# Learning Video Localization on Segment-Level Video Copy Detection with Transformer

Chi Zhang[1,2], Jie Liu[2], Shuwu Zhang[3], Zhi Zeng[2,3], and Ying Huang[3](✉)

[1] University of Chinese Academy of Sciences, Beijing, China
[2] Institute of Automation, Chinese Academy of Sciences, Beijing, China
[3] Beijing University of Posts and Telecommunications, Beijing, China
ying.huang@bupt.edu.cn

**Abstract.** At present, research on segment-level video copy detection algorithms mainly focuses on end-to-end optimization from key frame selection and feature extraction to similarity pattern detection, causing the deployment of such algorithms to be difficult and expensive, and ignoring specific research on optimizing detectors for similarity pattern detection. To address the above issues, we propose the segment-level Video Copy Detection Transformer (VCDT), a transformer-based detector designed for similarity pattern detection. Its main novelty can be summarized by two points: (1) An anchor training strategy that allows the model to use the positional prior information in the anchor boxes to make predictions more precisely, (2) A query adaptation module to fine-tune the anchor boxes dynamically. Our experiments show that, without bells and whistles, VCDT achieves state-of-the-art performance while showing an impressive convergence speed.

**Keywords:** Video Copy Localization · Content Based Video Retrieval · Temporal Alignment

## 1 Introduction

Segment-level video copy detection, also known as video copy localization, aims to locate the start and end times of the copied segments in a pair of potentially copied videos. As the phenomenon of copying emerges endlessly in many videos, segment-level video copy detection has turned into one of the most urgent technologies in copyright protection. An efficient and popular pipeline has been summarized in [11]. As shown in Fig. 1, first, the two videos get their respective key frames through key frame selection, and then the features of frames are obtained through feature extraction. After that, cosine similarities of the features are calculated to generate the frame-to-frame similarity matrix. Finally, by detecting similarity patterns of copied segments on the matrix, the video copy localization system can give predictions of the start and end timestamps of the copied segments. Traditional methods for similarity pattern detection include Temporal

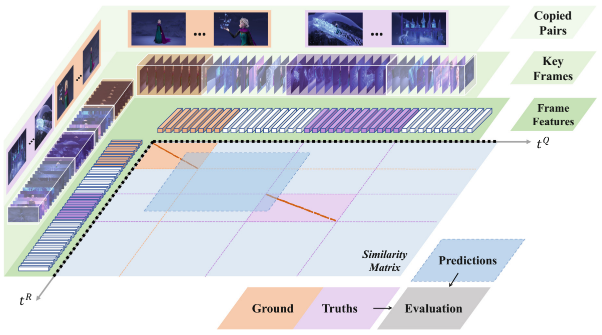


**Fig. 1.** A popular video copy localization pipeline nowadays. Elements marked in dark red on the similarity matrix indicate that they have a greater value than surrounding elements. (Color figure online)

Networks [13,18,19], Dynamic Programming [2], Temporal Hough Voting [4,13], etc. In recent years, due to the superior performance of deep object detectors, many works such as [8,11] have replaced these traditional methods with deep detection models to detect similarity patterns on the similarity matrix.

However, there are some omissions in the current research. First, as the performance improvement of object detection algorithms slows down [3], the focus of research has become to simultaneously optimize key frame extraction, feature extraction, and similarity pattern detection to seek the end-to-end optimum to improve performance [6,8,11,19]. But most of the video databases store the extracted key frame features, so this kind of algorithms are difficult to completely deploy, leading to performance limitations. Second, most video copy localization algorithms directly use existing deep object detectors for similarity pattern detection, but these detectors are usually designed for detecting natural objects in the real world. From the two points above, it is necessary to modify the general object detectors to make them more suitable for detecting similarity patterns.

To address the two issues mentioned above, we propose VCDT: the segment-level **V**ideo **C**opy **D**etection **T**ransformer, a transformer-based detector to detect similarity patterns of copied segments on similarity matrix. The VCDT is a DETR-like model, since it is improved from DINO [22], which is a variant of DETR [1]. There are 2 main contributions in VCDT:

- **Anchor training:** We propose a novel method to introduce anchor boxes into transformer detector for similarity pattern detection. In this way, the positional prior information in anchor boxes effectively improves the performance of the model. Besides, with this strategy, the Hungarian matcher can be removed to speed up the convergence of the model.
- **Query adaptation module:** We propose a query adaptation module to dynamically adjust anchor boxes. In this way, the anchor box can provide a more accurate initial position and size for the final prediction, which further improves the performance of the model.

Our experiment results show that, without bells and whistles, VCDT outperforms the previous state-of-the-art (SOTA) method without optimizing key frame selection and feature extraction. In addition, our method yields an impressive convergence speed.

## 2 Related Work

Video copy detection is one of the important tasks in multimedia retrieval. Datasets like FIVR [14] and SVD [12] are commonly used benchmarks of video-level copy detection algorithms. Since 2014, the most widely used segment-level video copy detection dataset is VCDB [13], which contains more than 9,000 copied segment pairs. VCSL [9] proposed by He et al. in 2022 is the largest dataset for segment-level video copy detection, including 280k copied segments.

As for video copy localization algorithms, common traditional methods are Temporal Networks [13,18], Dynamic Programming [2], and Temporal Hough Voting [4,13], etc. [10] formulated the similarity pattern detection as an object detection problem, [11] inherited this way, and proposed a novel frame extraction method, so that the model can be trained end-to-end. Besides, both [8] and [19] use attention layer to enhance frame features to improve performance.

For transformer [20] detectors, DETR [1] is the earliest transformer-based detector. There are also attempts on other structures of transformer-based detectors, and a typical work is ViTDet [15]. Anchor-DETR [21] uses anchor boxes to introduce prior information, which is close to our method. Besides, some vision transformers also perform well on detection tasks, such as Swin Transformer [16]. Deformable-DETR [23] proposed deformable attention, thus greatly boosting up the convergence speed. Based on Deformable-DETR, Zhang et al. proposed DINO [22], which is currently the most powerful DETR variant.

## 3 Method

### 3.1 Problem Formulation

Given a pair of potentially copied videos $(V^Q, V^R) = (\{f_n^Q\}_{n=1}^{N^Q}, \{f_n^R\}_{n=1}^{N^R})$, where $Q$ and $R$ denote query video and reference video, $N$ is the number of video frames, and $f$ is the frame feature. Usually, the videos are organized as matrices $V = \{f_n^Q\}_{n=1}^{N^Q} \in R^{N\times d}$, and each row is a normalized feature vector. The format of annotations is $[(i_{s1}^Q, i_{e1}^Q, i_{s1}^R, i_{e1}^R), (i_{s2}^Q, i_{e2}^Q, i_{s2}^R, i_{e2}^R, \ldots]$, where $sn$ and $en$ are the start and end frames of the $n$-th copied segment, and $i$ represents the frame index. The frame to frame similarity matrix will be:

$$S = V^Q(V^R)^T = (s_{i,j}) \in R^{N^Q\times N^R} \tag{1}$$

Then, the similarity pattern detection is executed on the matrix, and the detector can give the prediction as one or several bounding box(s), whose four coordinate values are just the prediction of a copied segment.

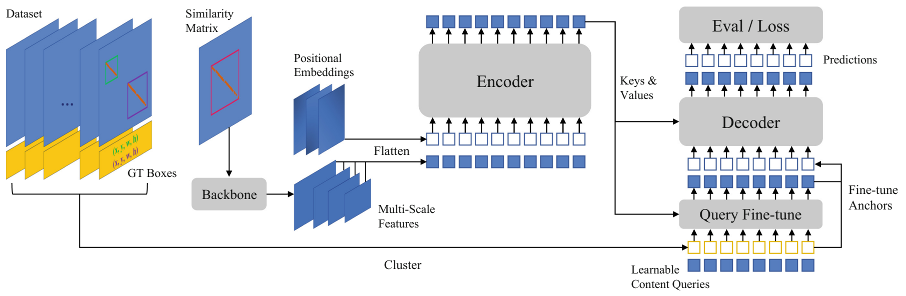


**Fig. 2.** Overview of our method. The solid and hollow squares represent content queries and positional queries respectively. The “Query Fine-tune” is our proposed query adaptation module, which is responsible for boosting the initial queries.

## 3.2 Model Overview

VCDT is mainly modified from DINO [22], and just like other DETR-like models, it contains a multi-layer transformer encoder, a multi-layer transformer decoder, and a MLP prediction head. Figure 2 shows the pipeline of our method. Given a similarity matrix, the backbone like ResNet [7] treats it as an image and yields multi-scale features. Then, features with corresponding positional embeddings are fed into the encoder to be enhanced. Instead of initializing reference boxes from encoder output as DINO, the anchor boxes clustered from the training data will serve as the initial reference boxes in our method. In addition, we add a query adaptation module before decoder, which initializes decoder queries and fine-tunes the anchor boxes. Finally, the decoder outputs the prediction bounding boxes by refining the initial reference boxes layer by layer, and each prediction has a confidence score.

## 3.3 Anchor Training

It has been proven in classic deep detection models like YOLO [5], that the anchor boxes can improve the performance of model by introducing positional prior information. As shown in Fig. 1, the similarity patterns basically follow the same two characteristics: **a.** Appearing as a path on, or near the main or auxiliary diagonal of the similarity matrix. **b.** The element values on the path are greater than the surrounding elements. Therefore, similarity pattern detection can be formulated as a single-class object detection problem.

The most representative work of introducing anchor boxes into the DETR architecture is Anchor-DETR [21]. However, since Anchor-DETR is a general detector, it is designed not to be sensitive to the number of classes, and its anchor boxes are manually set without obtaining prior information from data, because the positional prior information obtained from multiple classes will confuse each other and lose the ability to predict a certain category's target. As shown in Fig. 2, we take the cluster centers of training data as the anchor boxes:

$$A = Cluster(tgt) \tag{2}$$

where $A \in R^{N^q \times 4}$ denotes anchor boxes. $tgt \in R^{N^t \times 4}$ denotes all the ground truth bounding boxes in training set. $N^q$ is the number of decoder queries. Note that the anchor boxes are set as parameters of the model, so that the model can utilize the positional prior information in them during inference.

In a single input, namely a similarity matrix, there may be more than one target that corresponds to the same anchor box. In order to handle this, Anchor-DETR [21] sets $N^p$ patterns for each decoder query, so that one query can predict $N^p$ boxes. However, when the number of targets belonging to the same anchor is greater than $N^p$, there will be predictions refined by other anchors are selected by the Hungarian matcher to calculate the loss with the extra targets, thus misleading the learning process. Here, our solution is to merge the targets belonging to the same anchor:

$$t_k = Merge(\{t_{k,j}\}_{j=1}^{N^k}) \tag{3}$$

$$t_k^i = \begin{cases} min(\{t_{k,j}^i\}_{j=1}^{N^k}), & i \in \{0,1\} \\ max(\{t_{k,j}^i\}_{j=1}^{N^k}), & i \in \{2,3\} \end{cases} \tag{4}$$

where $k$ denotes the $k$-th anchor box, $t_k$ denotes the merged target, and $N^k$ is the number of all targets belonging to this anchor box in the current training sample. $t_{k,j}$ means the $j$-th target belongs to the $k$-th anchor. Note that both in Eqs. 3 and 4, $t_k$ and $t_{k,j}$ are of the form $(x_0, y_0, x_1, y_1)$, which means the first two dimensions are normalized coordinates of left top, and the last two dimensions are right bottom. Considering it is important to recall all the copied segments in copyright protection, we design this $Merge$ function so that when there are multiple targets corresponding to one anchor, a prediction wrapping these target boxes will be output to recall them all.

Additionally, because each target has been bound to its cluster center, that is, the anchor box, in the clustering process, the Hungarian matcher is no longer needed to match predictions and targets, which makes the convergence of the model being greatly faster than the original DINO [22].

### 3.4 Query Adaptation Module

Just as DETR [1] and almost all its variants follow, the decoder query consists of two parts, which respectively contain the positional information of the initial reference box and the content information of current input:

$$Q = Q^c + Q^p \tag{5}$$

where $Q \in R^{N^q \times C}$ is the decoder queries, and $C$ denotes the dimension of the query. Superscript $c$ and $p$ stand for content query and positional query. One thing that's very important is the initialization of queries $Q$. Usually $Q^p$ is obtained by sine-cosine encoding the initial reference boxes, so the problem of initializing $Q^p$ is how to initialize the reference boxes:

$$Q^{p,init} = Sine(Ref^{init}) \tag{6}$$

where $Ref^{init} \in R^{N^q \times 4}$ denotes initial reference boxes, and $Sine$ is the sine-cosine encoding. DINO [22] initializes $Ref^{init}$ from the encoder output embeddings, and leaves the $Q^c$ as model's parameters and learnable:

$$Q^c = Embedding(N^q, C) \tag{7}$$

Recall Sect. 3.3 and Fig. 2, in our model, $Ref^{init}$ is the anchor boxes, namely $A$ in Eq. 2, lacking the information of current input similarity matrix included in encoder output embeddings. Therefore, we add a query adaptation module before the decoder, which is composed of a cross-attention layer, a self-attention layer and a feed-forward network:

$$Q^f = FFN(MHA(MSDA(Q, K, V))) \tag{8}$$

where

$$Q = Q^c + Q^p = Embedding(N^q, C) + Sine(A) \tag{9}$$

$$K = V = encoder\_output \tag{10}$$

In Eq. 8, $MSDA$ is the multi-scale deformable attention layer that implements cross-attention, $MHA$ is the multi-head attention layer to implement self-attention, and $FFN$ is the feed-forward network. In Eq. 9, we set content queries $Q^c$ the same as Eq. 7, and positional queries $Q^p$ is the sine-cosine embedding of anchor boxes in Eq. 2. In Eq. 10, the $encoder\_output \in R^{H \times W \times C}$ serves as the keys and values in Eq. 8. At last, after $Q^f \in R^{N^q \times C}$ pools the information of current input similarity matrix from encoder output, it is used to fine-tune the anchor boxes, and serves as the content queries of decoder:

$$Q^d = Q^{d,c} + Q^{d,p} \tag{11}$$

where

$$Q^{d,c} = Q^f \tag{12}$$

$$Q^{d,p} = Sine(Encode(Q^f) + A) \tag{13}$$

Superscripts $c$ and $p$ denote content and positional query respectively, as always. $A$ is the anchor boxes as in Eq. 2, and the function $Encode$ is an auxiliary prediction layer in the query adaptation module to predict the offset of $A$ from $Q^f$. Finally, the queries that will be feed into decoder are the $Q^d \in R^{N^q \times C}$ in Eq. 11, which contain not only the content information of the current input similarity matrix, but also the positional prior information from the training set.

## 3.5 Learning Objective

We adopt the same configuration as DINO [22] to optimize the model: L1 loss and GIOU loss for box regression and focal loss for classification.

**Table 1.** Main results of the comparison on VCSL dataset. * indicates that the performance of these methods are published by the VCSL official team in [8]. † indicates that these methods follow the same hyperparameters, and the performance are taken from the convergence point.

| Method | HV* | TN* | DP* | DTW* | SPD* | TransVCL* | DINO† | Deformable DETR† | VCDT (ours)† |
|---|---|---|---|---|---|---|---|---|---|
| Recall | 86.94 | 75.25 | 49.98 | 45.10 | 56.49 | 65.59 | 64.78 | 64.89 | 65.70 |
| Precision | 36.82 | 51.80 | 60.61 | 56.67 | 68.60 | 67.46 | 67.83 | 67.19 | 70.34 |
| F1-score | 51.73 | 61.36 | 54.48 | 50.23 | 61.96 | 66.51 | 66.27 | 66.02 | **67.94** |

**Table 2.** Comparison of the video-level performance.

| Method | FRR | FAR | F1-score |
|---|---|---|---|
| SPD | 0.2974 | 0.0958 | 79.08 |
| TransVCL | 0.1666 | 0.0173 | 90.19 |
| VCDT | 0.1594 | 0.0157 | **90.68** |

**Table 3.** Comparison on VCDB. * indicates fixed anchors.

| Method | Recall | Precision | F1-score |
|---|---|---|---|
| TransVCL | 76.69 | 74.09 | 75.37 |
| VCDT* | 77.12 | 75.01 | 76.05 |
| VCDT | 77.37 | 75.34 | **76.34** |

## 4 Experiments

### 4.1 Datasets and Evaluation Metrics

We use the VCSL dataset [9] due to its remarkably large scale. Besides, we adopt VCDB [13] to verify the generalization ability of the anchor boxes.

For evaluation metrics, we use the novel method proposed in VCSL [9] to calculate recall and precision. Rather than metrics like Intersection over Union (IOU) that are used in object detection, this metric can better reflect the performance of video copy localization. Besides, we adopt F1-score to reflect the overall performance. Because there are many distractive samples in the testing set that do not have any copied segments, we also adopt False Rejection Rate (FRR) and False Alarm Rate (FAR) to evaluate the video-level retrieval performance.

### 4.2 Implementation Details

Experiments are performed on a single NVIDIA GeForce RTX 3090. The cluster method is K-Means, and the number of queries, namely the number of anchor boxes, 100. Backbone is ResNet50 [7], which extracts 4-scale features. The optimizer is AdamW, and the initial learning rate is $1 \times 10^{-4}$. Batch size of 8, and weight decay $1 \times 10^{-4}$. The confidence threshold of valid predictions is 0.32.

### 4.3 Main Results

**Performance.** We compare the performance of our method with several typical algorithms. There are four traditional methods, including Hough Voting (HV), Dynamic Programming (DP), Temporal Network (TN) and Dynamic Time Warping (DTW). Two CNN-based methods SPD [11] and TransVCL [8]

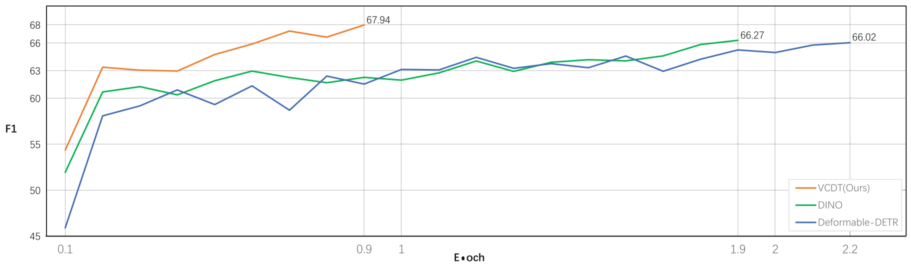


**Fig. 3.** Convergence curves on VCSL of the three transformer-based methods.

(SOTA). In addition, since our method is mainly improved from DINO [22], and DINO is derived from Deformable-DETR [23], we also add them in the comparison. The main results are shown in Table 1, where we can see that our method (VCDT) outperforms all other methods. Compared with the previous SOTA method TransVCL, our method (VCDT) has a significant improvement in the F1-score (+1.43%), reaching 67.94%. Especially in terms of precision, our method reaches 70.34%, having strong advantages over TransVCL (+2.88%) and SPD (+1.74%), both of which use CNN detectors. For the video-level performance reflected by FRR and FAR, our method exceeds the previous SOTA TransVCL by +0.49% F1 as shown in Table 2. As for the other two transformer-based detectors, our method improves +1.92% and +1.67% F1 respectively compared to Deformable-DETR [23] and DINO [22].

**Convergence Speed.** During the experiment, we found that the transformer-based models converge very fast on the dataset in 20-epoch setting experiment. Therefore, in order to verify the effect of our strategies on improving the convergence speed, we save the models every 0.1 epoch and verify their performance on the testing set. The convergence curves are plotted in Fig. 3. Our method provides an impressive convergence speed, and it reaches convergence in less than 1 epoch, which means 2.1× faster than the original DINO (1.9 epochs) and 2.4× than the Deformable-DETR (2.2 epochs). Moreover, compared with the method SPD [11], which also does not enhance the frame features and similarity matrices as we do, our method reaches 63.35% F1 at only 0.2 epoch, and leads SPD (61.96%) by +1.39% F1. This means that VCDT only needs 2442 iterations, or 19536 training samples to defeat the CNN-based method significantly.

**Generalization.** Since the anchor boxes are obtained from the training data, it is necessary to verify the generalization ability of them on different benchmark. We trained two versions of VCDT on VCDB [13], one of which fixes the anchor boxes obtained from VCSL [9], and the other one is trained from scratch. The results are shown in Table 3, and it is intuitive that the version trained from scratch performs better. However, the performance gap (0.29% F1) between the

**Table 4.** Ablation comparison of our proposed strategies. "AT" and "QAM" represent our anchor training and query adaptation module strategies. In the "Epoch" column are convergence points of the methods.

| #. Method | Epoch | Recall | Precision | F1-score |
|---|---|---|---|---|
| 1. DINO (baseline) | 1.9 | 64.78 | 67.83 | 66.27 |
| 2. +AT | 1.1 | 65.52 | 67.53 | 66.46 |
| 3. +AT+QAM (ours) | 0.9 | 65.70 | 70.34 | 67.94 |

two versions is acceptable, indicating the generalization ability of anchor boxes. In addition, both versions are stronger than TransVCL (SOTA) [8].

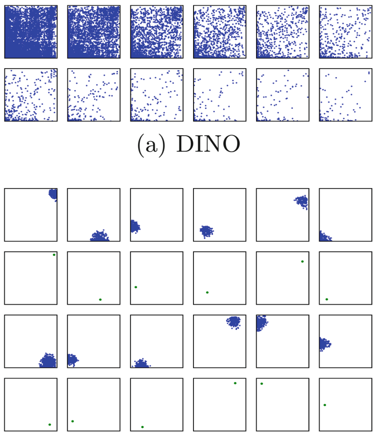
## 4.4 Ablation

The results of our ablation study are shown in Table 4. DINO [22] is set as the baseline, since our method is mainly modified from it. Note that the query adaptation module cannot be separated from the anchor training to be an independent strategy. Compared with the baseline, the anchor training reduces the convergence point from 1.9 epochs to 1.1 epochs, which means a 73% increase in convergence speed. In addition, the recall is obviously increased (+0.74%), which shows the effect of the merge function in Eq. 4. However, compared to the reference boxes from the encoder output in DINO, the anchor boxes obtained from training data lack information about the current input, despite including prior information, so the precision decreases slightly. Still, anchor boxes brings performance improvement (+0.19% F1) to the model and obviously accelerates the convergence, illustrating the superiority of positional prior information.

After appending the query adaptation module before decoder, the anchor boxes can provide more accurate initial reference positions and sizes, so that the overall performance of model is further improved, and achieves state-of-the-art.

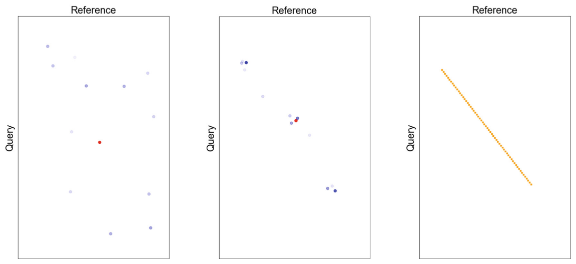
## 4.5 Visualization

We combine visualization to analyze the mechanism behind the performance growth. For DINO [22], since the training process is a single-class object detection task, in the initial stage of the learning, the model tends to use only a part of decoder queries to predict targets, and these queries therefore learn more information from the backpropagation. Then, the Hungarian matcher always matches targets with the predictions from these queries, because their predictions have higher confidence scores. Finally, there is an unbalanced learning between queries which result in a waste of parameters. In the PyTorch [17] implementation, the queries that dominate the initial learning stage are those in the front of the decoder, which have the smallest indexes. The above analysis can be easily verified in Fig. 4. In DINO, as the index increases, there is a very large gap between the number of predictions output by the subsequent queries and the previous



(b) VCDT(Ours)

**Fig. 4.** Visualization of the predictions from the first 12 decoder queries of DINO and VCDT. The blue points represent the centers of prediction boxes, and the green points are centers of the anchor boxes corresponding to each query. (Color figure online)



(a) DINO (b) VCDT(Ours) (c) Ground Truth

**Fig. 5.** Visualization of the cross-attention. The red points are reference points, and the blue points are sampling locations, where the darker points have greater attention. (Color figure online)

ones. With the anchor training strategy, the predictions are more concentrated and distributed near the corresponding anchor boxes, and the numbers of predictions between queries are also on the same level, which means VCDT can make full use of all decoder queries to predict the targets.

We also observed that our strategy makes the model more sensitive to the similarity patterns. Take the test sample in Fig. 5 as an example, where both DINO [22] and VCDT have a valid output with nearly identical confidence, and we visualize the cross-attention in the last decoder layer. The reference point marked in red is the center of the current reference box, and the sampling locations marked in blue are the positions on the input similarity matrix that the attention layer focuses on. It can be observed that the reference point of

VCDT is closer to the center of the ground truth than DINO. In Fig. 5(a), even at the last decoder layer, the cross-attention is still searching for the target in a lot of irregular positions. This is because DINO uses encoder output to initialize reference boxes, causing the initial reference box's center change frequently at different inputs, and this didn't happen in VCDT thanks to the anchor boxes. With a stable initial position provided by anchor boxes, cross-attention in VCDT learns a specific rule for finding similarity patterns. In Fig. 5(b), the sampling locations proposed by VCDT are mostly on the line from upper left to lower right, which just looks like the ground truth. Besides, points near the reference point, and the ones at top-left and right-bottom are given greater attention weights. Compared with DINO, the cross-attention of VCDT is obviously better at finding the centers and starting and ending points of similarity patterns.

## 5 Conclusion

In this paper, we propose VCDT, a transformer-based detector for video copy localization. Our method leads the previous SOTA by a large margin. In the future, we hope that other researchers will be interested in further exploring the potential of transformer architecture in segment-level video copy detection.

**Acknowledgement.** This work was supported by the National Key R&D Program of China under Grant 2021YFF0901604.

## References

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 213–229. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
2. Chou, C.L., Chen, H.T., Lee, S.Y.: Pattern-based near-duplicate video retrieval and localization on web-scale videos. IEEE Trans. Multimedia **17**(3), 382–395 (2015)
3. Code, P.W.: Object detection on coco test-dev (2018). https://paperswithcode.com/sota/object-detection-on-coco
4. Douze, M., Jégou, H., Schmid, C.: An image-based approach to video copy detection with spatio-temporal post-filtering. IEEE Trans. Multimedia **12**(4), 257–266 (2010)
5. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: YOLOX: exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021)
6. Han, Z., He, X., Tang, M., Lv, Y.: Video similarity and alignment learning on partial video copy detection. In: Proceedings of the 29th ACM International Conference on Multimedia, pp. 4165–4173 (2021)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 770–778 (2016)
8. He, S., et al.: TransVCL: attention-enhanced video copy localization network with flexible supervision. arXiv preprint arXiv:2211.13090 (2022)

9. He, S., et al.: A large-scale comprehensive dataset and copy-overlap aware evaluation protocol for segment-level video copy detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 21086–21095 (2022)
10. Hu, Y., Mu, Z., Ai, X.: STRNN: end-to-end deep learning framework for video partial copy detection. In: Journal of Physics: Conference Series, vol. 1237, p. 022112. IOP Publishing (2019)
11. Jiang, C., et al.: Learning segment similarity and alignment in large-scale content based video retrieval. In: Proceedings of the 29th ACM International Conference on Multimedia, pp. 1618–1626 (2021)
12. Jiang, Q.Y., He, Y., Li, G., Lin, J., Li, L., Li, W.J.: SVD: a large-scale short video dataset for near-duplicate video retrieval. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 5281–5289 (2019)
13. Jiang, Y.-G., Jiang, Y., Wang, J.: VCDB: a large-scale database for partial copy detection in videos. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8692, pp. 357–371. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10593-2_24
14. Kordopatis-Zilos, G., Papadopoulos, S., Patras, I., Kompatsiaris, I.: FIVR: fine-grained incident video retrieval. IEEE Trans. Multimedia **21**(10), 2638–2652 (2019)
15. Li, Y., Mao, H., Girshick, R., He, K.: Exploring plain vision transformer backbones for object detection. arXiv preprint arXiv:2203.16527 (2022)
16. Liu, Z., et al.: Swin transformer: hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10012–10022 (2021)
17. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
18. Tan, H.K., Ngo, C.W., Hong, R., Chua, T.S.: Scalable detection of partial near-duplicate videos by visual-temporal consistency. In: Proceedings of the 17th ACM International Conference on Multimedia, pp. 145–154 (2009)
19. Tan, W., Guo, H., Liu, R.: A fast partial video copy detection using KNN and global feature database. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 2191–2199 (2022)
20. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems, vol. 30 (2017)
21. Wang, Y., Zhang, X., Yang, T., Sun, J.: Anchor DETR: query design for transformer-based detector. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 36, pp. 2567–2575 (2022)
22. Zhang, H., et al.: DINO: DETR with improved denoising anchor boxes for end-to-end object detection. arXiv preprint arXiv:2203.03605 (2022)
23. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)