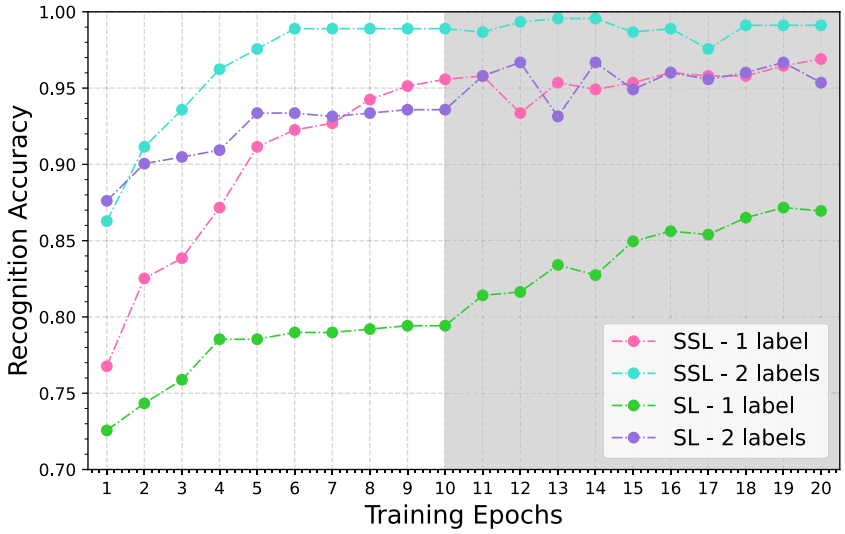

Fig. 5. Face recognition accuracy during training. The changes in the face recognition accuracy during the training process, where the first ten epochs use either supervised (SL) or semi-supervised (SSL) training, and the rest use the triplet loss training.

is trained for n epochs for the stage one semi-supervised training, and n epochs for the stage two triplet loss training. All hyperparameter values for this experiment are shown in Appendix C.

Experiment Results. Table 1 shows face recognition accuracy for both the supervised and semi-supervised training methods using one or two labels per character. In addition, it also lists the accuracy before the triplet loss training to showcase how triplet loss training contributes to the overall accuracy improvements. These results show that our proposed face recognition model can achieve more than **98.5**% accuracy using two faces per person. The use of semi-supervised learning leads to **1**% $\sim$ **4**% accuracy improvements, and the use of triplet loss training leads to **1**% $\sim$ **3**% further accuracy improvements. Fig. 5 shows the change in the face recognition accuracy during the training process using one or two labeled face images per character on *Dirty Harry 2*. Specifically, for the first ten epochs, we run either supervised or semi-supervised training, and for the last ten epochs, we run the triplet loss training.

4.3. Face embedding distances

For the video character tracking problem, since there are many faces in the video not existing in the labeled set, it is insufficient to use the face recognition network as a classifier as defined in Section 4.2. Therefore, we employ the distances between face embeddings to recognize faces as described in Section 3.1. In this experiment, we measure the embedding distances among faces from the same and different characters according to the following definition to evaluate how well the proposed face recognition network can distinguish faces using face embeddings.

Definition. Suppose $E_1^{(\mathcal{L})} = \{\mathbf{e}_i | y_i = 1\}_{i=1}^{N_i}$ is the embedding set for the labeled faces of character one, $E_1^{(\mathcal{T})} = \{\mathbf{e}_j | y_j = 1\}_{j=1}^{N_j}$ is the embedding set for the faces in the testset of character one, and $E_2^{(\mathcal{T})} = \{\mathbf{e}_k | y_k = 2\}_{k=1}^{N_k}$ is the embedding set for the faces of another character. For $\mathbf{e_j} \in E_1^{(\mathcal{T})}$ and $\mathbf{e_k} \in E_2^{(\mathcal{T})}$, the embedding distances to character one are defined in Eqs. (8) and (9), which is essentially the average distance to each embedding of character one. The values of $d(E_1^{(\mathcal{L})}, \mathbf{e}_j)$ and $d(E_1^{(\mathcal{L})}, \mathbf{e}_k)$ can give indications on how well the face recognition network can distinguish faces using embedding distances.

$$d(E_1^{(\mathcal{L})}, \mathbf{e}_j) = \frac{1}{N_i} \sum_{i=1}^{N_i} \|\mathbf{e}_i - \mathbf{e}_j\| \tag{8}$$

$$d(E_1^{(\mathcal{L})}, \mathbf{e}_k) = \frac{1}{N_i} \sum_{i=1}^{N_i} \|\mathbf{e}_i - \mathbf{e}_k\| \tag{9}$$

Experiment Details. For the set $E_1^{(\mathcal{L})}$, it is the face embedding set of the augmented labeled training faces for a certain character. $E_1^{(\mathcal{T})}$ is the embedding set of testset face images for the corresponding character, and $E_2^{(\mathcal{T})}$ is the embedding set of face images of a character that is never seen by the face recognition network. We then compute set $\mathcal{D}_1 = \{d(E_1^{(\mathcal{L})}, \mathbf{e}_j)\}_{j=1}^{N_j}$ and $\mathcal{D}_2 = \{d(E_1^{(\mathcal{L})}, \mathbf{e}_k)\}_{k=1}^{N_k}$, and employ the following two metrics to measure the degree of separation for the two different persons.

- *Average separation:* the difference between the average value of $\mathcal{D}_2$ and $\mathcal{D}_1$.
- *False negative rate:* If using $\min(\mathcal{D}_2)$ as the threshold distance d to decide whether a face belongs to character one or not, false negative rate refers to the percentage of face embeddings in $E_1^{(\mathcal{T})}$ is not correctly recognized as character one. Namely, $|\{j \text{ where } d(E_1^{(\mathcal{L})}, \mathbf{e}_j) > \min(\mathcal{D}_2)\}|/N_j$.

The experiments in this section are conducted on video *Dirty Harry 2* to compare the embedding distances between another character in the movie (but never appearing in the given movie clip) and all four characters in the labeled set.

Experiment Results.

Fig. 6 shows the embedding distances to certain faces before and after the triplet loss training. Specifically, the blue dots represent

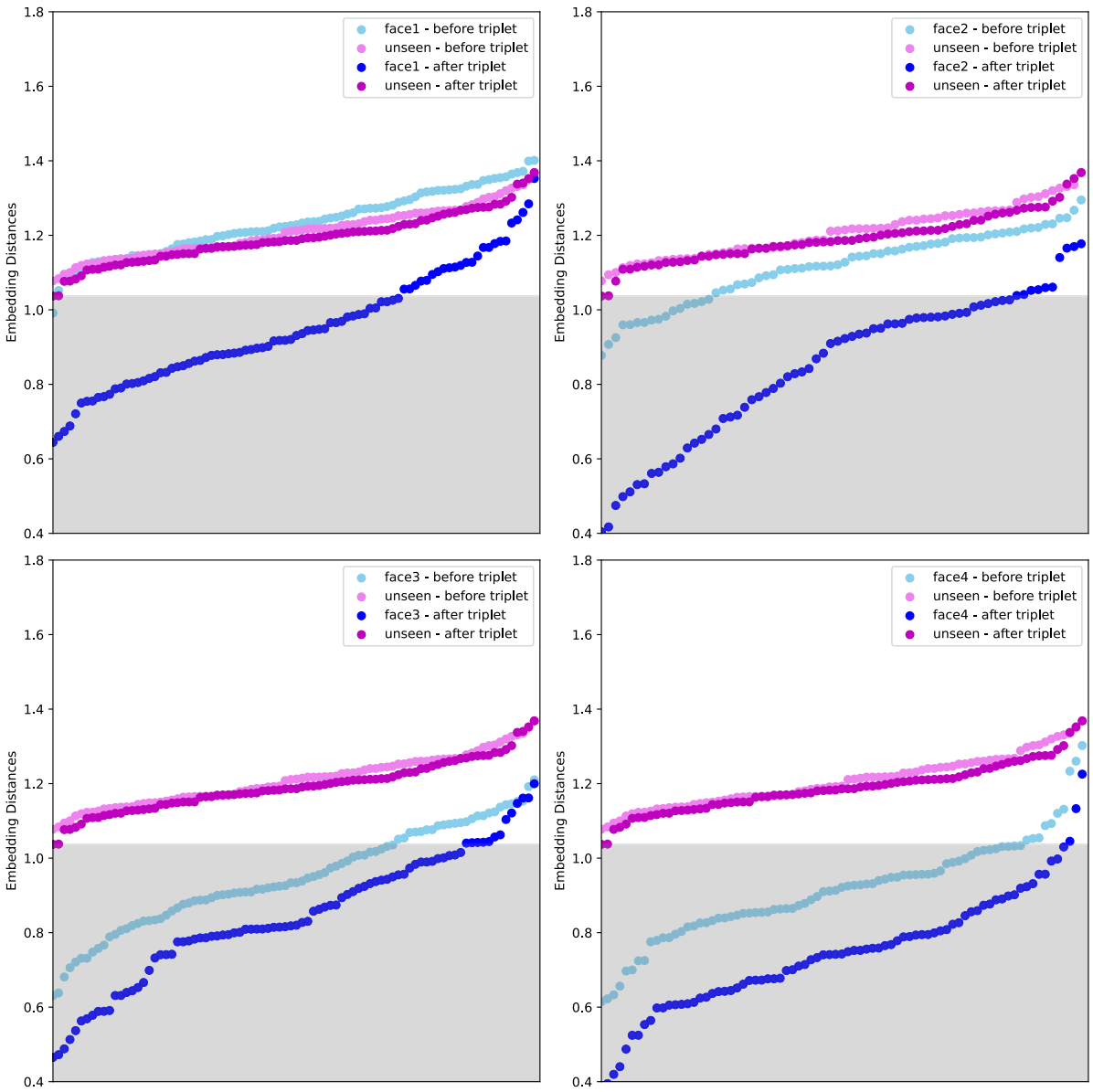

Fig. 6. Embedding distances to certain faces before and after the triple loss training. Blue dots represent faces from the same person, and violet dots represent faces from another person. Dots with lighter colors are embedding distances before the triplet loss training, and dots with darker colors are embedding distances after the triplet loss training. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

faces from the same person, and the violet dots represent faces from another person. Dots with lighter colors are embedding distances before the triplet loss training, and dots with darker colors are embedding distances after the triplet loss training. The figures illustrate how the faces of the same person and another person get separated during the triplet loss training process and emphasize the importance of the triplet loss training stage. Additionally, the gray area is the area with distances less than $\min(D_2)$ after the triplet loss training, which can show a significant drop in the false negative rate.

Figs. 7 and 8 show the average separations and false negative rates before and after the triplet loss training on four different faces. The triplet loss training enlarges the average separations and reduces the false negative rates. From Fig. 8, we can see that Face 1 still has a 27.9% false negative rate after the triplet loss training stage, but as mentioned before, the ViCTer model does not require recognition of faces in every frame with the integration of the multi-human tracker.

4.4. Face clustering

Experiment Details. There are several ways to reduce the dimension of embeddings to 2-d or 3-d for visualizing how the embeddings from the same class cluster together, such as principle component analysis (PCA), t-SNE (Van der Maaten & Hinton, 2008), and UMAP (McInnes et al., 2018). In this section, we employ t-SNE and UMAP to visualize the embeddings of faces in the testset of *Dirty Harry 2* during the training process. The face recognition model is trained for 10 epochs in stage one semi-supervised training, and 25 epochs in stage two triplet loss training.

Experiment Results. Figs. 9 and 10 show the t-SNE and UMAP clustering results of the face embeddings during the training of the face recognition network. Specifically, the first row shows the clusters after every two training epochs in stage one, and the second row shows the clusters after every five training epochs in stage two. From the plots, we can clearly see that the clusters get tighter and the number of outliers

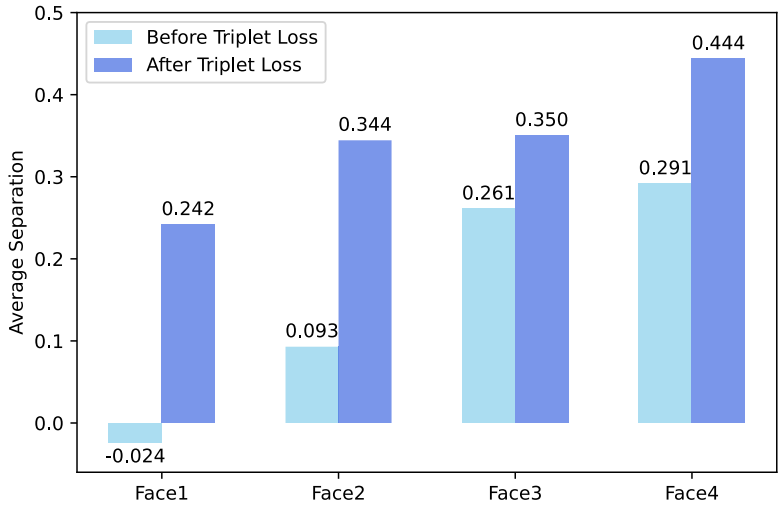

Fig. 7. Impact of triplet loss training on average separation.

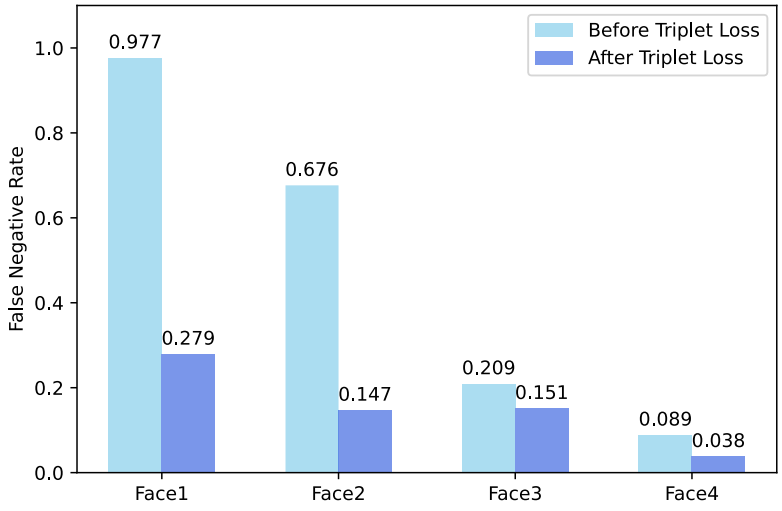

Fig. 8. Impact of triplet loss training on false negative rate.

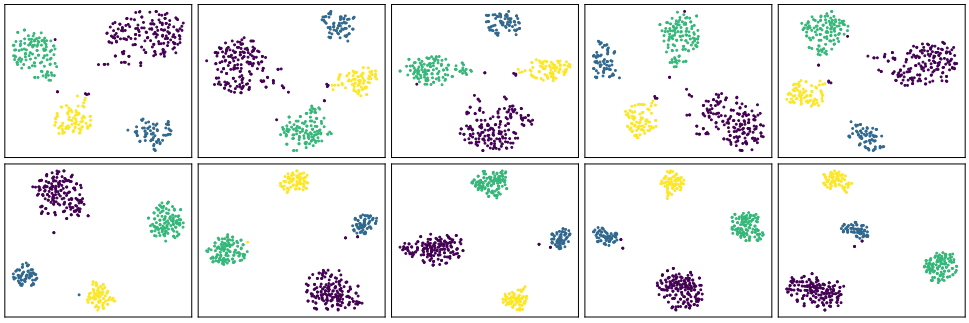

Fig. 9. t-SNE cluster results of the learned face embeddings during the training process. The first row shows the t-SNE clusters after every two training epochs in stage one, and the second row shows the t-SNE clusters after every five training epochs in stage two.

decreases during the triplet loss training stage. Fig. 11 shows some outlier examples in the clustering results after the complete training process. The outliers are all non-frontal faces so it is difficult for the face recognition network to obtain good embeddings for them.

4.5. Character tracking accuracy

Definition. Given a video V and l labeled faces per character, the video character tracker is supposed to return the appearing time slots for all l characters $\{\mathbf{T_i}\}_{i=1}^{l}$. For each $\mathbf{T_i}$, it is a list of $(t_{\text{start}}, t_{\text{end}})$ tuples, describing the start and end points of a time period. For measuring the accuracy of the predicted appearing time slots, we employ the Intersection-over-Union (IoU) accuracy with details shown in Fig. 12, which is the length of the intersection period of the prediction and the ground truth divided by the length of the union period of the prediction and the ground truth.

Experiment Details. We conduct experiments on all ten videos from the CFIV dataset. To set up a baseline for the video character

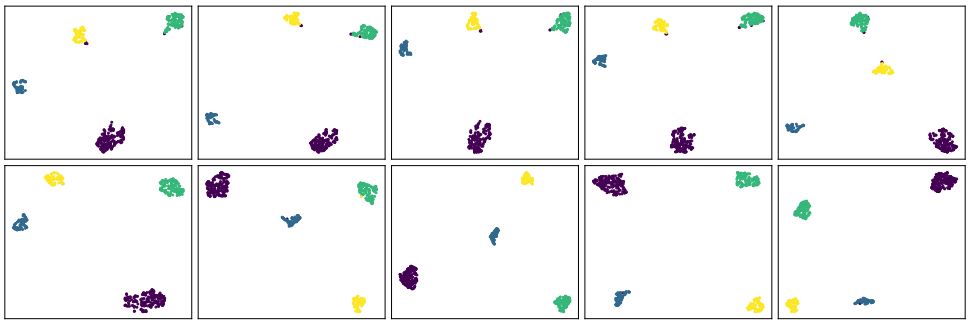

Fig. 10. UMAP cluster results of the learned face embeddings during the training process. The first row shows the UMAP clusters after every two training epochs in stage one, and the second row shows the UMAP clusters after every five training epochs in stage two.

Fig. 11. Outlier images. Outliers of the clustering results after the complete training process.

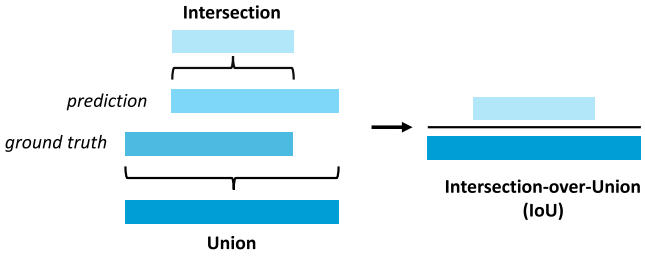

Fig. 12. Illustration of intersection-over-union accuracy.

tracking problem, we only employ our proposed face recognition network and use the appearing time for each face as the appearing time for that particular character. As for the ViCTer model, we run the experiments with two different detection-track association algorithms, StrongSORT and OCSORT. Additionally, we also run the experiments with different strides, where the stride stands for out of how many continuous video frames we should select one frame for analysis.

All the hyperparameters used for training the face recognition network have the same values as the experiments in Section 4.2 detailed in Appendix C. As for the embedding distance threshold d, it is determined adaptively by utilizing the unlabeled face images from the video. For each image $\mathbf{x}_i$ from the batch of unlabeled face images $\mathcal{B}_{\mathcal{U}} = \{\mathbf{x}_i\}_{i=1}^{\mu B}$, we first get its learned embedding $\mathbf{q}_i$. If $\max(\text{softmax}(\mathbf{q}_i)) > 0.995$, indicating that the unlabeled face image has a very high probability of being a character of interest, then we compute its embedding distance to the certain character and add the embedding distance to a distance list L. Afterward, we sort the list L in ascending order and set d according to the following rule:

$$d = \begin{cases} 0.85 & \text{if } L[\text{len}(L) * 0.8] \leq 0.85, \\ 1.05 & \text{if } L[\text{len}(L) * 0.8] \geq 1.05, \\ L[\text{len}(L) * 0.8] & \text{otherwise.} \end{cases} \tag{10}$$

This rule essentially makes the face recognition network able to recognize 80% faces in L and bounds the value of d in the range from 0.85 to 1.05.

Experiment Results. Table 2 shows the average IoU accuracy for different methods with various stride values over ten collected videos in the CFIV dataset. The ViCTer model can achieve around **70**% ∼ **80**% IoU accuracy and has a significant accuracy boost compared with a single face recognizer. Table 3 shows the execution time of the model training and model inference on one NVIDIA Tesla V100 SXM2 16 GB GPU. Since the execution time of the face recognition model training is a constant overhead that is irrelevant to the length of the video, the absolute execution time in seconds is shown in the table. For the inference time, we show the relative execution time, which is the inference time divided by the video length. From the table, we derive four main observations. (1) For most videos, the ViCTer model with the OCSORT algorithm can achieve near-real-time tracking performance (inference time ratio approaches one) with stride one. (2) The OCSORT algorithm is around twice faster as the StrongSORT algorithm since it does not use a neural net to encode appearance information during the tracking process. (3) Increasing stride value can significantly reduce inference time while not sacrificing too much of accuracy. (4) There are some videos (V3, V6) that take much longer inference time since the inference time is dependent on the total number of characters in the video, and there are many characters appearing in those videos.

Fig. 13 gives the tracking results on three sample videos V1 to V3. Different colors correspond to the appearing time slots of different characters. For each character, the top line is the tracking result of the baseline method which only uses the face recognizer, the central line is the result of ViCTer using the StrongSORT association algorithm, and the bottom line is the ground truth appearing time slot for that character. We can see that the recognition-based method has many gaps in the predicted appearing time slots since the character faces are not always sufficiently clear for recognition. However, with the integration of the multi-human tracker, those gaps can be linked together to generate more accurate predictions.

5. Discussion and future work

In this paper, we address the video character tracking problem by combining a face recognition network with a multi-human tracker to deal with the scenarios when a single face recognition network cannot recognize faces. However, the bottleneck of this model still lies in the multi-human tracker. The state-of-the-art multi-object tracker still suffers from the re-identification problem, and it gets even worse when there are significant camera movements or many scene changes in the video. Therefore, further work in the MOT field to improve the model's

Table 2
Video character tracking Intersection-over-Union (IoU) accuracy over ten collected videos in the CFIV dataset. The average IoU accuracy for three different methods (a single face recognizer, ViCTer with StrongSORT association algorithm, and ViCTer with OCSORT association algorithm) and stride equal to 1, 2 or 4 over ten videos in CFIV.

		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
Stride = 1	Face recognition only	62.47	56.46	84.80	71.85	75.84	64.98	48.62	56.98	58.81	70.69
	ViCTer (StrongSORT)	83.50	75.71	91.53	75.75	84.43	68.32	60.37	68.67	77.68	74.63
	ViCTer (OCSORT)	78.40	78.90	83.13	71.28	81.93	59.37	71.11	77.82	77.29	80.32
Stride = 2	Face recognition only	61.97	53.55	82.88	70.33	74.82	64.52	46.86	55.52	59.69	70.39
	ViCTer (StrongSORT)	79.82	72.10	92.18	71.92	80.44	67.46	59.20	65.38	74.07	71.07
	ViCTer (OCSORT)	75.55	77.82	87.07	70.62	83.18	55.78	70.66	77.19	77.49	74.63
Stride = 4	Face recognition only	60.57	54.25	81.36	67.65	72.82	63.73	46.34	53.92	54.24	66.90
	ViCTer (StrongSORT)	73.83	68.17	89.77	67.59	77.47	64.41	56.23	61.91	65.19	66.96
	ViCTer (OCSORT)	75.73	74.69	84.83	69.04	84.29	56.59	69.13	70.97	73.45	74.00

Table 3
Execution time for model training and inference over ten collected videos in the CFIV dataset. The model training time is shown in seconds, and the model inference time is shown as the inference time divided by the video length. The results are obtained on one NVIDIA Tesla V100 SXM2 16 GB GPU.

		V1	V2	V3	V4	V5	V6	V7	V8	V9	V10
	Model training [s]	215.9	213.4	210.2	201.2	210.6	191.9	193.8	188.9	193.6	193.1
Stride = 1	Face recognition only	0.758	0.906	2.651	1.084	0.861	2.062	1.068	1.555	0.726	0.805
	ViCTer (StrongSORT)	1.706	1.816	4.798	2.253	2.085	4.353	2.103	2.695	1.776	1.851
	ViCTer (OCSORT)	0.864	1.018	2.845	1.208	0.992	2.261	1.178	1.671	0.844	0.922
Stride = 2	Face recognition only	0.393	0.474	1.332	0.559	0.446	1.054	0.551	0.790	0.383	0.422
	ViCTer (StrongSORT)	0.878	0.939	2.389	1.142	1.050	2.168	1.081	1.369	0.916	0.950
	ViCTer (OCSORT)	0.449	0.526	1.436	0.622	0.520	1.160	0.608	0.859	0.442	0.483
Stride = 4	Face recognition only	0.210	0.253	0.676	0.297	0.240	0.542	0.296	0.415	0.211	0.229
	ViCTer (StrongSORT)	0.458	0.488	1.219	0.591	0.541	1.101	0.561	0.714	0.491	0.497
	ViCTer (OCSORT)	0.241	0.283	0.739	0.329	0.281	0.604	0.324	0.448	0.244	0.262

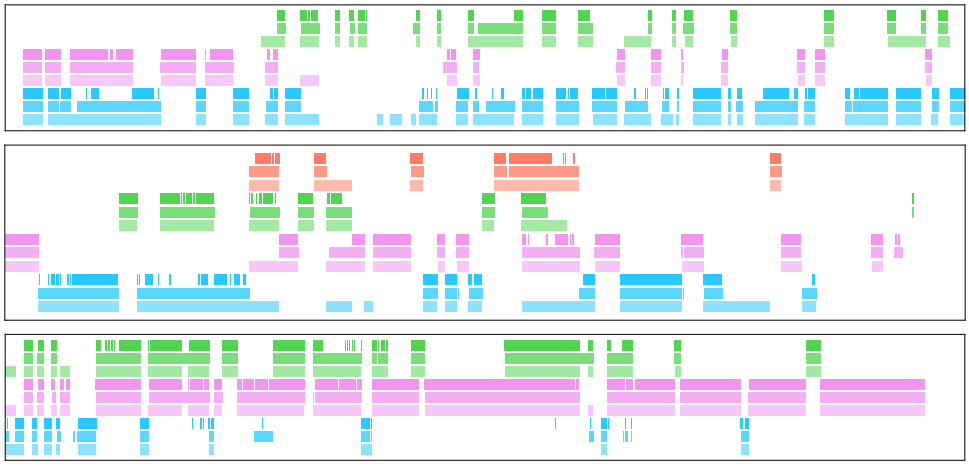

Fig. 13. Tracking results on three videos (V1, V2, and V3). Different colors correspond to the appearing time slots of different characters. For each character, the top line is the result of the baseline tracking method using a single face recognizer, the central line is the result of ViCTer using StrongSORT association algorithm, and the bottom line is the ground truth appearing time slot for that character. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

robustness to constant scene changes could help ViCTer with tracking accuracy.

In addition to combining a multi-human tracker with the face recognizer, there may be other possible ways, e.g., combining a keyframe extractor with the face recognizer, and only applying the recognizer to the keyframe to get the character appearing information for the whole time period represented by the keyframe. However, the current key frame extractor cannot guarantee that each character inside has a clear face for recognition. If a keyframe extractor can extract a keyframe with clear faces for characters inside, then combining it with a face recognizer may yield good results as well.

6. Conclusions

In order to find the appearing time slots for certain characters of interest in videos, this paper proposes a semi-supervised Video Character Tracker model (ViCTer) by combining a face recognition network and a multi-human tracker. To obtain better recognition accuracy, ViCTer leverages large amounts of unlabeled images from the video by employing semi-supervised learning methods and triplet loss to train the recognition network. To address the cases when characters do not show clear faces for recognition, ViCTer creatively aggregates the results of the face recognizer and the multi-human tracker to boost the tracking accuracy. Extensive evaluations of the ViCTer model are conducted on our collected dataset CFIV, and the experiment results showcase the performance of ViCTer along various dimensions. Specifically, ViCTer can achieve more than 98.5% face recognition accuracy and 70%~80% tracking accuracy using two labeled face images per character on CFIV. As for the execution time, ViCTer with the OCSORT algorithm can achieve near-real-time tracking performance with stride one for most videos in CFIV. We believe that this work will serve as a solid starting point for further investigations on the video character tracking problem.

Table A.4
Augmentations for faces in the labeled set.

Augmentations	Description
Augmentation 1	Random horizontal flip with probability = 0.5, color jitter with brightness = 0.5, random rotation with max angle = 5, and 16 × 16 Cutout
Augmentation 2	Random horizontal flip with probability = 0.5, color jitter with contrast = 0.5, random rotation with max angle = 5, and 16 × 16 Cutout
Augmentation 3	Random horizontal flip with probability = 0.5, color jitter with saturation = 0.5, random rotation with max angle = 5, and 16 × 16 Cutout
Augmentation 4	Random horizontal flip with probability = 0.5, random resize crop with scale = (0.8, 1) and ratio = (0.83, 1.2), random rotation with max angle = 5, and 16 × 16 Cutout
Augmentation 5	Resize 160 × 160 image to 100 × 100, then resize back to 160 × 160 for reduced resolution
Augmentation 6	Resize 160 × 160 image to 70 × 70, then resize back to 160 × 160 for reduced resolution
Augmentation 7	Resize 160 × 160 image to 50 × 50, then resize back to 160 × 160 for reduced resolution

Table B.5
Properties of each video in the CFIV dataset.

Video name	Video duration	Characters #	Testset available?	Country	Release year
Dirty Harry 1	00:04:33	3	No	USA	1971
Dirty Harry 2	00:02:42	4	Yes	USA	1971
Scent of a Woman 1	00:05:38	3	No	USA	1992
Scent of a Woman 2	00:06:57	3	Yes	USA	1992
The Pursuit of Happyness	00:04:14	3	No	USA	2006
Legally Blonde	00:05:23	2	No	USA	2001
Growing Pain I	00:04:57	3	No	China	2019
Growing Pain II	00:03:40	4	Yes	China	2022
All is Well 1	00:02:36	2	No	China	2019
All is Well 2	00:03:08	2	No	China	2019

Table C.6
Hyperparameter values for the face recognition experiment described in Section 4.2.

	Value	Description
B	8	Batch size of the augmented label face images.
μ	10	Scaling factor for the batch size of the unlabeled face images.
λ	1.0	Scaling factor for the training loss for the unlabeled face images.
c	0.99	Confidence threshold for generating pseudo labels.
α	1.0	Forced margin between positive and negative samples in triplet loss training.
lr_1	1.5×10^{-4}	Initial learning rate for backbone in stage one semi-supervised training.
lr_2	3×10^{-3}	Initial learning rate for backbone in stage two triplet loss training.
β	180	Scaling factor for the initial learning rate of fully connected layers in stage one.
n	10	Number of training epochs for stage one and stage two.

CRediT authorship contribution statement

Zilinghan Li: Methodology, Software, Visualization, Data curation, Writing – original draft. **Xiwei Wang:** Methodology, Validation, Data curation, Writing – review & editing. **Zhenning Zhang:** Software, Investigation, Data curation, Writing – review & editing. **Volodymyr Kindratenko:** Conceptualization, Resources, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

We have shared the link to our data and code at the Attached Files step

CHARACTER FACE IN VIDEO (Original data) (IEEE Dataport)
Research Code (Original data) (GitHub)

Acknowledgments

This work utilizes resources supported by the National Science Foundation's Major Research Instrumentation program, USA, grant #1725729, as well as the University of Illinois at Urbana-Champaign. All the experiments are run on the HAL cluster (Kindratenko et al., 2020) at the National Center for Supercomputing Applications. We are also thankful to Prof. Rini Bhattacharya Mehta from the University of Illinois at Urbana-Champaign for her guidance on the application.

Appendix A. Augmentation details

For each face image in the labeled set, we use the following seven augmentations listed in Table A.4 to obtain overall eight images per face. For augmentations 1 to 5, we combine common transformations such as contrast adjustment and small rotations with Cutout (DeVries & Taylor, 2017). Specifically, Cutout randomly covers the image with a small grey square to simulate the scenarios when faces are slightly blocked by obstacles. Augmentations 5 to 7 reduce the image resolution by resizing the 160 × 160 image to a smaller size to increase the model's ability for recognizing smaller face images.

As for each face image in the unlabeled set, the weak augmentation for it is simply the random horizontal flip, and strong augmentation uses a combination of RandAugment (Cubuk et al., 2020) and Cutout.

Appendix B. CFIV dataset properties

See Table B.5.

Appendix C. Experiment hyperparameters

See Table C.6.

References

Ahonen, T., Hadid, A., & Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(12), 2037–2041.

Bachman, P., Alsharif, O., & Precup, D. (2014). Learning with pseudo-ensembles. In *Advances in neural information processing systems*.

Belhumeur, P. N., Hespanha, J. P., & Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *19*(7), 711–720.

Berthelot, D., Carlini, N., Cubuk, E. D., Kurakin, A., Sohn, K., Zhang, H., & Raffel, C. (2020). ReMixMatch: Semi-supervised learning with distribution matching and augmentation anchoring. In *8th international conference on learning representations, ICLR*.

Berthelot, D., Carlini, N., Goodfellow, I. J., Papernot, N., Oliver, A., & Raffel, C. (2019). MixMatch: A holistic approach to semi-supervised learning. In *Advances in neural information processing systems*.

Bewley, A., Ge, Z., Ott, L., Ramos, F. T., & Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing, ICIP 2016, Phoenix, AZ, USA, September 25-28, 2016* (pp. 3464–3468). IEEE.

Breitenstein, M. D., Reichlin, F., Leibe, B., Koller-Meier, E., & Van Gool, L. (2009). Robust tracking-by-detection using a detector confidence particle filter. In *2009 IEEE 12th international conference on computer vision* (pp. 1515–1522). IEEE.

Butt, A. A., & Collins, R. T. (2013). Multi-target tracking by lagrangian relaxation to min-cost network flow. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1846–1853).

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age. In *2018 13th IEEE international conference on automatic face gesture recognition (FG 2018)* (pp. 67–74).

Cao, J., Weng, X., Khirodkar, R., Pang, J., & Kitani, K. (2022). Observation-Centric SORT: Rethinking SORT for robust multi-object tracking. arXiv:2203.14360.

Cao, Z., Yin, Q., Tang, X., & Sun, J. (2010). Face recognition with learning-based descriptor. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 2707–2714). IEEE.

Chan, T.-H., Jia, K., Gao, S., Lu, J., Zeng, Z., & Ma, Y. (2015). PCANet: A simple deep learning baseline for image classification? *IEEE Transactions on Image Processing*, *24*(12), 5017–5032.

Cubuk, E. D., Zoph, B., Shlens, J., & Le, Q. V. (2020). Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops* (pp. 702–703).

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). Ieee.

DeVries, T., & Taylor, G. W. (2017). Improved regularization of convolutional neural networks with cutout. arXiv:1708.04552.

Dicle, C., Camps, O. I., & Sznaier, M. (2013). The way they move: Tracking multiple targets with similar appearance. In *Proceedings of the IEEE international conference on computer vision* (pp. 2304–2311).

Du, Y., Song, Y., Yang, B., & Zhao, Y. (2022). Strongsort: Make deepsort great again. arXiv:2202.13514.

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

Good, I. J. (1952). Rational decisions. *Journal of the Royal Statistical Society. Series B. Statistical Methodology*, *14*(1), 107–114.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative adversarial networks. *Communications of the ACM*, *63*(11), 139–144.

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision* (pp. 2961–2969).

He, X., Yan, S., Hu, Y., Niyogi, P., & Zhang, H.-J. (2005). Face recognition using laplacianfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *27*(3), 328–340.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Huang, G. B., Mattar, M., Berg, T., & Learned-Miller, E. (2008). Labeled faces in the wild: A database forstudying face recognition in unconstrained environments. In *Workshop on faces in 'real-life' images: Detection, alignment, and recognition*.

Jocher, G. (2021). ultralytics/yolov5: v6.0-YOLOv5n 'Nano' models, Roboflow integration, TensorFlow export, OpenCV DNN support.

Kalman, R. E. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*, 83–97.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, *82*(1), 35–45.

Kim, B., Choo, J., Kwon, Y., Joe, S., Min, S., & Gwon, Y. (2021). SelfMatch: Combining contrastive self-supervision and consistency for semi-supervised learning. CoRR, abs/2101.06480.

Kim, C., Li, F., Ciptadi, A., & Rehg, J. M. (2015). Multiple hypothesis tracking revisited. In *Proceedings of the IEEE international conference on computer vision* (pp. 4696–4704).

Kindratenko, V., Mu, D., Zhan, Y., Maloney, J., Hashemi, S. H., Rabe, B., Xu, K., Campbell, R., Peng, J., & Gropp, W. (2020). HAL: Computer system for scalable deep learning. In *Practice and experience in advanced research computing* PEARC '20, (pp. 41–48). New York, NY, USA: Association for Computing Machinery.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, *25*.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84–90.

Laine, S., & Aila, T. (2017). Temporal ensembling for semi-supervised learning. In *5th international conference on learning representations*.

Lee, D. (2013). Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *ICML 2013 workshop: Challenges in representation learning (WREPL)*.

Li, Z., Wang, X., Zhang, Z., & Kindratenko, V. (2022). Character face in video.

Lin, T., Maire, M., Belongie, S. J., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common objects in context. In *Lecture notes in computer science: vol. 8693, Computer vision - ECCV 2014 - 13th European conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V* (pp. 740–755). Springer.

Liu, C., & Wechsler, H. (2002). Gabor feature based classification using the enhanced fisher linear discriminant model for face recognition. *IEEE Transactions on Image Processing*, *11*(4), 467–476.

Loshchilov, I., & Hutter, F. (2017). SGDR: Stochastic gradient descent with warm restarts. In *5th international conference on learning representations*. ICLR.

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T.-K. (2021). Multiple object tracking: A literature review. *Artificial Intelligence*, *293*, Article 103448.

McInnes, L., Healy, J., & Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. arXiv:1802.03426.

Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep face recognition.

Pirsiavash, H., Ramanan, D., & Fowlkes, C. C. (2011). Globally-optimal greedy algorithms for tracking a variable number of objects. In *CVPR 2011* (pp. 1201–1208). IEEE.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 779–788).

Rezatofighi, S. H., Milan, A., Zhang, Z., Shi, Q., Dick, A., & Reid, I. (2015). Joint probabilistic data association revisited. In *Proceedings of the IEEE international conference on computer vision* (pp. 3047–3055).

Rodriguez, M., Sivic, J., Laptev, I., & Audibert, J.-Y. (2011). Data-driven crowd analysis in videos. In *2011 international conference on computer vision* (pp. 1235–1242). IEEE.

Sajjadi, M., Javanmardi, M., & Tasdizen, T. (2016). Regularization with stochastic transformations and perturbations for deep semi-supervised learning. In *Advances in neural information processing systems*.

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. In *IEEE conference on computer vision and pattern recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015* (pp. 815–823). IEEE Computer Society.

Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C., Cubuk, E. D., Kurakin, A., & Li, C. (2020). FixMatch: Simplifying semi-supervised learning with consistency and confidence. In *Advances in neural information processing systems*.

Song, B., Jeng, T.-Y., Staudt, E., & Roy-Chowdhury, A. K. (2010). A stochastic graph evolution framework for robust multi-target tracking. In *Computer vision – ECCV 2010* (pp. 605–619). Berlin, Heidelberg: Springer Berlin Heidelberg.

Sun, Y., Wang, X., & Tang, X. (2014). Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1891–1898).

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1701–1708).

Tarvainen, A., & Valpola, H. (2017). Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Advances in neural information processing systems*.

Turk, M., & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, *3*(1), 71–86.

Van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. *Journal of Machine Learning Research*, *9*(11).

Wang, M., & Deng, W. (2021). Deep face recognition: A survey. *Neurocomputing*, *429*, 215–244.

Wang, Z., Zheng, L., Liu, Y., Li, Y., & Wang, S. (2020). Towards real-time multi-object tracking. In *European conference on computer vision* (pp. 107–122). Springer.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing* ICIP, (pp. 3645–3649).

Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S., & Ma, Y. (2008). Robust face recognition via sparse representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *31*(2), 210–227.

Xie, Q., Dai, Z., Hovy, E. H., Luong, T., & Le, Q. (2020). Unsupervised data augmentation for consistency training. In *Advances in neural information processing systems*.

Xie, Q., Luong, M.-T., Hovy, E. H., & Le, Q. V. (2020). Self-training with noisy student improves ImageNet classification. In *2020 IEEE/CVF conference on computer vision and pattern recognition*.

Xing, J., Ai, H., & Lao, S. (2009). Multi-object tracking through occlusions by local tracklets filtering and global tracklets association with detection responses. In *2009 IEEE conference on computer vision and pattern recognition* (pp. 1200–1207). IEEE.

Yang, B., & Nevatia, R. (2012). Multi-target tracking by online learning of non-linear motion patterns and robust appearance models. In *2012 IEEE conference on computer vision and pattern recognition* (pp. 1918–1925). IEEE.

Yoon, J. H., Yang, M.-H., Lim, J., & Yoon, K.-J. (2015). Bayesian multi-object tracking using motion context from multiple objects. In *2015 IEEE winter conference on applications of computer vision* (pp. 33–40). IEEE.

Yuan, X., Li, Z., & Wang, G. (2022). Activematch: End-to-end semi-supervised active representation learning. In *2022 IEEE international conference on image processing* ICIP, (pp. 1136–1140). IEEE.

Zhang, L., & van der Maaten, L. (2013). Structure preserving object tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. CVPR.

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, *23*(10), 1499–1503.

Zhao, Y., Xiong, Y., Wang, L., Wu, Z., Tang, X., & Lin, D. (2017). Temporal action detection with structured segment networks. In *Proceedings of the IEEE international conference on computer vision* (pp. 2914–2923).