

TDK-thesis

Adrián Szlatincsán

Developing a Semi-supervised Clustering Transformer for Video Instance Segmentation

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPT. OF SOFTWARE TECHNOLOGY AND METHODOLOGY



Author:

Adrián Szlatincsán

Artificial Intelligence MSc

II. grade

Supervisor:

Áron Fóthi

PhD Student

Budapest, 2024

Abstract

This research project is about applying advanced computer vision techniques by proposing the development of a Mask2Former based Semi-supervised Clustering Transformer tailored for Video Instance Segmentation (VIS), which we call Cluster2Former. The research seeks to investigate the model's ability to perform pixel clustering – putting pixels of an object into the same cluster - without altering its fundamental architecture. The primary goal is to enrich the model's understanding of segmentation tasks, enhance its pixel clustering capabilities with the integration of similarity-based pixel clustering loss and heuristics while using the original transformer-based architecture. The research also involves the creation of scribble like, reduced annotation datasets derived from existing databases, effectively simulating a scenario with sparse labeled data. These created datasets have only a very small part of the annotation (0.5%) of the original databases. The objective is to train the Cluster2Former with the original and the modified scribble datasets as well, probing the boundaries of its capabilities in semi-supervised learning and comparing and analyzing the results. The significance of the model lies in its ability to harness the power of semi-supervised learning, enabling effective clustering with minimal annotated data. This not only addresses the challenges posed by limited labeled datasets, but also enables to use video instance segmentation techniques with greater efficiency and broader applicability.

Contents

1	Introduction, definition of the research objectives	3
2	Related works	6
3	Literature and background review	10
3.1	A paradigm shift with DETR	10
3.2	Components of the Mask2Former	11
3.2.1	Backbone	12
3.2.2	Pixel decoder	14
3.2.3	Transformer decoder	15
3.3	Pixel clustering using pairwise constraints	17
3.3.1	Understanding the pairwise constraints for pixel clustering	17
3.3.2	Future Directions and Challenges	20
4	Description of the methods	22
4.1	Annotation based on scribbles	22
4.1.1	Obtaining the scribble datasets	23
4.2	Similarity-based constraint loss	24
4.3	Pairing strategies	26
4.4	Point-pair sparsification	28
4.5	Architecture and training	29
4.5.1	Losses and obtaining output predictions	29
5	Experiments and Results	33
5.1	Datasets	33
5.2	Implementation Details	33
5.3	Tensorboard	36
5.4	Experiments	36
5.5	Ablation Experiments	39

6 Summary and Discussions	43
6.1 Future usage and application of Cluster2Former and our scribble-based method	45
6.2 Softwer implementations	47
Acknowledgements	48
Bibliography	48
List of Figures	54
List of Tables	57

Chapter 1

Introduction, definition of the research objectives

Video instance segmentation (VIS) involves identifying and delineating individual objects in a video sequence, while distinguishing between different instances of the same object category. This task combines three main components: *object detection*: identifying and localizing objects within each video frame; *object tracking*: keeping track of the same objects as they move across frames, maintaining their identities; and *instance segmentation*: providing a pixel-level mask for each object instance.

Video instance segmentation plays a key role across diverse applications, such as autonomous driving, surveillance, robotics, and video analysis. By identifying and tracking individual instances of objects, this technique is instrumental in enabling systems to comprehend the details of object movements and interactions within a temporal context in a given video stream. Its applications extend beyond visual comprehension, providing insights into the spatial and temporal relationships among objects. In autonomous driving scenarios, for instance, video instance segmentation is indispensable for real-time decision-making, ensuring the safe navigation of vehicles through complex environments. In surveillance and robotics, this technology facilitates the precise monitoring and manipulation of objects, improving the efficiency and accuracy of these systems. Furthermore, in video analysis, the capability to discern and isolate specific instances greatly contributes to extracting meaningful information and patterns from large datasets. As technological advancements continue, the importance of video instance segmentation becomes increasingly apparent, changing the way we perceive and interact with visual data across various domains.

Cutting-edge VIS models currently rely on comprehensive video annotations from VIS datasets [1, 2] to face this demanding challenge. However, video annotation is expensive, especially when creating precise object masks. In addition, there are instances where it becomes difficult to precisely delineate the boundaries of objects due to occlusion [3] or blurring. Even the less precise polygon-based mask annotations are significantly more time-consuming than annotating videos with bounding boxes [4, 5] or scribbles. This issue is particularly critical for recent transformer-based VIS models [6, 7, 8], as they have a high demand for training data.

Our main objective in this research project is to develop a competitive video instance segmentation model - Cluster2Former - using lightweight annotations. We found that *scribbles* — free-hand curvy lines drawn across the characteristic parts of the instance — are not only an effective way to provide partial mask information, but are also sufficient to achieve competitive performance on VIS benchmarks. With the having of this information our research project also involves obtaining scribble-like lightweight annotation version of benchmark datasets[1, 3].

We combine the Mask2Former[9, 7] model (to achieve high VIS performance) with Similarity-based Constraint loss (to enable the semi-supervised nature of clustering-based methods). For an illustration of clustering-based methods, see Fig. 1.1.

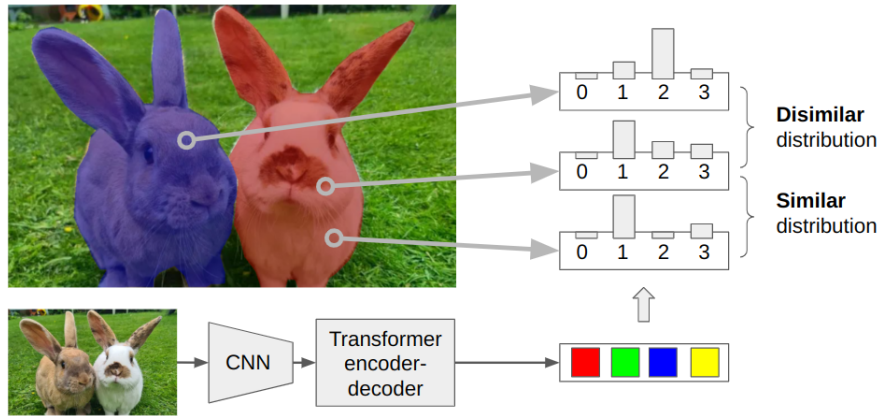


Figure 1.1: Illustration of the clustering-based approach. Based on the learned object queries we compute the output of the decoder for a pixel, representative of the features of the object the pixel is embedded in, as a probability distribution. Our learning objective guides the decoder to output similar distributions for pixels embedded in the same object, and different distributions for a pixel pair embedded in different objects.

To our knowledge, our Cluster2Former model is the first VIS method trained in partial masking that achieves high-quality segmentation results (for a survey of recent VIS models see [10]). The main contributions of this research project are summarized as follows:

- We propose a VIS model, which is trained by scribbles drawn on the training video frames. Our model achieves competitive performance, despite using only 0.54% (1.74% if we include the background) of the pixel count of the original full training masks as annotation.
- The above result is achieved by modifying the learning objective only, leaving the architecture of the transformer (in this work Mask2Former) intact. This not only eliminated costly architecture-specific hyperparameter optimization, but also enables to apply the same loss function modification on future, more advanced VIS architectures.
- We demonstrate that the pair-wise approach for training, based on feature vectors obtained by transformers, provide an efficient solution to video instance segmentation.

The importance and timeliness of our research project is very relevant, because the most commonly used full mask annotation made by humans for VIS is expensive and time consuming. Also the current state-of-the-art VIS methods are supervised methods, which means that they need big amount of annotated training data to perform well. With our method, these costs can be reduced. Consequently, we can conclude that we could annotate more data with scribbles within the same time-frame compared to full mask annotations. As a result, we can train the model with more training data.

Chapter 2

Related works

With the development of the DEformable TRansformer (DETR) model [11], a paradigm shift took place in the field of image instance detector development. Beforehand a technical change implemented to improve performance typically complicated the algorithm to such a degree, that further incremental enhancements became increasingly difficult. Starting with DETR it has been important to maintain the simple and transparent end-to-end scheme, replacing proxy based approaches. This has enabled developments such as the Mask2Former model [9] for still images, which, due to its simplicity, reaches a state-of-the-art level in solving several different tasks, such as semantic segmentation, instance segmentation and panoptic segmentation. With a minor change, Mask2Former became capable of handling video instance segmentation, beating the competition at the time of its introduction [7].

Clustering-based approaches like [12, 13] have already shown that they can achieve state-of-the-art performance in proposal-free instance segmentation. They utilized one important property of instance labeling: the pairwise relationship between pixels as the supervision to formulate the learning objective. That leads to a semi-supervised clustering problem, which we also employ. This approach beats the conventional two-stage method (feature embedding with k-means) by a significant margin [12].

In a related work pairwise constraints and subset allocation have been employed to redefine similarity-based constraints and accurately utilize strong-supervised information [14]. The efficacy of pixel-wise clustering techniques has been demonstrated in challenging video instance segmentation tasks, such as tracking identical objects, utilizing pairwise clustering methods [15].

Clusterint Mask Transformer (CMT-DeepLab) [16] has a targeted segmentation as a

pixel clustering problem. It considers the object queries as cluster centers, and is responsible for grouping pixels for segmentation. That way it improved the performance of the existing approaches, and achieved a state-of-the-art result on Common Object in Context (COCO) test-dev. Although it achieved a remarkable result, the price of this was the complexity of the architecture, which limits its applicability in various research areas.

Clustering, in particular incorporating clustering features into the learning process has been shown to improve image segmentation with class imbalanced datasets [17]. The precision of the segmentation at critical edges can be enhanced by using a supervised edge attention module [18].

We aim to keep the idea of the cluster-based approach and provide a much simpler solution, which can then be further developed to target successfully other areas, like video instance segmentation and multi-view problems.

Video Instance Segmentation (VIS) stands as a significant domain within computer vision, tasked with the challenging goal of simultaneously detecting, tracking, and segmenting objects in video sequences. Unlike conventional image-based tasks, VIS operates in the dynamic realm of videos, necessitating the ability to identify objects across multiple frames and provide precise pixel-level segmentation masks for each object instance. To gain a better understanding of this field, it is important to categorize the existing methods as two-stage approaches and one-stage approaches. The two-stage approaches [19, 20, 1] first tackle object detection in each frame, and then proceed to perform instance segmentation. It is akin to the well-established two-stage architecture seen in image-based tasks and includes well-known models like Faster R-CNN[21] and Mask R-CNN[22]. One-Stage methods, on the contrary, integrate object detection and instance segmentation into a single process. Although they are more computationally efficient, they may sacrifice accuracy. YOLACT [23] and BlendMask [24] are examples of one-stage approaches.

Track-then-segment approaches [25, 26] initially focus on object tracking across video frames, and subsequently apply instance segmentation. They rely on specialized tracking algorithms to establish object identities across frames before segmentation. Recent advances in deep learning have led to end-to-end deep learning models [27] that directly address video instance segmentation, often using temporal information to enhance results. Online and real-time methods [28, 29, 30, 31] are tailored for applications like autonomous vehicles and robotics; these methods are optimized for real-time or online video processing, emphasizing low-latency inference. Multi-object tracking and segmentation [32, 33, 34, 35, 36, 37] aim to track and segment multiple objects simultaneously

to address complex scenarios involving multiple interacting or overlapping objects.

Temporal consistency models [28, 38, 27, 6, 31, 16, 39] tackle object tracking and segmentation challenges by leveraging temporal relationships between frames in videos. Attention-based models, which use attention mechanisms to focus on relevant frame details at different time steps, excel in capturing object motion, occlusion, and appearance changes over time. This makes attention a crucial component for maintaining consistency in video instance segmentation. Most of these models use full pixel-level mask annotation of the objects to be segmented. In contrast, our approach only uses a fraction of this during training. CMT [16] is a transformer-based segmentation framework that transforms traditional transformer architectures for segmentation and detection to utilize object queries as cluster centers, which play a pivotal role in pixel grouping for segmentation. The clustering process involves two alternating steps: initially assigning pixels to clusters based on feature similarity and subsequently updating cluster centers and pixel features. Cluster2Former follows similar principles, but only with the application of the training objective, without changing the architecture.

Given the expense and complexity of annotating videos, semi-supervised and weakly supervised methods aim to reduce annotation requirements by using fewer annotated frames or less detailed annotations (such as bounding boxes) for training. In the early stages of VIS there were explorations of using videos for segmentation tasks that involve weak, semi-supervised, or unsupervised methods, with focus on motion or temporal consistency. However, many of these earlier methods did not specifically tackle object coherence and relied on optical flow for frame-to-frame matching. An approach to unsupervised feature learning leverages low-level motion-based grouping cues [40], resulting in an effective visual representation trained using unsupervised motion-based segmentation on videos. Ref. [41] predicts segmentation masks of multiple instances by learning instance tracking networks using labeled images and unlabeled video sequences. MinVIS [42] achieves VIS performance without specialized video architectures by training an image-based instance segmentation model and treating video frames as independent images thanks to its query-based approach for temporal consistency and memory-efficient inference online. MaskFreeVIS [5] achieves competitive VIS performance using only bounding-box annotations. The approach leverages temporal mask consistency through the Temporal KNN-patch loss (K Nearest Neighbours) without any labeled masks, significantly reducing annotation costs. That method outperforms the optical flow-based baselines, using bounding box annotation. While boundary boxes require even less informa-

tion than the scribble method we employ, it can be misleading in scenarios of significant occlusion, making uncertain which object it encompasses. Additionally, for objects with ambiguous boundaries, obtaining precise human annotation becomes challenging.

Chapter 3

Literature and background review

In this chapter, we are going to briefly review the key concepts, that enabled us to develop the Cluster2Former. Starting with an important research in the field of neural network based computer vision research, which is led to the creation of the Mask2Former. Followed by the introduction of the components of the Mask2Former, to understand how the model works and last but not least review the previous pixel clustering works, which let us to use the Mask2Former for pixel clustering with the appropriate modifications.

3.1 A paradigm shift with DETR

Object detection, the task of identifying and localizing objects within an image, has traditionally relied on complex architectures involving region proposal networks and convolutional neural networks (CNNs). However, in 2020, a groundbreaking approach emerged that transformed the landscape of object detection: DETR, or DEtection TRansformer[11].

DETR introduced a paradigm shift by leveraging transformers, originally designed for natural language processing tasks, to tackle object detection challenges. This innovation was different from the conventional anchor-based methods in eliminating the need for manual design choices and offering an end-to-end solution. The key Features of DETR:

Transformer Architecture: DETR adopts the transformer architecture, which had success in sequence modeling tasks, to process entire images in a novel manner. By capturing global context information effectively, transformers enable DETR to overcome the limitations of local feature extraction.

Set-Based Prediction: Unlike previous approaches that predicted bounding boxes individually, DETR revolutionized object detection by predicting the entire set of object bounding boxes and their corresponding class labels simultaneously. This set-based approach simplifies the detection pipeline and enhances scalability.

Learnable Object Queries: In DETR, learnable object queries play a crucial role in attending to relevant features within the image during the transformer’s self-attention mechanism. These object queries enable the model focus on relevant objects during inference by serving as adaptable templates for various object types.

Positional Encoding: To incorporate spatial information, DETR employs positional encoding. This encoding enables DETR to understand the relative positions of objects within the image, enhancing localization accuracy.

DETR’s revolutionary advancements have moved the field of object detection into a new era, offering state-of-the-art performance while simplifying the detection process. As researchers continue to build upon its foundation, novel approaches such as Mask2Former emerge, further pushing the boundaries of what is possible in computer vision.

In order to perform effective (video) instance segmentation, the Mask2Former model integrates several crucial components, each contributing to its overall performance. In this chapter we are going to review the components of the Mask2Former and the working mechanism of those in details, as the neural network model which we used in our research project. Then, we are going to make a brief introduction and review about the pixel clustering using pairwise constraints as well.

3.2 Components of the Mask2Former

The meta architecture of the Mask2Former is built upon the MaskFormer[43]. The Components are

1. *backbone*, which extracts features from an input video (or image),
2. *pixel decoder* that creates high-resolution per-pixel embeddings by progressively upsampling low-resolution features from the backbone’s output,
3. *transformer decoder* that processes object queries by using features from videos (images), given by the backbone and pixel decoder.

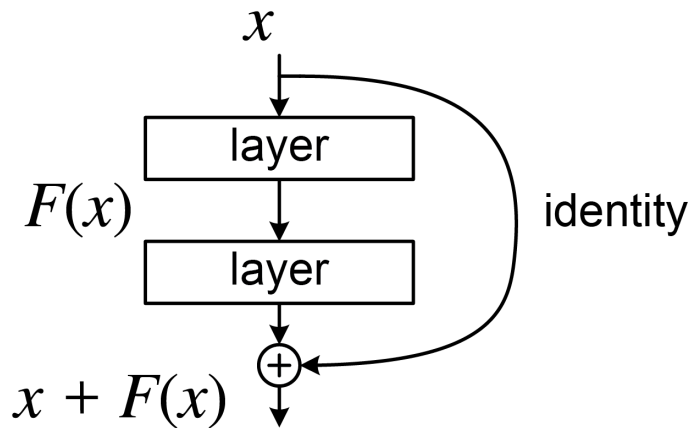
The final binary mask predictions are obtained by the embeddings using features from the pixel decoder and object queries from the transformer decoder.

The Mask2Former was originally made for image instance-, semantic- and panoptic segmentation as an universal architecture, but later[7] they realized that with a trivial extension - adding the time dimension to the input as representing videos (frames over time) from images - the Mask2Former can achieves state-of-the-art performance in video instance segmentation (they believe it works for video semantic- and panoptic segmentation as well). In the following we will explain the components of the Mask2Former for the video instance segmentation case.

3.2.1 Backbone

The Mask2Former uses convolution based neural networks, specifically ResNet-50[44] and ResNet-101[44] (Residual Network) and different types of Swin Transformers[45] as backbones. In our research we only worked with the Mask2Former, which uses ResNet-50 backbone, considering our hardware resources, because this has the least hardware resource requirements (and also this has the lowest performance). The Mask2Former has better performance with ResNet-101 and even better performance with the Swin Transformer backbones. We believe that this will work out similarly with the Cluster2Former, but for now we are going to briefly explaining the working mechanism of ResNet(-50) only:

ResNet is a deep convolutional neural network architecture that revolutionized the field of computer vision by introducing residual blocks. These residual blocks enable the training of very deep neural networks, addressing the vanishing gradient problem that arises when training deep networks.



1

Figure 3.1: A Residual Block in a deep Residual Network. Here the Residual Connection skips two layers.

How ResNet Works:

Basic Convolutional Layers: Like traditional convolutional neural networks, ResNet starts with basic convolutional layers that extract features from the input image. These layers consist of convolutional filters applied across the input image to detect low-level features like edges and textures.

Residual Blocks: The key innovation of ResNet lies in its residual blocks. Instead of directly mapping the input to the output, each residual block learns to approximate the residual mapping, i.e., the difference between the input and output features. This is achieved by introducing skip connections, also known as shortcut or skip connections, that bypass one or more layers.

In a residual block: The input is passed through two (depends on implementation) successive convolutional layers. The output of the second convolutional layer is added to the input (shortcut/skip connection). See this in Fig. 3.1. Finally, the ReLU activation function is applied to the sum, producing the output of the residual block. The skip connections help to reduce the vanishing gradient problem by allowing the gradient to flow directly through the network, facilitating the training of deeper architectures.

Downsampling: To reduce the spatial dimensions of feature maps and increase the receptive field, ResNet periodically applies downsampling operations such as stride-2 convolution or pooling layers.

Fully Connected Layers: Originally towards the end of the network, ResNet typically includes one or more fully connected layers followed by a softmax activation function for classification tasks. These layers aggregate the extracted features and produce the final output predictions, but when we use it as a backbone feature extractor we don't use these layers.

Using ResNet as a Backbone: When ResNet is used as a backbone in another neural network architecture, such as in Mask2Former, its role is primarily to extract hierarchical features from the input video (image). The ResNet backbone is pretrained on

¹source: <https://en.wikipedia.org/wiki/File:ResBlock.png>

large-scale image classification datasets like ImageNet, leveraging transfer learning to capture generic visual patterns. The process of using ResNet as a backbone involves:

- Taking an input video (image) and passing it through the ResNet backbone.
- Extracting features from different layers of the ResNet architecture to capture both low-level and high-level visual information. In the case of Mask2Former only the final feature map is used.
- These extracted features serve as the input for subsequent modules in the network, such as decoders or classification heads. In the case of Mask2Former for the pixel decoder.

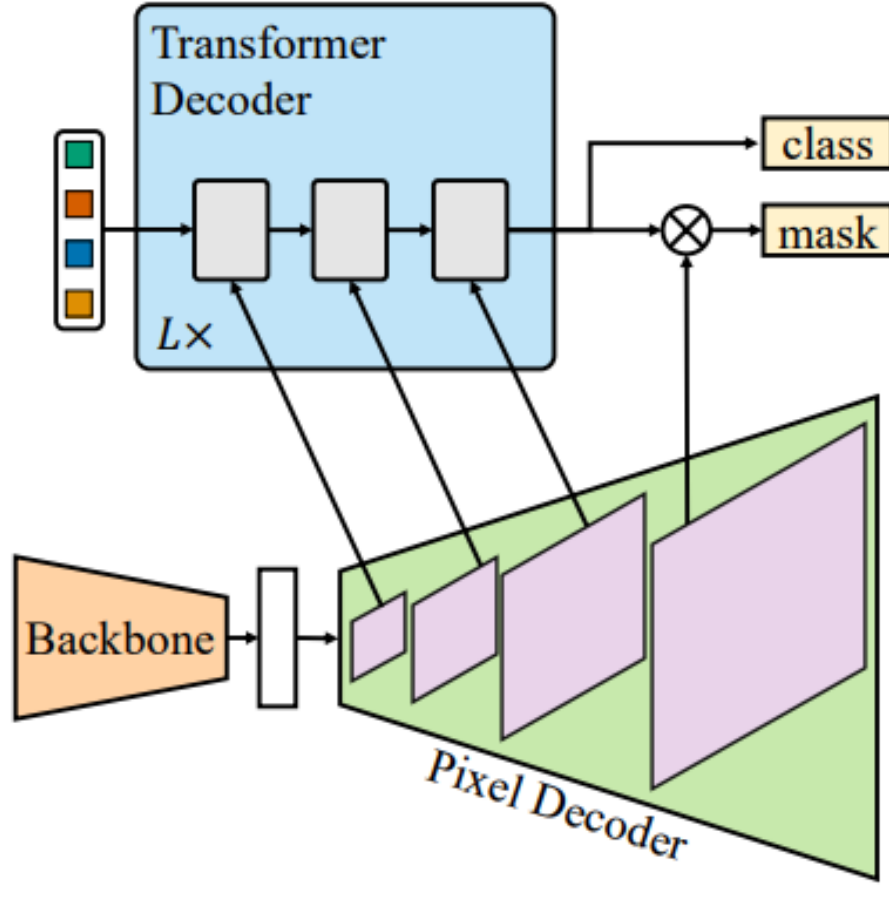
In summary, the ResNet backbone acts as a feature extractor, enabling the downstream components of the network to leverage rich and informative representations of the input video (image) for various computer vision tasks.

3.2.2 Pixel decoder

The performance of the model is enhanced by high-resolution features, particularly for small objects[11]. It takes a lot of computation to do this, though. Therefore, in order to incorporate high-resolution features while managing the rise in computation, they use an effective multi-scale approach. They feed one resolution of the multi-scale feature to one Transformer decoder layer at a time utilizing a feature pyramid made up of both low- and high-resolution features rather than constantly using the high-resolution feature map.

In particular, they use of the 1/32, 1/16, and 1/8 resolution feature pyramid that the pixel decoder generated. They add a learnable scale-level embedding $e_{lvl} \in \mathbb{R}^{1 \times C}$, following [46], and a sinusoidal positional embedding $e_{pos} \in \mathbb{R}^{H_l W_l \times C}$, following [11], for each resolution. C means the embedding dimension and H_l and W_l means the height and width (resolution) of the features. As seen in Figure 3.2, they feed the features to the Transformer decoder layers, going from lowest resolution to the higher resolutions. This 3-layer Transformer decoder is repeated $L=3$ times. This resulting, that the Transformer decoder has $3L=9$ layers altogether. To be more precise, a feature map with the resolutions $H_1 = H/32$, $H_2 = H/16$, $H_3 = H/8$, and $W_1 = W/32$, $W_2 = W/16$, and $W_3 = W/8$ is applied

²source: Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov Rohit Girdhar: Masked-attention Mask Transformer for Universal Image Segmentation, <https://arxiv.org/pdf/2112.01527.pdf>



2

Figure 3.2: Overview of the architecture of the Mask2Former. The backbone extracts features from an input image (video), then the pixel decoder makes different resolution embeddings from those features, which are fed to the successive layers of transformer decoder. For readability reasons from the figure they omitted the positional embeddings and predictions from intermediate Transformer decoder layers.

to the first three layers, where H and W represent the original resolution of the frame(s), then it is repeated in a round robin fashion.

They use the more sophisticated Multi-Scale Deformable Attention Transformer (MSDeformAttn)[46] as pixel decoder, since their aim is to show good performance on several segmentation tasks. They apply six MSDeformAttn layers to feature maps with resolutions of $1/8$, $1/16$, and $1/32$. Then, they utilize a straightforward upsampling layer with a lateral connection on the last $1/8$ feature map to create the per-pixel embedding for the feature map of resolution $1/4$.

3.2.3 Transformer decoder

The Transformer decoder component of Mask2Former introduces several innovative features, including masked attention, to enhance convergence and performance and

changing the order of self- and cross-attention. These are shown in Figure 3.3.

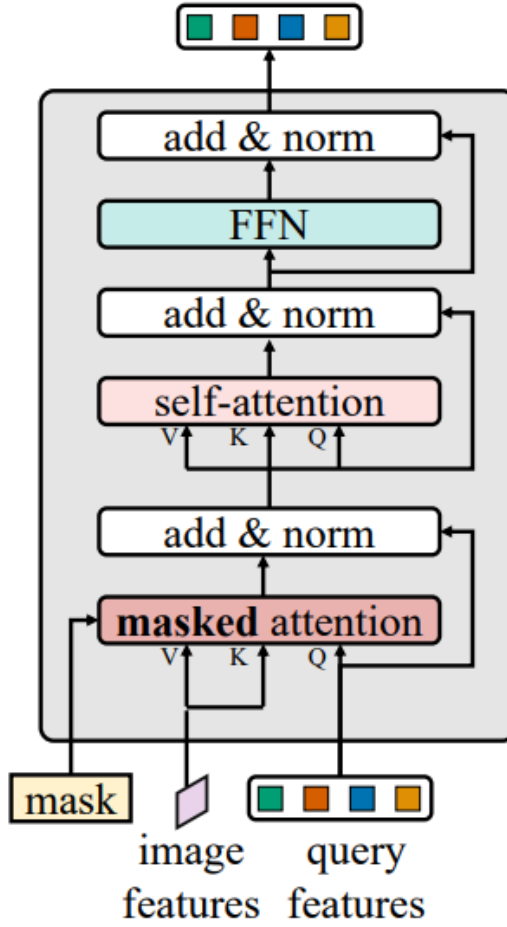
In recent research, the significance of context features in image segmentation has been widely acknowledged [47, 48, 49]. However, investigations [50, 51] into Transformer-based models have revealed a notable challenge: the slow convergence attributed to the global context in the cross-attention layer. It often requires numerous training epochs for cross-attention mechanisms to effectively focus on localized object regions [51]. To address this issue, they assumed that local features are sufficient for updating query features, and context information can be effectively captured through self-attention mechanisms. In response to this hypothesis, they introduced a novel approach (at that time) known as masked attention. This variant of cross-attention selectively attends within the foreground region of the predicted mask for each query, providing a more efficient and effective means of feature extraction for image segmentation tasks.

The Transformer decoder in Mask2Former incorporates this masked attention operator, a critical element for improving convergence and accuracy. This operator selectively extracts localized features by constraining cross-attention within the foreground region of the predicted mask for each query. Unlike standard cross-attention, which attends to the entire feature map, masked attention focuses on relevant regions, facilitating more efficient and effective feature extraction.

The final predictions (binary masks) are obtained by the product of the per-pixel embeddings and the learned object queries.

Optimization and training efficiency improvements: Mask2Former incorporates other optimization improvements to enhance model performance and training efficiency without introducing additional computational complexity. These optimizations streamline the training process and contribute to more efficient convergence. By fine-tuning the optimization strategies, Mask2Former achieves superior results while maintaining computational efficiency. We won't go into the details of these improvements, since the goal is not that, but briefly explain how Mask2Former works in order to understand our advancements. We refer readers to [7] for more details.

³source: Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov Rohit Girdhar: Masked-attention Mask Transformer for Universal Image Segmentation, <https://arxiv.org/pdf/2112.01527.pdf>



3

Figure 3.3: Overview of the architecture of the introduced Transformer decoder part of the Mask2Former. We can see this gray components in the previous Figure 3.2. This figure shows only one decoder layer for the simplicity. They changed the order of self- and cross-attention and use the proposed masked-attention instead of the the cross-attention. They made the query features learnable. The image features comes from the pixel decoder successively. The masks are the predictions from the previous layer or from the query features in case of the first layer.

3.3 Pixel clustering using pairwise constraints

3.3.1 Understanding the pairwise constraints for pixel clustering

Pixel clustering, a fundamental task in computer vision and image processing, involves grouping pixels into meaningful clusters based on their characteristics such as color, texture, or intensity. Pairwise constraints, which specify relationships between pairs of pixels (or points) (e.g., must-link or cannot-link; or positive- or negative pair), have played a significant role in enhancing the accuracy and effectiveness of pixel clustering algorithms.

Several notable advancements have been made in the field of pixel clustering using

pairwise constraints. One such approach explores the integration of neural networks with pairwise constraints for pixel clustering tasks [12]. By leveraging the power of deep learning models, this approach aims to learn meaningful representations of pixel (point) data while considering the pairwise constraints, leading to improved clustering accuracy and robustness.

Another innovative approach focuses on instance segmentation without the need for predefined object proposals[13]. This method learns to cluster pixels directly into instance segments, incorporating pairwise constraints to ensure accurate segmentation boundaries, thereby facilitating proposal-free instance segmentation in complex scenes.

In the common approach used in these advancements, softmax is applied on top of the outputs of the neural network to generate probability distributions over the pixels. See the general architecture of such an implementation in Fig. 3.4. Let $p(x_p)$ and $p(x_q)$ represent the probability distributions for pixels x_p and x_q , respectively. The Kullback-Leibler (KL) divergence between these distributions is then used as a measure of dissimilarity between pixels. In general the mathematical formulation of this problem is as follows:

$$\text{Softmax}(x_p)_i = p(x_p)_i = \frac{e^{(x_p)_i}}{\sum_j e^{(x_p)_j}}$$

$$\text{KL_loss}(p(x_p), p(x_q)) = \sum_i p(x_p)_i \log \left(\frac{p(x_p)_i}{p(x_q)_i} \right) + \sum_i p(x_q)_i \log \left(\frac{p(x_q)_i}{p(x_p)_i} \right)$$

if the pixel x_p and x_q are from the same cluster for symmetricity reasons use both the KL-divergence, where $(x_p)_i$ and $(x_q)_i$ are the outputs of the neural network for pixel p and pixel q for each cluster i and $p(x_p)_i$ and $p(x_q)_i$ represent the probabilities of pixel x_p and x_q respectively for the i -th cluster. If the the two pixel comes from different clusters, then they incorporate the Hinge-loss in the final loss (for the simplicity symmetricity doesn't shown now):

$$\text{Hinge_loss} = \max(0, \text{margin} - \text{KL}(P||Q))$$

where the *margin* is a hyper-parameter.

By minimizing the KL divergence with hinge loss between the probability distributions of pairs of pixels, the clustering algorithm encourages similar pixels (hopefully pix-

⁴source: Yen-Chang Hsu, Zsolt Kira: Neural Network-based Clustering Using Pairwise Constraints, <https://arxiv.org/pdf/1511.06321.pdf>

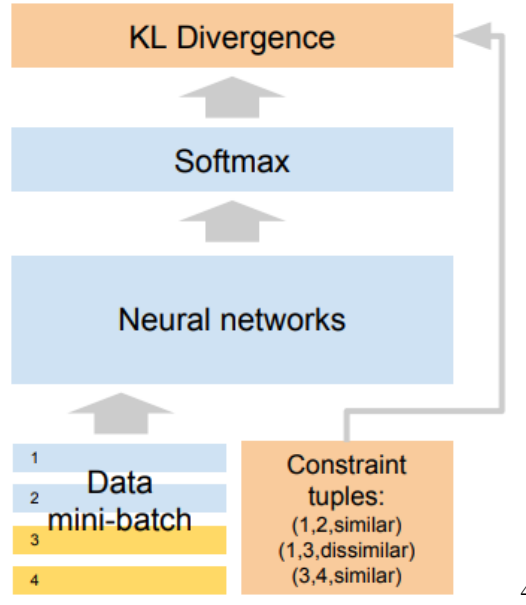


Figure 3.4: Overview of the general architecture of pixel clustering for pairwise constraints with KL-divergence.

els from one objects) to be grouped together while ensuring that dissimilar pixels (from different objects) are assigned to different clusters. This approach effectively integrates pairwise constraints into the clustering process, leading to improved clustering accuracy and robustness. See the Fig 1.1. to help to understand the similiraty of the pixels, when they are from the same or from different clusters.

Semi-supervised learning techniques have also been applied to pixel clustering, leveraging both labeled and unlabeled data [14]. By integrating pairwise constraints into deep embedded clustering algorithms, these methods enable the model to learn discriminative representations of pixel data while respecting user-provided constraints. Additionally, a subset allocation strategy further enhances clustering performance by effectively utilizing available labeled data.

Research in multi-object tracking has provided valuable insights into handling similarities between instances. A hybrid architecture for multi-object tracking in video sequences [15] emphasizes the importance of addressing similarities between instances. These observations can be extended to pixel clustering tasks, particularly in scenarios where clustering similar pixels or objects is essential for accurate segmentation.

The commonalities between these two approaches include the comparison of two pixel vectors, p and q . Let p_i and q_i denote the i -th elements of vectors p and q , respectively.

The first approach [14] utilizes cosine similarity between the pixel vectors p and q to

calculate the similarity score:

$$\text{similarity}(p, q) = \frac{\sum_i p_i \cdot q_i}{\sqrt{\sum_i p_i^2} \cdot \sqrt{\sum_i q_i^2}}$$

The second approach [15] employs the vector dot product to measure the similarity between pixel vectors p and q :

$$\text{similarity}(p, q) = \sum_i p_i \cdot q_i$$

Both approaches utilize a binary vector R (the first approach call it a), where R_i equals 1 (must-link or positive-pair), if the two pixels come from the same cluster (object) and 0 (cannot-link or negative-pair), if they come from different clusters (objects) and they use cross-entropy loss for pixel clustering after applying softmax on the similarity scores (See the mathematical formula in section 4.2).

Pixel clustering using pairwise constraints has gained significant attention due to its effectiveness in various computer vision applications, including image segmentation, object recognition, and scene understanding. Pairwise constraints provide valuable supervision information, guiding the clustering process to produce more meaningful and semantically coherent results.

The incorporation of neural networks and deep learning techniques has further enhanced the state-of-the-art in pixel clustering, enabling models to learn complex and meaningful patterns and relationships in pixel data. By leveraging pairwise constraints alongside deep learning frameworks, researchers have achieved remarkable improvements in clustering accuracy, robustness, and scalability.

3.3.2 Future Directions and Challenges

Despite the progress made, several challenges remain to be addressed in pixel clustering using pairwise constraints. These include handling large-scale datasets, improving computational efficiency, and ensuring scalability to different applications and domains. Future research directions may focus on developing more sophisticated algorithms, integrating domain-specific knowledge, and exploring novel techniques, with pairwise constraints in clustering frameworks.

Pairwise constraints provide valuable supervision information, guiding the clustering process to produce more meaningful results. These constraints specify whether pairs of

pixels should belong to the same cluster (must-link) or different clusters (cannot-link), allowing the clustering algorithm to incorporate high-level semantic information into the clustering process.

Chapter 4

Description of the methods

In this section, we present an overview of the key aspects of our proposed methodology. We explain how scribble-based annotation can effectively represent and replace the full mask ground truth; describe the loss function that can capture data from partial annotation; provide details on our strategy to select positive and negative pairs; show how we obtain the feature vectors of the pixel points to pass them to the loss functions; how we replaced the Hungarian Matcher of the Mask2Former; and finally show how all this is put together and trained. Note: From now, we won't really use the term pixel, instead we use point (to be more general), because it can be misleading. We can speak about the original image, augmented image, like ground truth, feature map, output, etc. Example an original image has pixels, but when we talk about a feature map, a "pixel" of that can means something different, can carries other information, because of different resolutions, etc.

4.1 Annotation based on scribbles

As mentioned in the Introduction, in this research we advocate the utilization of partial annotation in the form of scribbles. Scribbles are continuous curvy lines passing through representative parts of the image of the instance.

While the full mask contains complete information about the pixels of the instance, partial annotations such as a bounding box and scribbles [see Fig. 4.1(left) for illustration] have their own advantages. Scribbles have the edge, especially for more challenging situations, where the annotator can select regions of high confidence, avoiding ambiguous locations; see Fig. 4.1(right) for illustration.

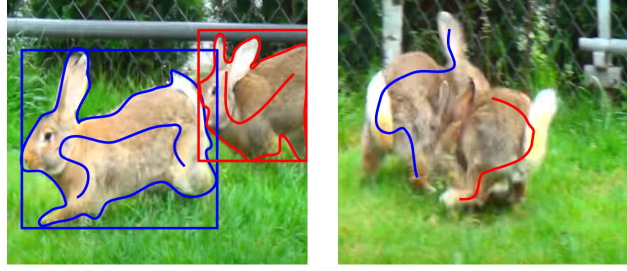


Figure 4.1: Annotations – the first instance is marked by blue, the second by red. *Left:* Three annotation methods are displayed: full mask (shown as a surrounding boundary), bounding box (minimal enclosing rectangle), and scribble. *Right:* For a more challenging example containing both occlusion and blurry regions, neither the full mask nor the bounding box can provide unambiguous annotation. Using scribbles, however, one can select representative regions with high confidence.

We return to a detailed comparison of the different annotation techniques in the Discussion section (6).

4.1.1 Obtaining the scribble datasets

We have converted the following VIS datasets into the scribble VIS dataset format:

- YouTubeVIS 2019 [1]
- YouTubeVIS 2021 [52]
- OVIS [3]

(Note: because of our hardware resources, we didn't use the Cluster2Former with the OVIS or OVIS scribble datasets in this work.)

Features of the Scribble datasets:

- The datasets are generated for the Cluster2Former, but those can be used for other projects as well.
- The scribble annotations, which are similar to skeletons (e.g. from the scikit-image python library), but not the same, are generated by the help of DAVIS-Interactive [53, 54].
- Following the COCO video format.
- Due to how the Cluster2Former works, there are background scribble annotations as well, with "category_id": None. Note: there are no annotations for the background

in the original datasets, the full background area (the not annotated area) can be obtained trivially.

- The area of the scribble annotations are 0.53% of the original annotation area. If we include the background scribble annotations, it's 1.73% of the original annotation area.

In the implementation, we converted the format of the videos into a format, which the DAVIS-Interactive can work with. We also made the background annotation as an object instance, to be able to get scribble annotation for that and, in this way, get points from that area of the background. The DAVIS-Interactive Robot converted this full mask annotations into scribble annotations, then we saved those in COCO video format (following the format of the original datasets). In the code, we use multi threaded processes for the conversion, to boost up the conversation time. Each thread is dealing with the conversion of one video. For further implementation details, see Section 6.2.

4.2 Similarity-based constraint loss

A novel learning objective was introduced in Ref. [13] to train deep neural networks to perform end-to-end image pixel clustering, with a specific focus on instance segmentation. This method takes advantage of the fundamental pairwise relationships between pixels in the context of instance labeling as a form of supervision. It utilizes the KL loss to encourage the model to generate distinct representations for pixels belonging to different objects while producing similar representations for pixels associated with similar objects. The resulting clusters serve as direct instance labeling, contributing to the field of image segmentation.

Meta-Classification Learning (MCL) is a multi-class classification technique that relies on pairwise similarity information instead of class-specific labels. The method optimizes a binary classifier for predicting pairwise similarities, ultimately learning a multi-class classifier as a submodule. MCL provides several notable advantages for various machine learning and computer vision applications. Firstly, it simplifies the model training process by eliminating the need for hyperparameter tuning, resulting in more straightforward and efficient development. Secondly, MCL excels in complex and dynamic scenarios, demonstrating robustness in handling diverse data types and variations in data distribution. Its flexibility allows it to adapt to the nuances of challenging settings. Additionally,

MCL is well suited for unsupervised and semi-supervised learning, making it valuable in situations with limited labeled data. ClusterRCNN[15] has exploited these properties to track identical individuals. By pairwise clustering of pixels on consecutive frames, we adapt it for to the requirements of VIS.

The Similarity-based Constraint loss is a key element for guiding our semi-supervised clustering process. We employ the Pairwise Constrain and Subset Allocation Deep Embedded Clustering (PCSA-DEC) approach [14], as it has shown the effectiveness of Similarity-based Constraint loss over other pair-based clustering procedures. The method relies on the comparison of point pairs, distinguishing between “positive” pairs (where the points from the pair belong to the same cluster) and “negative” pairs (where the points from the pair belong to different clusters). A cluster can be considered the region of an object or the background, and a point is represented by a feature vector p , whose components are the results of the queries with a softmax applied. The positive and negative point-point relationships are represented by a binary vector R , where the entries are set to 1 for a positive point pair and 0 for a negative pair.

For each point pair, we compute the cosine similarity S^{cos} , which ranges between -1 and 1 . The cosine similarity quantifies the degree of resemblance between the two points:

$$S_{ij}^{(\text{cos})} = \frac{p_i \cdot p_j}{|p_i| |p_j|},$$

where p_i and p_j are the feature vectors of the points i and j . The cosine similarity is transformed into a dissimilarity measure D , in the range $[0, 1]$, by

$$D_{ij} = \frac{1 - S_{ij}^{(\text{cos})}}{2},$$

which is clamped to the interval $[\varepsilon, 1 - \varepsilon]$ to ensure numerical stability for the calculation of the cross-entropy in the next step. We used $\varepsilon = 10^{-6}$ in our numerical experiments.

The core of the Similarity-based Constraint loss function involves a cross-entropy loss L , designed to minimize the deviation between in each pair of points, based on the computed dissimilarities D and the positive-negative relationships R :

$$R_{ij} = \begin{cases} 1 & \text{if } i \text{ and } j \text{ are in the same instance} \\ 0 & \text{otherwise} \end{cases}$$

This loss is computed as

$$L_{ij} = -R_{ij} \log(1 - D_{ij}) - (1 - R_{ij}) \log(D_{ij})$$

Additionally, to account for variations in the importance of point pairs based on their distances, a weighting mechanism is introduced, see section 4.3 for details. The weights W are applied to the computed loss, resulting in a weighted loss term. Finally, the Similarity-based Constraint loss $\mathcal{L}_{\text{SC}}$ is obtained as the mean across the N point pairs (ij) of the weighted loss terms:

$$\mathcal{L}_{\text{SC}} = \frac{1}{N} \sum_{(ij)} L_{ij} W_{ij}$$

The total loss for training is then

$$\mathcal{L} = \lambda_{\text{cls}} \mathcal{L}_{\text{cls}} + \lambda_{\text{SC}} \mathcal{L}_{\text{SC}},$$

where $\mathcal{L}_{\text{cls}}$ is the classification loss (as in Mask2Former), and λ_{cls} and λ_{SC} are hyperparameters.

4.3 Pairing strategies

The performance of our method on VIS benchmarks critically depends on the way pixel pairs are sampled. In the context of semi-supervised clustering, to keep resource requirements under control, we do not use all pixels within the instance mask, unlike the previous approaches [12, 13, 22].

Our approach leverages the flexibility and robustness of transformers, mitigating the limitations associated with similarity-based constraint losses used with convolutional networks [12, 13]. In particular, we do not employ binary pre-classification to separate foreground and background, and we do not impose an upper limit on the number of possible clusters. Instead, our upper limit is determined by the number of transformer queries. Therefore annotating the background with a simple scribble, similarly to the instances, is sufficient. We make a distinction between how we select positive pairs for individuals and background. While we choose positive pairs within individual instances, we do not do so for the background, see Fig. 4.2 for illustration. This choice permits the model to autonomously segment the background into multiple clusters, a practical solution given that the semantic classes in the training data do not encompass all objects in the video.

Consequently, distinct and clearly discernible entities may appear on the unannotated background, and we do not constrain the model to interpret them as a single entity, as demonstrated of Fig. 4.3.

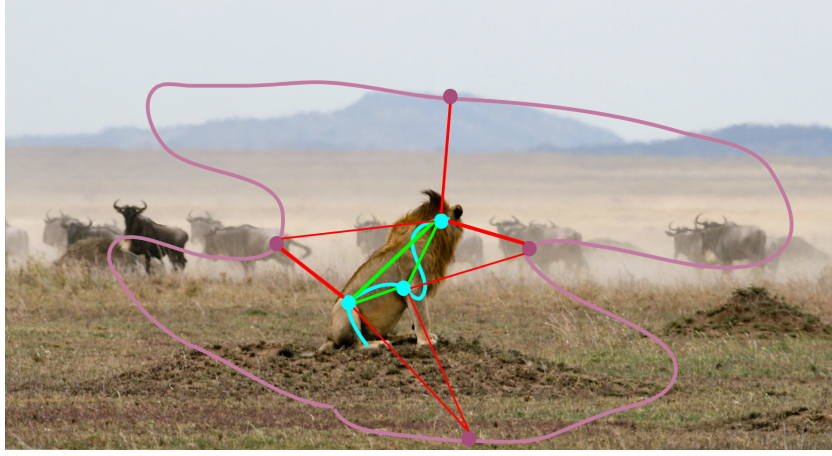


Figure 4.2: Selection of pixel pairs on foreground and background regions. The foreground object (lion) is annotated by a cyan scribble, the background by a pink scribble. Positive pairs (green lines) are selected between points of the foreground object, but not between points of the background. This way the model is not forced to recognize the background as a single homogeneous object, instead it can segment into unlabeled distinct clusters, e.g., other animals, ground, mountain and sky on this figure.

Within the same (foreground) instance, we establish positive connections between all selected pixels, the pairs being weighted by

$$W_{ij}^{(\text{positive})} = w_{\min} + (1 - w_{\min}) \frac{d_{ij}}{d_{\max}},$$

where d_{ij} is the Euclidean distance between the points i and j , $d_{\max}$ is the maximum possible distance (the diagonal of the image frame), and $w_{\min}$ is a hyperparameter. This way far away pixels of the same instance, including those separated by occlusion, are helped to be clustered together. Conversely, for negative pixel pairs from different instances, we assign weights decreasing with distance, preventing nearby, similar objects to be grouped as a single entity:

$$W_{ij}^{(\text{negative})} = 1 - (1 - w_{\min}) \frac{d_{ij}}{d_{\max}},$$

To address the variations in object sizes and the availability of annotated pixels, we adjust the sampling to select roughly equivalent portions from each object. This not only enhances the training process, but also reduces the use of computational and memory resources. Furthermore, we introduce sparsification into our approach (removing some of the low-weight negative pairs), recognizing that separating distant, dissimilar objects is



Figure 4.3: Benefits of omitting the positive pairs between background points. Top row: ground truth frames; the scribbles are shown at original resolution (1 pixel wide curves), the bounding boxes are displayed only as visual guides for the scribbles. Bottom row: prediction. The parrot at the top right corner of both frames is detected correctly, even though it was not annotated. By not enforcing similarity between different regions of the background, it can maintain its heterogeneity.

straightforward. Therefore, retaining only a fraction of the pairs is sufficient to achieve our clustering objectives. We explain it more deeply in the next section. The weighting of point-pairs and inter-frame point-pair connections can be seen in the figure 4.4.

For further implementation details, see Section 5.2.

4.4 Point-pair sparsification

Based on the idea that points that are further away from each other are not that important to form a pair (it is not necessarily true, especially when they are positive pairs) and also for memory management purposes, we came up with a point-pair sparsification method. This means that we can drop out point-pairs based on the distance of the points, that make up the pairs. We create a binary mask, with which we index the array of the point pairs using Bernoulli distribution to drop out the point pairs. In the mask, 0 (False) means we drop out the pair, and 1 (True) means we keep the pair. The mathematical formulation for obtaining an element of the binary (indexing) mask m is the following:

$$m_i = \text{Bernoulli}(\min(1, \delta + \frac{1 - \frac{d_i}{d_{\max}}}{\beta})),$$



Figure 4.4: Pairing of points on consecutive frames. The two objects are annotated by a magenta scribble (antelope) and cyan scribble (cheetah). The dots on the scribbles are sampled pixels (for clarity only a few representative sample pixels and pixel pairs are shown on this figure). The positive pairs are connected by green lines, the negative pairs by red lines. The thickness of the connecting lines represent the weight of the pair: close negative and distant positive pairs are given larger weight. The lines connecting pixels on different frames are drawn by dashed lines.

where d_i means the Euclidean distance between the points of the i -th point pair, and δ and β are hyperparameters to adjust the probability of the dropout.

4.5 Architecture and training

Leveraging the Mask2Former[7] architecture, our approach, introduces pivotal changes in the training process, while maintaining the model’s original structure. Inspired by [13], we adopt a similar pixel sampling strategy. This involves randomly sampling points from scribble-based ground truth masks for segmentation. Our method diverges in the use of similarity-based constraint loss for segmentation and cross-entropy loss for classification. The Hungarian Matcher is adapted to incorporate this new loss, ensuring Cluster2Former retains the inference efficiency of the original Mask2Former.

4.5.1 Losses and obtaining output predictions

There are notable differences between the training of the Cluster2Former and the Mask2Former:

- We don't use the mask cross entropy-loss and the dice loss at all, we only use the similarity-based constraint loss for the segmentation and the cross-entropy loss for the classification.
- We don't use the Hungarian Matcher, which they use in the Mask2Former. Instead we use our recently developed matching method. The role of the Hungarian Matcher in the Mask2Former is that it should assign the best matching output predictions from the Q (number of queries) output predictions for each ground truth object in the current video (image). For every ground truth object, it must assign only one output prediction (assume $N < Q$) and it cannot assign one output prediction to multiple ground truth objects. Now N is the number of ground truth objects and can change from video to video. In this way for each video there will be $Q - N$ output prediction, which won't be matched to any ground truth object and those will be treated as "No object". This is a bipartate matching between the output predictions and the ground truths (plus including the No object). An example is shown in Figure 4.5. The matching is based on a linear combination of the focal, dice and the classification losses between the predictions and the ground truths.

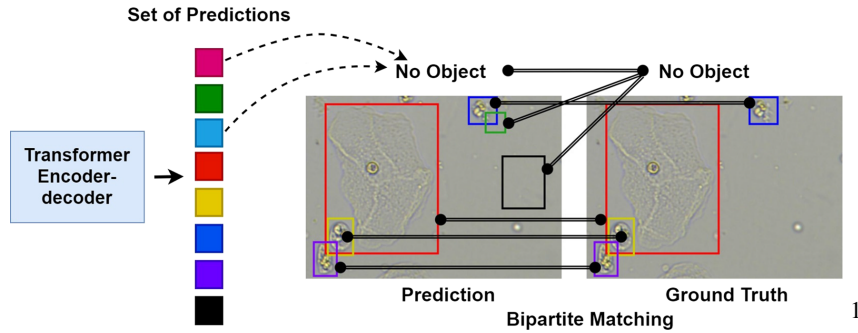


Figure 4.5: An example of how bipartite matching works between the ground truths and the outputs of a transformed based model. Each ground truth object get an output prediction assigned to it. The remaining outputs and predictions will be assigned with the "No Object" entity.

In the Similarity-based constraint loss, which is dealing with the pixel clustering, what is now the generation of the objects (pixel masks), the cosine-similarity constraining, that one query should generate one object along the video. From this, we can conclude that, after enough training iteration (with the right hyperparameters), from the Q output predictions we will clearly have N , which generates the objects. We don't care how the other

¹source: Ruqian Hao, Xiangzhou Wan, Xiaohui Du, Jing Zhang, Juanxiu Liu, Lin Liu: End-to-End Deep Learning-Based Cells Detection in Microscopic Leucorrhea Images, <https://pubmed.ncbi.nlm.nih.gov/35232520/>

queries represent the background, because as we mentioned above, we don't make positive connections between background points, so the background can be represented by multiple clusters (queries). In the early part of the training, it could happen, that multiple queries want to generate the really same object, but because of the cosine-similarity, after a while only one query should generate the object along the video. Until now we can generate the object masks, but we don't know which cluster (query) represents which object. To perform classification, we have to match the output class predictions, with the ground truth class labels. The Hungarian Matcher is an extra step and loss calculations between all the output predictions and the ground truths before the real loss calculations with the matched outputs and ground truths. We wanted to adjust this matching mechanism to our method. The Hungarian Matcher uses focal and dice losses to match predictions with the ground truth masks, but in our case, the ground truth masks are scribbles, in this way, comparing the output predictions via the previously mentioned losses with the scribble ground truths could be problematic and could lead to unexpected behaviors. The advantage of our matching method is that it works with either scribble or with full mask annotations as well. In the implementation, we named the later one MaskCluster2Former. In the following, we are going to explain how we process the output of the model to calculate the losses and how our matching method works. We do the next operations for each video in the batch:

In the case of segmentation for each frame in the video we randomly select P points from the ground truth (scribble), then we sample these points using the same point sampling method as they used in the Mask2Former (motivated by PointRend[55] and Implicit PointRend[56]) from the mask output of the model, which has a shape of $Q \times H \times W$ (Q denotes the number of queries, H denotes the height and W denotes the width size) for each frame of the video. After reshaping the sampled points, we get a $P \times Q$ shaped matrix. This is how we get the point feature vectors mentioned in section 4.2. We apply a softmax to this matrix along the dimension of the queries (Q). In that way, we get the values, the source clusters, a "distribution" for each point over the clusters (queries), which we can pass to the Similarity-based Constraint Loss with the target clusters, point pair connections (index pairs), and point pair weights. The target clusters are the IDs of the ground truth clusters (object masks) in that order, how we get them from the dataset mapper in case of the MaskCluster2Former, or how we order (sort) them according to the class labels in case of Cluster2Former (see in section 5.2 why and how we ordering them).

In the case of classification we apply an argmax to the mask output ($Q \times T \times H \times W$) along the dimension of the queries, so we get a $T \times H \times W$ shaped tensor (T denotes the

number of frames). In this way, for each frame, we get a $H \times W$ shaped matrix, where each element gives us the index of the most active (mask) output from 0 to $Q-1$ for that point. Here, the most active mask output for a pixel point means which mask output gives the highest value for that pixel point. After the upsampling of this matrix to the size of the ground truth mask for each ground truth object, we select that (label) output, where the corresponding mask output is the most active under the scribble mask annotations along all frames. In other words, select that (label) output, where the corresponding mask output generates the most number of pixel points for the given object under the scribble mask annotations along all frames. A mask output generates a pixel point if it is the most active for that point. To help the understanding briefly, we have query IDs for each point under the scribbles along the video, and we take the mode of the values, with that, we get the most active query for that object. During testing, since there is no architecture modification, the inference of Cluster2Former is the same as that of the baselines.

Chapter 5

Experiments and Results

In this section, we present details about the datasets used, the implementation, and the experimental results. We provide detailed ablation studies in addition to standard benchmark results to illustrate the effectiveness of individual parameter settings and their combinations.

5.1 Datasets

We conducted our experiments using datasets YouTube-VIS 2019 and 2021. The YouTube-VIS 2019 dataset consists of 2,883 videos with annotations for 131,000 object instances spanning 40 categories. To address more intricate scenarios, the 2021 version of YouTube-VIS introduces an additional 794 training videos and 129 validation videos, featuring tracklets with intricate motion trajectories.

We made a scribbled version of both datasets. With the DAVIS Interactive Robot [54, 53], which generates realistic scribbles that simulate human interaction, we modified the annotations for the training process. Instead of the original Ground Truth masks, which cover the whole objects, we use the scribble annotations.

5.2 Implementation Details

As mentioned in the Methods section, we adapt Mask2Former[7] (especially sampling and training loss), but keep the architecture unchanged. Unless otherwise specified, all other training schedules and settings are kept the same as in the original model.

In the case of MaskCluster2Former, when we work with the original datasets, we obtain the background trivially (not foreground object regions) when training the model from the ground truth. We have the $N \times T \times H_{gt} \times W_{gt}$ shaped ground truths, which have value 1 in the annotated area and 0 otherwise. Now N means the number of object instances, and H_{gt} and W_{gt} are the height and width of the ground truth masks. We concatenate a tensor with 0 values along the first dimension, so we get a $(N + 1) \times T \times H_{gt} \times W_{gt}$ shaped tensor. We apply an argmax on the first dimension, so we get a $T \times H_{gt} \times W_{gt}$ shaped tensor, which contains the IDs for the clusters. With this operation, the background will get ID 0, and the object IDs will come after that in that order, depending on how we get them from the dataset mapper.

On the contrary, in the case of Cluster2Former, when we work with the scribble datasets, we have scribble annotations for the background area as well, because otherwise, when we want to sample points from the background, we could mistakenly sample from the region of the objects, which are not covered by the scribble annotations. For this reason, we treat the background (annotation) as a distinct object instance. That is why we needed to make a custom dataset loader (to convert the dataset into Detectron2 format) and a custom dataset mapper (to convert the Detectron2 format into a format that the model can use) for the scribble datasets. There we give the background class label -1 to distinguish it from other objects. We give -1 because when we order (sort) the ground truths according to the class labels to get the IDs for the clusters (objects), the background is going to always be the first ground truth. In this way, when we make the classification, we always drop out the first - background - ground truth instance, because we don't have to use it there. After the sorting of the ground truths according to the class label, we will have a $(N + 1) \times T \times H_{gt} \times W_{gt}$ shaped tensor, where the background is the first element. Then we do the same thing again: concatenate a tensor with 0 values and apply argmax to the first dimension, so we get a $T \times H_{gt} \times W_{gt}$ shaped tensor with the cluster IDs, but of course now the background ID is 1 and the object ids follow it. The 0 ID means that parts of the frames have no scribble annotations (not background and not object). Of course, we don't select points from this area.

To generate the pixel-pairs for the similarity-based constraint loss, 300 pixels (the name of the hyperparameter in the implementation: *MODEL.CLUSTER_2_FORMER.TOTAL_INSTANCE_PIXEL_SAMPLE_NUM*) are sampled randomly from each frame foreground scribbles (or full masks), distributed evenly between the instances on that frame, as well as 300 pixels from the background

scribbles. The choice of frame-wise sample count enables good memory control, as opposed to instance-wise sample count as in [13], but we implemented that as well with default values $MIN_NUM_SAMPLE = 40$ and $MAX_NUM_SAMPLE = 40$ (from now if we don't specify otherwise each hyperparam is under *MODEL.CLUSTER_2_FORMER* in the configuration hierarchy), which are defining the minimum and maximum number of points sampled from each object based on their size compared to the frame size. We also use a hyperparameter called $MAX_INST = 30$ to manage memory. With that, we can set the maximum number of instances we want to work with. In cases where either the foreground or the background scribbles contained less than 300 possible pixels for sampling, e.g., due to cropping, the sample count was reduced for both to maintain balance. We make two disjoint sets of pixel-pair connections: in-frame and inter-frame connections. The in-frame connections contain pixel-pairs, where the two pixels come from the same frame, while the inter-frame connections contain pixel-pairs, where the two pixels come from different frames. When only a single frame is considered, there are no inter-frame connections set. For in-frame connections, we connect each pixel from the instances with all other pixels from the instances (they can be positive or negative pairs, depending on whether the pixels are from the same instance) and with the background pixels (these are all negative pairs). In the case of inter-frame connections for each frame pair, we connect each instance pixel from one frame with all other instance pixels from the other frame (they can be positive or negative pairs, see Fig. 4.4). Also, we connect the background pixels for each of the two frames with all instance pixels from the other frame (negative pairs). These pairings can be controlled by the $MAKE_INTER_FRAME_POINT_CONNECTIONS = \text{True}$, $MAKE_INTER_FRAME_BG_POINTS_CONNECTIONS = \text{True}$, and $MAKE_POSITIVE_BG_PAIRS = \text{False}$ (you can see the default values after the equal sign). We don't use pixel-pair sparsification; we don't drop out any connection, which can be controlled by the $BETA = 1.2$ and $DELTA = 1$ (in default 0.05) hyperparameters. We set the prefactor of the Similarity-based Constraint Loss to $\lambda_{SC} = 10$ ($CLUSTER_WEIGHT$, keeping $\lambda_{cls} = 2$) and the minimum weight $w_{min} = 0.9$ ($MIN_POINT_PAIR_WEIGHT$). In the case of inference, after applying a softmax for the mask logits along the query dimension, we produce the predictions by thresholding the given values with 0.1 ($MODEL.CLUSTER_2_FORMER.TEST.INFERENCE_THRESHOLD$). In this way, a point makes a prediction mask if its value is greater than 0.1. We call this hyperparam-

ter the inference threshold. If we do not specify otherwise, we use these hyperparameters in the experiments.

Originally, Mask2Former[7]-based models were trained with batch size 16 and a learning rate of 10^{-4} . We adapted these to our hardware resources, which were initially two Nvidia RTX TITAN GPUs, followed by Nvidia A100 GPUs. If we do not specify otherwise, we use batch size of 4 with a learning rate of 2.5×10^{-5} . We also increased the number of iterations to 24k (32k) and the steps for the learning rate decay to 16k (22k) for the YouTube-VIS 2019 (2021) val datasets to allow the model to see the same number of inputs as with the original configuration. As the architecture is unchanged, the computational requirements for prediction (inference) are identical to those of Mask2Former.

5.3 Tensorboard

We were using the tensorboard along and after the training to analyze mainly the training times, the losses and predictions. I implemented a tensorboard visualization as well, where for the first input video (now 2 frames) from a given batch we can see the ground truths, and the predictions for every `TB_VISUALIZATION.VIS_PERIOD`-th training iteration. With this we could understand more how the model learns, because we can see how the predictions changes along the training. In the beginning of the training there could be many predictions with low confidents, so we introduced the `TB_VISUALIZATION.VIS_CONF_THRESHOLD` hyperparameter to regulate how many predictions we want to visualize: visualize a prediction if it's confidence is higher than the value of the hyperparameter. See a tensorboard visualization example on Figure 5.1.

5.4 Experiments

In the following, I will show some experiment results using tables. We used the AP (Average Precision) and AR (Average Recall) as segmentation metrics. The higher value is the better. The AP50 and AP75 are meaning the AP using only IoU (Intersection over Union) 0.5 and 0.75. The AP (leftmost metric column in the tables) is averaged over Average Precision values with `IoU=.50:.05:.95`. That means from 0.5 to 0.95 (including both), with 0.05 steps. Each of these values are averaged over the categories as well. The AR1 and AR10 means the maximum recall given a fixed number (1 and 10) of detec-

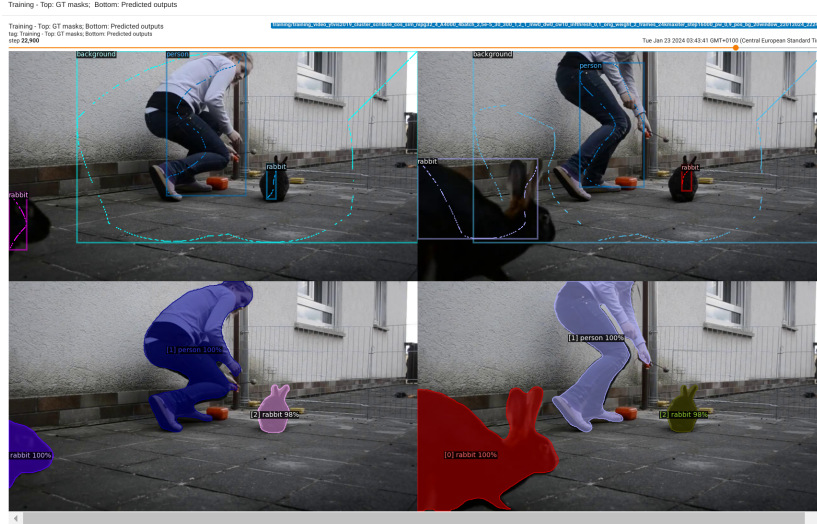


Figure 5.1: An example of the tensorboard visualization. On the top you can see the ground truths of the video (2 frames). On the bottom you can see the corresponding predictions for each frame in the 22900th training iteration.

tions per image, averaged over categories and IoUs. Our first results, shown in Table 5.1, demonstrate the power of our method. We compare the standard YouTube-VIS average precision and average recall benchmarks [1] for different model configurations on the YouTube-VIS 2019 val dataset. The integration of Similarity-based Constraint loss with Mask2Former’s segmentation losses (first row) showed no change in performance. When training with our Cluster2Former schedule (few sampled pixels only, randomly selected from the full mask; third row), a slight decay is observed, which we attribute to the information reduction of sampled pixels vs. full mask use (note that Mask2Former also samples pixels, but only 12544 of them with the default configurations). Our main point is the fourth row: when the sampling of Cluster2Former is performed from pixels on a scribble curve, only a slight reduction in performance is observed, while the training involves typically 100 times fewer pixels.

The same experiment has been performed on the more challenging YouTube-VIS 2021 val dataset as well; see Table 5.2. It is remarkable that simply adding Similarity-based Constraint loss (second row) outperforms the original Mask2Former model without altering resource requirements. We stress that if, unlike the rest of this paper, the objective is to use full annotation masks to achieve the best segmentation performance, then the best strategy is to take the linear combination of Mask2Former’s original mask losses with the similarity-based constraint loss. This dataset is more difficult than the 2019 version, which is reflected in the benchmark figures for model configurations using only the

Losses	Annotation	AP	AP50	AP75	AR1	AR10
Mask2Former[7]	mask	46.4	68.0	50.0	-	-
Mask2Former[7] + SC loss (ours)	mask	46.3	68.1	50.3	47.7	59.5
Cluster2Former (ours)	mask	41.7	66.1	45.6	42.9	51.5
Cluster2Former (ours)	scribble	38.3	62.5	42.5	39.3	46.6

Table 5.1: Comparison of different configurations of the Mask2Former and our Cluster2Former model on the YouTube-VIS 2019 validation dataset using the ResNet-50 backbone. From top to bottom: the original Mask2Former model; Mask2Former with Mask Losses and Similarity-based Constraint loss; Cluster2Former with Similarity-based Constraint loss using the original full mask annotations; and Cluster2Former with Similarity-based Constraint loss using the scribble annotations.

pairwise sampling based loss. We offer a possible explanation in Chapter 6, see Fig. 6.5.

Losses	Annotation	AP	AP50	AP75	AR1	AR10
Mask2Former[7]	mask	40.6	60.9	41.8	-	-
Mask2Former[7] + SC loss (ours)	mask	41.6	65.1	44.4	39.0	52.9
Cluster2Former (ours)	mask	34.1	55.8	37.4	33.8	42.3
Cluster2Former (ours)	scribble	29.5	51.7	30.4	30.3	37.2

Table 5.2: Comparison of different configurations of Mask2Former and Cluster2Former on YouTube-VIS 2021 using the R50 backbone. The model configurations are the same as in Table 5.1, except that we used inference threshold 0.06 in the case of the experiment with the scribble annotation.

Next, we compare our system to a few (original) state-of-the-art models for the two datasets; the results are presented in Tables 5.3 and 5.4. Similar conclusions can be drawn as above.

Method	Annotation	AP	AP50	AP75	AR1	AR10
Mask2Former[7]	mask	46.4	68.0	50.0	-	-
MaskFreeVIS[5]	bbox	43.8	70.7	46.9	41.5	52.3
SOLO-Track[41]	wo video	30.6	50.7	33.5	31.6	37.1
Cluster2Former (ours)	scribble	38.3	62.5	42.5	39.3	46.6

Table 5.3: Comparison of state-of-the-art models on the YouTube-VIS 2019 val dataset using the R50 backbone.

Method	Annotation	AP	AP50	AP75	AR1	AR10
Mask2Former[7]	mask	40.6	60.9	41.8	-	-
MaskFreeVIS[5]	bbox	37.2	61.9	40.3	35.3	46.1
Cluster2Former (ours)	scribble	29.5	51.7	30.4	30.3	37.2

Table 5.4: Comparison of state-of-the-art models on the YouTube-VIS 2021 val dataset using the ResNet-50 backbone. We used an inference threshold of 0.06

5.5 Ablation Experiments

In the context of our ablation study within the YouTube-VIS 2019 validation set and using the ResNet-50 backbone, we meticulously dissect Cluster2Former. As our baseline VIS method, we employ Mask2Former[7], fully integrating it into our approach, the only modification being the replacement of Mask losses with our custom losses. We analyse the distinct components of the pixel pair selection strategies, with a focus on the following aspects (see Table 5.5):

(1) Investigation of the interaction among background pixels. This exploration stems from the hypothesis that allowing our model the freedom to distinguish background elements into multiple clusters could potentially enhance performance.

(2) Examination of pixel relations within instances based on their spatial proximity. By emphasizing distant positive pairs and nearby negative pairs, we aim to facilitate the connection of occluded regions while effectively separating different instances in close contact.

(3) Evaluation of pixel relationships across instances in successive frames, elucidating the impact of temporal relations on tracking. This involves the classification of consecutive pixels belonging to the same instance into a shared cluster.

Neg BG only	Weighted pairs	Temp pairs	AP	AP50	AP75	AR1	AR10
-	✓	✓	35.1	59.8	37.5	36.5	43.1
✓	✓	-	33.8	58.4	36.6	36.2	42.8
✓	-	✓	36.4	61.2	39.7	39.2	45.4
✓	✓	✓	38.3	62.5	42.5	39.3	46.6

Table 5.5: Ablation results of Cluster2Former with the ResNet-50 backbone on the YouTube-VIS 2019 val dataset. Column headings: “Neg BG only”: background pixels participate only in negative pairs. “Weighted pairs”: use weights for the pixel pairs. “Temp pairs”: temporal pairs employ pairing across consecutive frames.

Furthermore, we extend our analysis to consider scenarios where more than two

frames are interconnected in this manner, offering a perspective on the temporal aspect. In this experiment, a varying number of frames (“tube length”) are selected randomly from a 20-frame video sequence; benchmark results are shown on Table 5.6. As expected, discarding temporal links (tube length 1) deteriorates performance. However, it is interesting to see that the best results are obtained for tube length 2; for larger tube lengths, the temporal connections might have been diluted.

Tube Length	AP	AP50	AP75	AR1	AR10
1	33.1	57.9	35.1	35.4	42.4
2	38.3	62.5	42.5	39.3	46.6
3	37.0	61.6	40.8	39.7	46.7
4	35.9	59.6	39.9	38.7	45.5

Table 5.6: Results of varying tube length (number of frames taking part in temporal pixel pairing) during training on YouTube-VIS 2019 val using the R50 backbone. Tube length 1 denotes the training of the model with only spatial pixel pairs.

In the Table 5.7 we experimented with the learning rates, it shows us that the original learning rate from the Mask2Former adjusted to the batch size is the best according to the experiments.

Learning rate	AP	AP50	AP75	AR1	AR10
3.5×10^{-5}	37.5	59.3	40.7	35.2	43.8
3.0×10^{-5}	37.3	58.5	39.5	34.7	43.1
2.5×10^{-5}	38.3	62.5	42.5	39.3	46.6
2.0×10^{-5}	36.9	57.0	37.9	34.1	42.2
1.5×10^{-5}	35.5	55.4	37.2	32.7	41.7

Table 5.7: Comparison of experiments with different learning rates using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.

In the table 5.8, we can see the performance of the models using different δ hyperparameters for point-pair sparsification. As we expected, the more point-pair connections we have, the better the performance will be. The best case is when we don’t drop out of point-pairs. See more details about point-pair sparsification in the section 4.4.

The table 5.9 shows us some experiments about point-pair weighting, which regulates the importance of a point-pair when calculating the Similarity-based constraint loss. See more details about point-pair weighting in the section 4.3.

We made some experiments with using or not using certain kinds of point-pair sets. In the table 5.10 we can see that if we don’t make (negative) background point con-

Delta	AP	AP50	AP75	AR1	AR10
0.05	34.3	52.4	36.6	32.7	40.9
1	38.3	62.5	42.5	39.3	46.6

Table 5.8: Comparison of experiments with different delta hyperparameters, which control point-pair sparsification, using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.

Point-pair weight	AP	AP50	AP75	AR1	AR10
1	37.6	59.9	41.5	38.3	46.0
0.9	38.3	62.5	42.5	39.3	46.6
0.8	37.8	61.4	42.1	38.7	46.2
0.7	37.5	59.8	41.5	38.3	45.9

Table 5.9: Comparison of experiments with different point-pair weights using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.

nections between frames, then the performance will drop. Also, when we allow positive background point-pairs to form. These were somehow expected behaviors, but we were also able to prove this assumption. See this in table 5.11. Find more information about point-pair connections in Section 5.2.

Inter-frame background point connections	AP	AP50	AP75	AR1	AR10
-	33.8	51.4	35.6	31.7	39.9
✓	38.3	62.5	42.5	39.3	46.6

Table 5.10: Comparison of experiments when allowing and not allowing background point-pair connections between frames using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.

Positive background connections	AP	AP50	AP75	AR1	AR10
✓	35.1	54.4	38.3	33.7	42.0
-	38.3	62.5	42.5	39.3	46.6

Table 5.11: Comparison of experiments with different delta hyperparameters, which control point-pair sparsification, using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.

Experimenting with the inference threshold was another interesting case. We luckily found out that this "tricky" hyperparameter can play a big role in achieving the right performance. It was not that trivial to find good values; we can see that in the tables 5.12 and 5.13.

Inference threshold	AP	AP50	AP75	AR1	AR10
0.2	35.3	54.9	38.9	33.7	42.1
0.15	35.9	55.1	38.5	34.3	42.8
0.1	38.3	62.5	42.5	39.3	46.6
0.05	32.5	52.8	33.6	32.3	38.7

Table 5.12: Comparison of experiments with different inference thresholds using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.

Inference threshold	AP	AP50	AP75	AR1	AR10
0.2	27.3	47.9	26.6	26.4	33.9
0.1	28.5	49.7	28.0	38.2	35.7
0.08	28.9	50.4	29.2	29.0	36.1
0.06	29.5	51.7	30.4	30.3	37.2

Table 5.13: Comparison of experiments with different inference thresholds using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2021 val.

Chapter 6

Summary and Discussions

In the previous sections, we presented our Cluster2Former model designed to tackle VIS and showed that competitive results can be obtained despite using lightweight scribble annotation.

Annotation based on scribbles have several benefits compared to both full pixel level masks and another popular lightweight masks: the bounding boxes. When compared to *full masks*, they have a number of advantages: (1) Scribbles require significantly less annotator time and training. Full-mask annotation can be extremely time-consuming and costly, while scribbles are quicker and easier for annotators. Scribbles are more forgiving to annotator errors and require less skills to provide good-quality annotations. (2) Scribbles are less prone to ambiguity: in challenging scenarios with blurred images or strong object overlap, determining precise object boundaries for full masks can be ambiguous. Scribbles, on the other hand, provide a clear indication of object locations without the need for exact boundary delineation. (3) Scribbles are computational efficient: training VIS models with scribbles typically requires less computational resources than full mask-based training, making it a more feasible option for resource limited scenarios.

It has been shown by MaskFreeVIS [5] that the performance of state-of-the-art traditional VIS algorithms can be approached by using another light annotation: *bounding boxes*. Still, scribbles offer a number of advantages over bounding boxes as well: (1) Better object separation and reduced ambiguity. Scribbles provide a more effective means of separating objects, especially when they overlap or intersect. The ambiguity of object reference for nearly coinciding bounding boxes are dissolved by applying scribbles on clearly identifiable parts of the image. See Fig. 6.1 for illustration.

(2) Enhanced object location and adaptability to object shape. Scribbles offer finer-

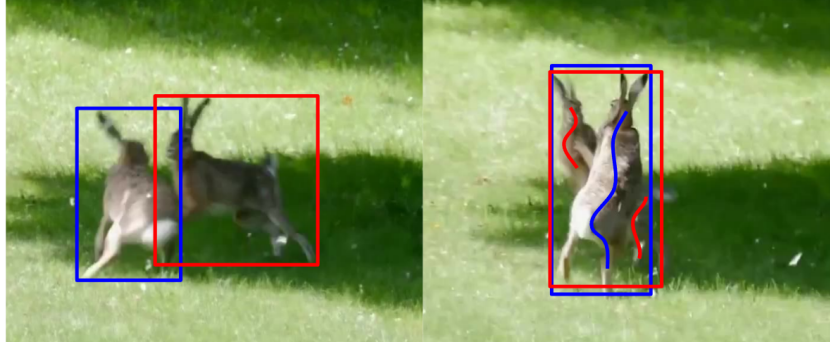


Figure 6.1: Left: Although the image is not completely sharp, the extent of the individuals can be specified with the help of a bounding box. Right: The instances are still clearly distinguishable, despite the fact that the edges are blurred, but the bounding boxes that border them almost completely coincide, since the two individuals cross each other. It is not clear which bounding box is the annotation of which Instance. On the other hand, the surfaces belonging to each individual can be clearly marked with scribbles.

grained localization information. Unlike bounding boxes, which encompass a fixed rectangular area, scribbles can guide the model to capture better the object contour, especially complex shapes. (3) Reduced annotation effort. Although both methods are partial annotations, scribbles typically require less annotator time than meticulously aligning bounding boxes to object edges, making the annotation process more efficient.

We show that Cluster2Former can successfully address challenging VIS situations. Figure 6.2 demonstrates that disjoint parts of an occluded object are combined to make a proper instance. On Fig. 6.3 we show that the edges separating neighboring instances are correctly delineated, despite the fact that scribbles provide only an approximation to the full shape.

Figure 6.4 demonstrates that Cluster2Former successfully copes with situations of multiple occlusions. The two objects (turtle and human) are correctly segmented, despite the fact that the pixels of the human are split into a number of disjoint regions.

Limitations: since the scribbles we use in the experiments (generated by DAVIS interactive [53, 54], which are similar to skeletons) sample pixels typically from the inner parts of the objects, pixels near the edges are less represented, consequently the segmentation quality near the edges might suffer. See Fig. 6.5 for illustration. One hand this is not necessarily a serious impediment, for example for applications where tracking of similar instances is the ultimate task. On the other hand there are solutions to overcome this issue, including using scribbles that approach object edges at places, or strengthening the segmentation near the edges for example using supervised edge attention [18].



Figure 6.2: Instance segmentation of an occluded object. Top row shows the annotated frames, bottom row is prediction. The bounding boxes are only guiding the eye to localize the annotation scribbles, as on Fig. 4.3.



Figure 6.3: VIS of directly adjacent instances; the separating edge is correctly found.

6.1 Future usage and application of Cluster2Former and our scribble-based method

Based on the above arguments, we believe that Cluster2Former can be one of the best overall VIS methods in a number of practical situations when training is limited by annotation resources. Even though for a fixed-size training dataset our model is slightly weaker than full-mask methods, using a fixed amount of annotator time, much larger training sets can be prepared with scribbles than full-masks, which could result in better

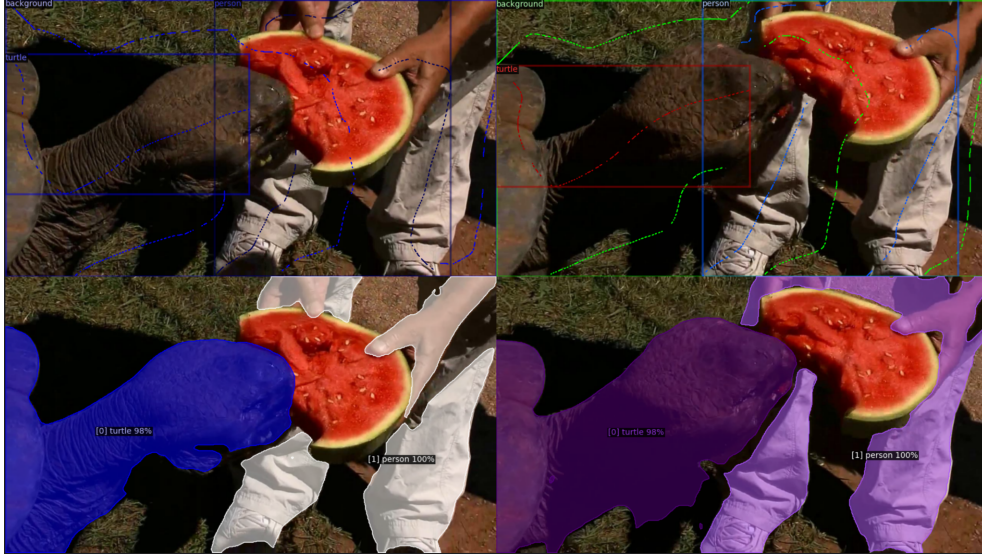


Figure 6.4: VIS of frames containing many occlusions.



Figure 6.5: Limitation of our model. Left column: frames segmented by Cluster2Former (ours) from the test set of the YouTube-VIS 2021 dataset. Even though both instances (person and snowboard) are correctly found, the detected edge of the snowboard is quite far from the ground truth. Right column: same frames segmented by Mask2Former; the edge of the snowboard is accurate.

overall performance.

Our method can be advantageous when you have a pretrained model on a bigger

dataset, e.g. COCO, which gives your model a good general understanding of a task, and you have somehow similar or even slightly different new data. With our method, it is possible to annotate them much quicker with scribbles, then fine-tune the model with the scribble-annotated data to specify it to your own new data while still getting competitive performance. An example of this is tracking similar-looking biological cells or other living beings or objects with simpler shapes, which some can annotate easily and quickly with scribbles. Thanks to simple shapes and/or the pretrained model, good segmentation results can be achieved along with cheaper annotation costs.

As an outlook, one of the applications we have in mind is tracking almost identical-looking individuals (“instances”) of animals, where maintaining the identity of individuals through frames is much more important than segmentation quality. At our department, there is a project where they track birds to analyze their behavior. When multiple birds come closer to each other and they flap their wings, it is hard to annotate the birds with full masks, because there are many unambiguous areas (which wing corresponds to which bird). Instead, they can just annotate them with scribbles and use our method; the tracking of the birds will still be doable without the big annotation effort.

Also, we are working on an extension of our model that is capable of handling a mixture of scribbles and full masks. Full or nearly full masks, where easy for the annotator, enable maintaining the full VIS benchmark of the underlying architecture, but can fall back to scribbles where full mask is costly or impossible due to blur, still enabling suitable precision for tracking.

6.2 Softwer implementations

- The implementation of the Scribble dataset converter can be found here: <https://github.com/szlAdrian/Scribble-Datasets-for-Segmentation>
- The implementation of Cluster2Former can be found here: <https://github.com/szlAdrian/Cluster2Former>

Acknowledgements

Funding:

This research was funded by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory; the “MOBOT” project (no. 2020-1.1.2-PIACI-KFI-2020-00115), implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2020-1.1.2-PIACI-KFI funding scheme; and the TKP project at ELTE (Project no. TKP2021-NVA-29), implemented with the support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme. The authors thank the Komondor computational facility operated by the Governmental Agency for IT Development (KIFÜ) in Hungary, where part of the computations have been performed, and to Robert Bosch, Ltd. Budapest, Hungary, for their generous support to the Department of Artificial Intelligence.

Acknowledgments:

We thank András Lőrincz and Ellák Somfai for many useful discussions and Xiang Linyun for her help with the early stages of this project.

Bibliography

- [1] Linjie Yang, Yuchen Fan, and Ning Xu. “Video instance segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 5188–5197.
- [2] Fisher Yu et al. “Bdd100k: A diverse driving dataset for heterogeneous multitask learning”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 2636–2645.
- [3] Jiyang Qi et al. “Occluded video instance segmentation: A benchmark”. In: *International Journal of Computer Vision* 130.8 (2022), pp. 2022–2039.
- [4] Bowen Cheng, Omkar Parkhi, and Alexander Kirillov. “Pointly-supervised instance segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2617–2626.
- [5] Lei Ke et al. “Mask-free video instance segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 22857–22866.
- [6] Junfeng Wu et al. “Seqformer: a frustratingly simple model for video instance segmentation”. In: *arXiv preprint arXiv:2112.08275* 1.2 (2021), p. 6.
- [7] Bowen Cheng et al. “Mask2former for video instance segmentation”. In: *arXiv preprint arXiv:2112.10764* (2021).
- [8] Lei Ke et al. “Video mask transfiner for high-quality video instance segmentation”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 731–747.
- [9] Bowen Cheng et al. “Masked-attention mask transformer for universal image segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 1290–1299.

- [10] Wei Shen et al. “A survey on label-efficient deep image segmentation: Bridging the gap between weak supervision and dense prediction”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2023).
- [11] Nicolas Carion et al. “End-to-end object detection with transformers”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*. Springer. 2020, pp. 213–229.
- [12] Yen-Chang Hsu and Zsolt Kira. “Neural network-based clustering using pairwise constraints”. In: *arXiv preprint arXiv:1511.06321* (2015).
- [13] Yen-Chang Hsu et al. “Learning to cluster for proposal-free instance segmentation”. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2018, pp. 1–8.
- [14] Yalin Wang et al. “Semi-supervised deep embedded clustering with pairwise constraints and subset allocation”. In: *Neural Networks* 164 (2023), pp. 310–322.
- [15] Áron Fóthi et al. “Multi Object Tracking for Similar Instances: A Hybrid Architecture”. In: *International Conference on Neural Information Processing*. Springer. 2020, pp. 436–447.
- [16] Qihang Yu et al. “Cmt-deeplab: Clustering mask transformers for panoptic segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 2560–2570.
- [17] Yaochi Zhao, Shiguang Liu, and Zhuhua Hu. “Focal learning on stranger for imbalanced image segmentation”. In: *IET Image Processing* 16.5 (2022), pp. 1305–1323.
- [18] Xier Chen et al. “Supervised edge attention network for accurate image instance segmentation”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*. Springer. 2020, pp. 617–631.
- [19] Gedas Bertasius and Lorenzo Torresani. “Classifying, segmenting, and tracking object instances in video with mask propagation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 9739–9748.
- [20] Lei Ke et al. “Prototypical cross-attention networks for multiple object tracking and segmentation”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 1192–1203.

- [21] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.
- [22] Kaiming He et al. “Mask r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2961–2969.
- [23] Daniel Bolya et al. “Yolact: Real-time instance segmentation”. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 9157–9166.
- [24] Hao Chen et al. “Blendmask: Top-down meets bottom-up for instance segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2020, pp. 8573–8581.
- [25] Qiang Wang et al. “Fast online object tracking and segmentation: A unifying approach”. In: *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*. 2019, pp. 1328–1338.
- [26] László Kopácsi et al. “RATS: Robust Automated Tracking and Segmentation of Similar Instances”. In: *International Conference on Artificial Neural Networks*. Springer. 2021, pp. 507–518.
- [27] Yuqing Wang et al. “End-to-end video instance segmentation with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 8741–8750.
- [28] Junlong Li et al. “TCOVIS: Temporally Consistent Online Video Instance Segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2023, pp. 1097–1107.
- [29] Haochen Wang et al. “Swiftnet: Real-time video object segmentation”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 1296–1305.
- [30] Shusheng Yang et al. “Crossover learning for fast online video instance segmentation”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2021, pp. 8043–8052.
- [31] Junfeng Wu et al. “In defense of online models for video instance segmentation”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 588–605.
- [32] Paul Voigtlaender et al. “Mots: Multi-object tracking and segmentation”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 7942–7951.

- [33] Ali Athar et al. “Stem-seg: Spatio-temporal embeddings for instance segmentation in videos”. In: *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI* 16. Springer. 2020, pp. 158–177.
- [34] Tim Meinhardt et al. “Trackformer: Multi-object tracking with transformers”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 8844–8854.
- [35] Jiangmiao Pang et al. “Quasi-dense similarity learning for multiple object tracking”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 164–173.
- [36] Jialian Wu et al. “Track to detect and segment: An online multi-object tracker”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2021, pp. 12352–12361.
- [37] Bin Yan et al. “Towards grand unification of object tracking”. In: *European Conference on Computer Vision*. Springer. 2022, pp. 733–751.
- [38] Sukjun Hwang et al. “Video instance segmentation using inter-frame communication transformers”. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13352–13363.
- [39] Miran Heo et al. “Vita: Video instance segmentation via object token association”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 23109–23120.
- [40] Deepak Pathak et al. “Learning features by watching objects move”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2701–2710.
- [41] Yang Fu et al. “Learning to track instances without video annotations”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2021, pp. 8680–8689.
- [42] De-An Huang, Zhiding Yu, and Anima Anandkumar. “Minvis: A minimal video instance segmentation framework without video-based training”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 31265–31277.

- [43] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. *Per-Pixel Classification is Not All You Need for Semantic Segmentation*. 2021. arXiv: 2107.06278 [cs.CV].
- [44] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].
- [45] Ze Liu et al. *Swin Transformer: Hierarchical Vision Transformer using Shifted Windows*. 2021. arXiv: 2103.14030 [cs.CV].
- [46] Xizhou Zhu et al. *Deformable DETR: Deformable Transformers for End-to-End Object Detection*. 2021. arXiv: 2010.04159 [cs.CV].
- [47] Liang-Chieh Chen et al. “DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs”. In: *CoRR* abs/1606.00915 (2016).
- [48] Liang-Chieh Chen et al. “Rethinking Atrous Convolution for Semantic Image Segmentation”. In: *CoRR* abs/1706.05587 (2017).
- [49] Hengshuang Zhao et al. “Pyramid Scene Parsing Network”. In: *CoRR* abs/1612.01105 (2016).
- [50] Peng Gao et al. “Fast Convergence of DETR with Spatially Modulated Co-Attention”. In: *CoRR* abs/2101.07448 (2021).
- [51] Zhiqing Sun et al. “Rethinking Transformer-based Set Prediction for Object Detection”. In: *CoRR* abs/2011.10881 (2020).
- [52] Linjie Yang et al. *The 3rd Large-scale Video Object Segmentation Challenge - video instance segmentation track*. June 2021.
- [53] Sergi Caelles et al. “The 2018 DAVIS Challenge on Video Object Segmentation”. In: *arXiv:1803.00557* (2018).
- [54] Jordi Pont-Tuset et al. “The 2017 DAVIS Challenge on Video Object Segmentation”. In: *arXiv:1704.00675* (2017).
- [55] Alexander Kirillov et al. “PointRend: Image Segmentation as Rendering”. In: *CoRR* abs/1912.08193 (2019).
- [56] Bowen Cheng, Omkar Parkhi, and Alexander Kirillov. “Pointly-Supervised Instance Segmentation”. In: *CoRR* abs/2104.06404 (2021). arXiv: 2104.06404. URL: <https://arxiv.org/abs/2104.06404>.

List of Figures

1.1	Illustration of the clustering-based approach. Based on the learned object queries we compute the output of the decoder for a pixel, representative of the features of the object the pixel is embedded in, as a probability distribution. Our learning objective guides the decoder to output similar distributions for pixels embedded in the same object, and different distributions for a pixel pair embedded in different objects.	4
3.1	A Residual Block in a deep Residual Network. Here the Residual Connection skips two layers.	12
3.2	Overview of the architecture of the Mask2Former. The backbone extracts features from an input image (video), then the pixel decoder makes different resolution embeddings from those features, which are feeded to the successive layers of transformer decoder. For readability reasons from the figure they omitted the positional embeddings and predictions from intermediate Transformer decoder layers.	15
3.3	Overview of the architecture of the introduced Transformer decoder part of the Mask2Former. We can see this gray components in the previous Figure 3.2. This figure shows only one decoder layer for the simplicity. They changed the order of self- and cross-attention and use the proposed masked-attention instead of the the cross-attention. They made the query features learnable. The image features comes from the pixel decoder successively. The masks are the predictions from the previous layer or from the query features in case of the first layer.	17
3.4	Overview of the general architecture of pixel clustering for pairwise constraints with KL-divergence.	19

4.1	Annotations – the first instance is marked by blue, the second by red. <i>Left:</i> Three annotation methods are displayed: full mask (shown as a surrounding boundary), bounding box (minimal enclosing rectangle), and scribble. <i>Right:</i> For a more challenging example containing both occlusion and blurry regions, neither the full mask nor the bounding box can provide unambiguous annotation. Using scribbles, however, one can select representative regions with high confidence.	23
4.2	Selection of pixel pairs on foreground and background regions. The foreground object (lion) is annotated by a cyan scribble, the background by a pink scribble. Positive pairs (green lines) are selected between points of the foreground object, but not between points of the background. This way the model is not forced to recognize the background as a single homogeneous object, instead it can segment into unlabeled distinct clusters, e.g., other animals, ground, mountain and sky on this figure.	27
4.3	Benefits of omitting the positive pairs between background points. Top row: ground truth frames; the scribbles are shown at original resolution (1 pixel wide curves), the bounding boxes are displayed only as visual guides for the scribbles. Bottom row: prediction. The parrot at the top right corner of both frames is detected correctly, even though it was not annotated. By not enforcing similarity between different regions of the background, it can maintain its heterogeneity.	28
4.4	Pairing of points on consecutive frames. The two objects are annotated by a magenta scribble (antelope) and cyan scribble (cheetah). The dots on the scribbles are sampled pixels (for clarity only a few representative sample pixels and pixel pairs are shown on this figure). The positive pairs are connected by green lines, the negative pairs by red lines. The thickness of the connecting lines represent the weight of the pair: close negative and distant positive pairs are given larger weight. The lines connecting pixels on different frames are drawn by dashed lines.	29
4.5	An example of how bipartite matching works between the ground truths and the outputs of a transformed based model. Each ground truth object get an output prediction assigned to it. The remaining outputs and predictions will be assigned with the "No Object" entity.	30

5.1	An example of the tensorboard visualization. On the top you can see the ground truths of the video (2 frames). On the bottom you can see the corresponding predictions for each frame in the 22900th training iteration.	37
6.1	Left: Although the image is not completely sharp, the extent of the individuals can be specified with the help of a bounding box. Right: The instances are still clearly distinguishable, despite the fact that the edges are blurred, but the bounding boxes that border them almost completely coincide, since the two individuals cross each other. It is not clear which bounding box is the annotation of which Instance. On the other hand, the surfaces belonging to each individual can be clearly marked with scribbles.	44
6.2	Instance segmentation of an occluded object. Top row shows the annotated frames, bottom row is prediction. The bounding boxes are only guiding the eye to localize the annotation scribbles, as on Fig. 4.3.	45
6.3	VIS of directly adjacent instances; the separating edge is correctly found.	45
6.4	VIS of frames containing many occlusions.	46
6.5	Limitation of our model. Left column: frames segmented by Cluster2Former (ours) from the test set of the YouTube-VIS 2021 dataset. Even though both instances (person and snowboard) are correctly found, the detected edge of the snowboard is quite far from the ground truth. Right column: same frames segmented by Mask2Former; the edge of the snowboard is accurate.	46

List of Tables

5.1	Comparison of different configurations of the Mask2Former and our Cluster2Former model on the YouTube-VIS 2019 validation dataset using the ResNet-50 backbone. From top to bottom: the original Mask2Former model; Mask2Former with Mask Losses and Similarity-based Constraint loss; Cluster2Former with Similarity-based Constraint loss using the original full mask annotations; and Cluster2Former with Similarity-based Constraint loss using the scribble annotations.	38
5.2	Comparison of different configurations of Mask2Former and Cluster2Former on YouTube-VIS 2021 using the R50 backbone. The model configurations are the same as in Table 5.1, except that we used inference threshold 0.06 in the case of the experiment with the scribble annotation.	38
5.3	Comparison of state-of-the-art models on the YouTube-VIS 2019 val dataset using the R50 backbone.	38
5.4	Comparison of state-of-the-art models on the YouTube-VIS 2021 val dataset using the ResNet-50 backbone. We used an inference threshold of 0.06	39
5.5	Ablation results of Cluster2Former with the ResNet-50 backbone on the YouTube-VIS 2019 val dataset. Column headings: “Neg BG only”: background pixels participate only in negative pairs. “Weighted pairs”: use weights for the pixel pairs. “Temp pairs”: temporal pairs employ pairing across consecutive frames.	39
5.6	Results of varying tube length (number of frames taking part in temporal pixel pairing) during training on YouTube-VIS 2019 val using the R50 backbone. Tube length 1 denotes the training of the model with only spatial pixel pairs.	40

5.7	Comparison of experiments with different learning rates using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.	40
5.8	Comparison of experiments with different delta hyperparameters, which control point-pair sparsification, using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.	41
5.9	Comparison of experiments with different point-pair weights using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.	41
5.10	Comparison of experiments when allowing and not allowing background point-pair connections between frames using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.	41
5.11	Comparison of experiments with different delta hyperparameters, which control point-pair sparsification, using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.	41
5.12	Comparison of experiments with different inference thresholds using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2019 val.	42
5.13	Comparison of experiments with different inference thresholds using scribble annotations and Cluster2Former with R50 backbone on YouTube-VIS 2021 val.	42