MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Davídek Hynek** Personal ID number: **435704**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Computer Science**

Study program: **Open Informatics**

Specialisation: **Artificial Intelligence**

II. Master's thesis details

Master's thesis title in English:

Label propagation for one-shot video object segmentation

Master's thesis title in Czech:

Algoritmus label propagation pro one-shot segmentaci objektů ve videu

Guidelines:

The goal of the project is reproduce results of the work of Zhang et al. CVPR 2020 and seek improvements in the propagation scheme and the feature quality.
The student is expected to:
1. Read and understand the CVPR2020 paper in detail. Background knowledge on graph-based label propgation is necessary too.
2. Reproduce the results of the paper for the proposed approach
3. Improve the proposed approach. Some of the candidate directions are to learn better representation by pairwise losses on the local embeddings and improved inference schemes that rely on label propagation.

Bibliography / sources:

Zhang, Yizhuo, Zhirong Wu, Houwen Peng, and Stephen Lin.: A Transductive Approach for Video Object Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6949-6958. 2020.
Zhou, Dengyong, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf: Learning with local and global consistency. In Advances in neural information processing systems, pp. 321-328. 2004.

Name and workplace of master's thesis supervisor:

Georgios Tolias, Ph.D., Visual Recognition Group, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **21.02.2021** Deadline for master's thesis submission: **21.05.2021**

Assignment valid until: **19.02.2023**

Georgios Tolias, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt	Student's signature

FACULTY
OF ELECTRICAL
ENGINEERING
CTU IN PRAGUE

Master's thesis

Label propagation for one-shot video object segmentation

Bc. Hynek Davídek

Department of Computer Science
Supervisor: Georgios Tolias, Ph.D.

May 20, 2021

Acknowledgements

I would like to extend my thanks to my supervisor, Georgios Tolias, Ph.D. for his insightful comments and suggestions. I would also like to thank my family and friends, for their unwavering support and belief in me.

Declaration

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis.

I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended. In accordance with Article 46 (6) of the Act, I hereby grant a nonexclusive authorization (license) to utilize this thesis, including any and all computer programs incorporated therein or attached thereto and all corresponding documentation (hereinafter collectively referred to as the "Work"), to any and all persons that wish to utilize the Work. Such persons are entitled to use the Work in any way (including for-profit purposes) that does not detract from its value. This authorization is not limited in terms of time, location and quantity. However, all persons that makes use of the above license shall be obliged to grant a license at least in the same scope as defined above with respect to each and every work that is created (wholly or in part) based on the Work, by modifying the Work, by combining the Work with another work, by including the Work in a collection of works or by adapting the Work (including translation), and at the same time make available the source code of such work at least in a way and scope that are comparable to the way and scope in which the source code of the Work is made available.

In Prague on May 20, 2021 . . . . . .

Czech Technical University in Prague
Faculty of Electrical Engineering

Citation of this thesis

Davídek, Hynek. *Label propagation for one-shot video object segmentation.* Master's thesis. Czech Technical University in Prague, Faculty of Electrical Engineering, 2021.

Abstract

This thesis focuses on improving the transductive learning approach to video object segmentation proposed by Zhang *et al.*'s paper *A Transductive Approach for Video Object Segmentation*. The paper is summarized, and necessary background knowledge of label propagation and metric learning, including deep metric learning, is introduced. An overview of current approaches to solving the video object segmentation was introduced as well. Two streams of improvements are proposed. The first one aims at improvements focusing on better inference by improving the label propagation by introducing probability propagation and by using various test-time augmentation strategies. The other improvements are focusing on better model training using triplet loss with different training triplet miners. In the last part, the thesis focuses on both quantitative and qualitative analysis of the proposed improvements. Suggested improvements increased the original paper's $\mathcal{J}$ and $\mathcal{F}$ metrics were increased by 12% and 13% for the inference-based improvements and 5% and 4% respectively using the training-based improvements.

Keywords video object segmentation, label propagation, deep metric learning, transductive learning

Abstrakt

Tato práce si klade za cíl vylepšení přístupu učení s omezenou supervizí k vyřešení problému segmentace objektů ve videu vylepšením přístupu navrženém ve článku Zhang *a kol. – A Transductive Approach for Video Object Segmentation*. Článek je shrnut a další nezbytné informace týkající se algoritmu šíření značek a učení metrik, včetně hlubokého učení metrik jsou nastíněny. Uveden je i přehled současných metod řešení problému segmentace objektů ve videu. Práce představuje dva nezávislé směry vylepšení. První směr se zaměřuje na vylepšení inference za pomoci vylepšení šíření značek šířením pravděpodobností značek a použitím různých inferenčních strategií. Druhý směr vylepšení se zaměřuje na zlepšení tréninku modelu za použití ztrátové funkce *triplet loss* a návrhem různých algoritmů pro získávání trénovacích trojic. V poslední části se práce zaměřuje na jak kvantitativní, tak i kvalitativní analýzu navržených vylepšení. Původní hodnoty metrik $\mathcal{J}$ a $\mathcal{F}$ byly zvýšeny o 12% a 13% za použití vylepšení upravujících inferenci a o 5% a 4% při použití vylepšení upravjících trénink původního modelu.

Klíčová slova segmentace objektů ve videu, šíření značek, hluboké učení metrik, transduktivní učení

Contents

List of Figures

List of Tables

Introduction

As stated in Russell and Norvig's *Artificial Intelligence: A Modern Approach*, artificial intelligence is a field of computer science that aims at building intelligent systems and understanding the principles behind them [1]. Machine learning is one of the branches of artificial intelligence that seeks to create algorithms that would learn patterns and connections from given data. It is actively used on a day-to-day basis practically everywhere around us, from image recognition in our smartphones to machine learning-powered language translation. However, one of the most popular fields of machine learning is computer vision. It is a field that focuses on analyzing either image or video data and providing insights from them on a scale that would usually be intractable for humans.

With the recent exponential increase of data volume globally, there is a tremendous need to effectively analyze and use video data. One of the tasks to be solved on such video data is video object segmentation – one of the essential tasks in the field of computer vision. Video object segmentation aims at separation of foreground pixels from the background pixels in a given video sequence. Additionally, we are given a pixel-level annotation of the first frame indicating which objects in the video are the ones of interest and the background. This task has raised a lot of attention, especially since the introduction of benchmarks like DAVIS 2016 [2] and YouTube-VOS [3]. An example of couple annotated frames from DAVIS 2017 dataset is shown in Figure 0.1. However, it is important to note that the images and annotations are not given in this form in the dataset but are given in two separate files. This task is quite challenging since there can be many occlusions, abrupt motions, camera shakes, background clutter, and other complexities.

This thesis focuses on the aforementioned video object segmentation using one-shot learning. As mentioned in its name, the one-shot learning in VOS's case aims to learn the object appearance from just one annotated sample. Annotation given from the sample is then propagated using a simple yet strong transductive method that relies on a proximity graph between local regions on

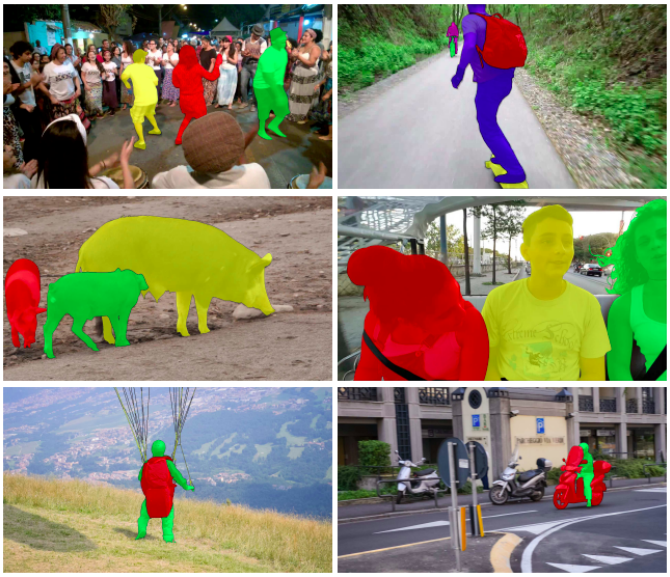

Figure 0.1: Example of couple annotated frames from the DAVIS 2017 dataset. Source: [4].

images from a video at different times. The image features for the similarities graph are obtained using a deep convolutional neural network, and regions correspond to feature map locations and their receptive field. The ultimate goal of this thesis is to improve the learning so that a better graph is established and also, given a trained convolutional neural network, to improve the propagation during inference. Since labels for only one frame in each video are provided, this task clearly corresponds to transductive learning.

Chapter 1

Goals

This thesis aims to improve the approach to video object segmentation task proposed by a paper called *A Transductive Approach for Video Object Segmentation* [5] by Zhang *et al.* presented at CVPR2020. The thesis has three main objectives:

1. Read and understand the mentioned paper in detail.
2. Reproduce results of the approach proposed by the paper.
3. Propose improvements to the paper's approach.

In order to complete the objectives, this thesis consists of several chapters dealing with the tasks at hand.

The first chapter (this one) lays down the thesis' tasks and describes the thesis' structure and its chapters.

The second chapter deals with the background knowledge needed to complete the goals, namely summarization of the CVPR2020 paper, description of label propagation, and metric learning, which are both required for later proposed improvements.

The following chapter briefly goes over other related approaches to solve the video object segmentation task.

The fourth chapter proposes improvements to the original paper's approach. There are two ways of improvements proposed, inference-based and training-based.

Finally, the last chapter measures the results of proposed improvements and puts them into perspective with the original paper's approach.

Chapter 2

Background

The Section 2.1 summarizes the paper [5] this thesis is built on. To be more precise, this thesis shows improvements and ideas where to focus on improving a novel approach to the video object segmentation problem described in [5]. The following Sections 2.2 and 2.3 lay the foundations for label propagation and metric learning, respectively, which are then further used in the next chapter describing the proposed approach to improve the original paper. Label propagation algorithm plays an important role in the predictions given by the approach from [5]. It is the background knowledge needed to understand better the propagation used in the paper. Metric learning is also important in both the paper and the thesis as the objective of the metric learning is aligned with the graph created during the label propagation. Also, one of the streams of proposed approaches exploits metric learning to learn better pixel embeddings.

2.1 A Transductive Approach for Video Object Segmentation [5]

The paper was written in 2020 by Yizhuo Zhang, Zhirong Wu, Houwen Peng, and Stephen Lin and submitted to the CVPR2020 (`http://cvpr2020.thecvf.com/`) conference. Overall it provides a simple yet high-performing and efficient method to solve the video object segmentation problem.

In the introduction part, the paper describes the problem of video object segmentation. It sets it into relation with other visual problems such as segmentation, object reidentification, optical flow estimation, or object tracking. It also claims that optical flow and tracking encourage local dependencies while instance segmentation and object reidentification enforces global dependencies. Another important thing it claims is the basic assumptions for semi-supervised learning (SSL) which are:

1. nearby samples tend to have the same label

2. samples that lie on the same manifold should should have the same label

Then it shows that the previous classical approaches to SSL include random walk, graph cut, and spectral methods. The authors then claim that they model the local dependency using a spatial prior and a motion prior and global dependency using visual appearance learned by CNN on the training data. The inference uses the regularization framework [6] using label propagation in the spatio-temporal graph. Then they claim that their approach better exploits the temporal volume compared to the current methods for the video object segmentation problem. Lastly, they claim their model does not rely on any additional modules (such as optical flow, reidentification, etc.) and works using ResNet-50 architecture [7] trained on the ImageNet dataset [8]. There are also claims of the speed of 37 FPS and achieving an overall score of 72.3% on DAVIS 2017 [4] validation set and 63.1% on DAVIS 2017 test set. Their proposed method is competitive to the current methods while being faster and simpler.

2.1.1 Approach

Third section called Approach describes the concrete methods used in the paper. The paper's approach doesn't finetune the model on a single annotated frame nor doesn't transfer knowledge but rather it exploits unlabelled structure in a video sequence. It uses the generic semi-supervised classification framework [6] modified to work for VOS problem. The framework assumes that we are given a dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), (x_l, y_l), x_{l+1}, \ldots, x_n\}$ containing l labelled data pairs and $n - l$ unlabelled data points. Task is to infer the labels $\{\hat{y}_{l+1}, \ldots, \hat{y}_n\}$ using the known data points $\{x_{l+1}, \ldots, x_n\}$. Inference is formulated using following formula,

$$\mathcal{Q}(\hat{\mathbf{y}}) = \sum_{i,j}^{n} w_{ij} \| \frac{\hat{y}_i}{\sqrt{d_i}} - \frac{\hat{y}_j}{\sqrt{d_j}} \|^2 + \mu \sum_{i=1}^{l} \|\hat{y}_i - y_i\|^2 \tag{2.1}$$

where w_{ij} encodes the similarity between datapoints x_i and x_j and d_i denotes the degree for pixel i which is computed as $\sum_j w_{ij}$. The first term of the equation denotes a smoothness constraint forcing similar points to have the same labels. The second term is a fitting constraint that penalizes the solutions that are different from the initial observations. The μ parameter helps with balancing the terms. Given the formula, the inference itself is made using the following,

$$\hat{\mathbf{y}} = \arg\min \mathcal{Q}(\mathbf{y}) \tag{2.2}$$

The above problem can be actually solved iteratively using the normalized similarity matrix $\mathbf{S}$ constructed from w_{ij} as $\mathbf{S} = \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$. The iterative solution is solved until convergence and is formulated as,

$$\hat{\mathbf{y}}(k+1) = \alpha \mathbf{S}\hat{\mathbf{y}}(k) + (1-\alpha)\mathbf{y}(0) \tag{2.3}$$

where $\alpha = \mu/(\mu+1)$ and $\mathbf{y}(0) = [y_1, \ldots, y_n]^T$. The paper then proposes a simplification of this using approximate by expanding the inference procedure through time,

$$\hat{\mathbf{y}}(t+1) = \mathbf{S}_{1:t \to t+1} \hat{\mathbf{y}}(t) \tag{2.4}$$

where $\mathbf{S}_{1:t \to t+1}$ represents the similarity matrix $\mathbf{S}$ constructed only between pixels up to the time t and the pixels in the following time. As no labels are provided apart the first frame the term $\mathbf{y}(\mathbf{0})$ is ommited for time $t+1$. Also the propagation procedure from 2.1 gets simplified to,

$$\mathcal{Q}^{t+1}(\hat{\mathbf{y}}) = \sum_i \sum_j w_{ij} \| \frac{\hat{y}_i}{\sqrt{d_i}} - \frac{\hat{y}_j}{\sqrt{d_j}} \|^2 \tag{2.5}$$

where i is the index of the pixels at target time $t+1$ and j is the index of pixels in all the previous times (including) to time t. The aforementioned approximation from Equation 2.4 is needed as the original iterative solution Equation 2.3 is unable to work online for a video stream. Also, the number of pixels in one video sequence can scale into many millions, and then the original similarity matrix would be intractable to compute.

Next part of the Approach section describes the label propagation procedure based on the Equation 2.4. It also points out that the VOS heavily depends on the similarity metric $\mathbf{S}$ and on its core component the affinity matrix $\mathbf{W}$. In the paper, the similarity matrix is defined as,

$$w_{ij} = \exp\left(f_i^T f_j\right) \exp\left(-\frac{\|\mathrm{loc}(i) - \mathrm{loc}(j)\|^2}{\sigma^2}\right) \tag{2.6}$$

with f_i, f_j being the feature embeddings for pixels p_i, p_j obtained using the CNN. $\mathrm{loc}(i)$ denotes the spatial location of pixel i. The latter term (spatial) is controlled by parameter σ. Another important part of label propagation is frame sampling. As mentioned earlier, it is intractable to compute the matrix $\mathbf{S}$ for a long video sequence. Therefore the paper proposes to sample only a small number of frames. The sampling strategy is shown in Figure 2.1. It was discovered that this sampling of a total of nine frames from the previous 40 provides a good balance between efficiency and effectiveness. The sampling approach was heavily inspired by the Temporal Segment Networks [9]. The last part of the label propagation talks about the simple motion prior model. It is represented by the second term in the Equation 2.6. It uses the knowledge of distant pixels (in the temporal domain) having weaker spatial dependencies. Therefore the paper proposes using $\sigma = 8$ for short term dependencies and $\sigma = 21$ for long term dependencies.

Another important part of the approach is learning the appearance embeddings. In the paper, it is done using a 2D CNN (more specifically ResNet-50 architecture [7]) trained on separate frames from the video, with each pixel being annotated whether or not it contains the segmented object (and its

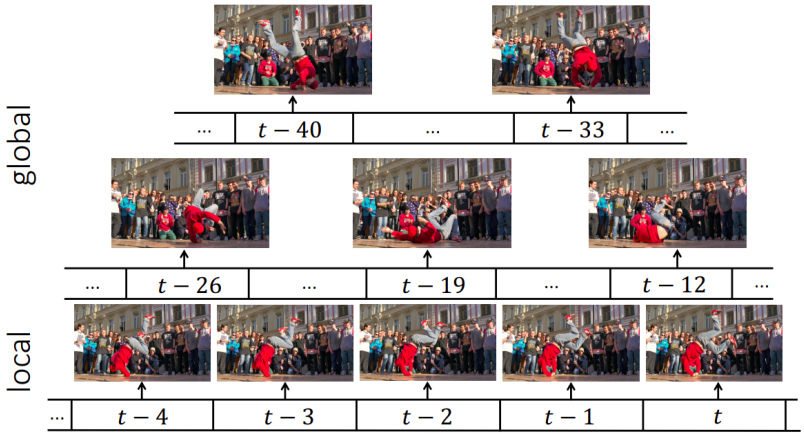

Figure 2.1: Sampling strategy for label propagation. In the recent history the samples are sampled more densely than in the distant history. Source: [5]

identity) or not. The prediction for pixel i is given by the formula,

$$\hat{y}_i = \sum_j \frac{\exp\left(f_i^T f_j\right)}{\sum_k \exp\left(f_i^T f_k\right)} y_j \tag{2.7}$$

where x_i is the target pixel, j spans over all previous pixels in prior frames. f_i and f_j represent the feature embedding from the CNN. The optimization of embedding is done using standard cross-entropy loss on all pixels in the target frame.

The next part of the Approach section goes over the implementation details of the paper's solution. As mentioned before, the paper uses ResNet-50 architecture with a minor change. That is, setting the convolution stride of the $3rd$ and $4th$ residual blocks to maintain high-resolution output. Additionally, one 1×1 convolutional layer is added to project the features to a final embedding of 256 dimensions. The final embedding also has a stride of 8. The neural net is trained from a pretrained model taken from the ImageNet model zoo and then finetuned on DAVIS 2017 [4] training set for 240 epochs and Youtube-VOS [3] for 30 epochs. Standard augmentations of random flipping (both vertical and horizontal) and random cropping of size 256×256 are applied on the input images. The model is optimized using SGD with a learning rate of 0.02 together with cosine annealing learning rate scheduling [10]. The training was done in 16 hours using 4 Tesla P100 graphic cards using a batch size of 16, each containing ten subsequences from a video sequence. During inference, the features are extracted from images of 480p resolution.

2.1.2 Results

The next section describes the results of the proposed methods and detailed ablations and discussion on temporal stability and the relationship to optical flow.

First, let's start with the experimental setup. As mentioned before, the used datasets were DAVIS 2017 and Youtube-VOS. The first one contains 150 video sequences and involves multiple objects, many occlusions, fast movements, and other challenges. The other one contains 4453 training sequences and 474 validation sequences. Both datasets contain high-resolution annotations for both train and validation sets. The difference is, though, in FPS. Each video in DAVIS 2017 is 24 FPS, while in YouTube-VOS, it is only 5 FPS.

Another important thing to mention is the evaluation metrics. The paper uses a standard evaluation metric of mean intersection over union (mIoU), averaged across objects, and summed over all frames. The mIoU gets evaluated on both full objects – then it is called $\mathcal{J}$ measure and only on object boundaries – $\mathcal{F}$ measure. Another measure used is the $\mathcal{G}$ measure which is a mean of $\mathcal{J}$ and $\mathcal{F}$ measures.

The next part of the results section talks about the ablation study. It is divided into three parts: dense local and global dependencies, transferred representations, and the simple motion prior. The first part of the ablation study, dealing with dense local and global dependencies, states that the paper focuses on building long-term models over the spatio-temporal volume. It also provides a table comparing the effects of local and global dependencies. It shows that inference over the longer-term improves the performance together with dense sampling near the target frame. Meanwhile, during training, it proved better to train on nine consecutive frames. The second part of the ablation study shows that inference using just a standard model pretrained on ImageNet without any further finetuning performed even better than some of the previous methods, which were actually trained on DAVIS data. Even an unsupervised model pretrained on images performed competitively using the transductive inference algorithm. The third part of the ablation study briefly goes over the simple motion prior. It shows that the simple motion prior model used in the paper leads to approximately 1% improvement. It is also stated that more complicated motion models could be even more effective.

The next part of the results section deals with the quantitative results compared to other recent methods and divides them into two groups. The first one which use the first frame for finetuning are: CNN-MRF [11], DyeNet [12], PReMVOS [13]. The other group contains the methods that work differently, such as FEELVOS [14], STM [15] and the paper itself. It also provides a simple table comparing which method requires which external module (re-identification, etc.). It also points out that methods such as PReMVOS, DyeNet, and CNN-MRF cannot run in an online fashion because they use

information from future frames to stabilize prediction for the current frame. Meanwhile, propagation-based methods can track objects online. Next, it provides a table comparing quantitative evaluation on DAVIS 2017 test-dev dataset showing that the paper's proposed method performs better than any other propagation-based method by about $3-4\%$ while being either the same or worse than the methods using finetuning on the first frame. The very same fact also shows the table comparing different methods on the YouTube-VOS dataset. But what is important to point out is the fact that the paper's proposed method is very fast compared to any other method. It is able to run up to 40 FPS on a single NVIDIA Titan Xp GPU, while the other propagation-based methods run about $2-3$ FPS. The best finetuning-based method (DyeNet) is able to run only about 0.5 FPS.

Last part of the results section discusses two topics. The first one is temporal stability. Even though it is usually not an evaluation criterion, the paper states it is significantly more temporal stable than other methods such as PReMVOS. It is clearly shown that the proposed method is robust against noise and therefore makes temporally consistent predictions. Another topic discussed is whether or not the model learns optical flow. Paper's method learns a soft mechanism for associating pixels in the target frame with pixels in the history frames, which is similar to optical flow. The paper provides a simple formula to calculate optical flow from two consecutive frames $\Delta d_i = \sum_j s_{ij} \Delta d_{ij}$ where s_{ij} is the normalized similarity and Δd_{ij} is displacement between i, j. It then compares the calculated flow to flow computed by the FlowNet [16] model. The flow net computed by the paper's model is much worse than the one computed by the FlowNet, even with adding a smoothness constraint.

2.2 Label propagation

Label propagation algorithm was first introduced in 2002 by Zhu and Ghahramani in [17]. It is a graph-based semi-supervised method that works iteratively to propagate labels and create communities based on this propagation. It uses two main assumptions:

1. **Smoothness Assumption** – if points are close to each other, their labels should likely be the same

2. **Cluster Assumption** – if points are in the same cluster, their labels should likely be the same

Thus intuition behind the algorithm is that a single label can quickly become dominant in a densely connected group of nodes but will have trouble crossing a sparsely connected region. Labels will get trapped inside a densely connected group of nodes, and those nodes that end up with the same label

when the algorithms finish can be considered part of the same community. Unfortunately, this suggests the algorithm will not work well when dealing with high-dimensional data or if the manifolds on which the data lies are highly curved. Though compared with other methods, the main advantages of the label propagation are relatively low running time and no prior information about the graph. The disadvantage is that the solution produced by the algorithm is not unique. As mentioned before the algorithm works on a graph representation $\mathsf{g} = (V, E)$ where nodes $V = \{1, \ldots, n\}$ represent the data points from the dataset and E represents set of weighed, undirected edges between the vertices, where weight is given by the similarity between the vertices. These similarities are given by a weight matrix $\mathbf{W}$ where $\mathbf{W}_{ij}$ is non-zero if and only if E contains edge (i, j) (with its weight given by $\mathbf{W}_{ij}$). In general, we assume that $\mathbf{W}$ is given by a symmetric positive function W_X by $\mathbf{W}_{ij} = W_X(x_i, x_j) \geq 0$. Some of the examples of matrix $\mathbf{W}$ include:

1. $\mathbf{W}_{ij} = 1$ if and only if E contains (i, j), otherwise $\mathbf{W}_{ij} = 0$. Thus being a variant of the k-NN algorithm.

2. $\mathbf{W}_{ij} = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}}$, Gaussian kernel of width σ.

The original algorithm proposed by [17] is outlined in Algorithm 1.

Algorithm 1: Label propagation

Result: Label point x_i by the sign of $\hat{y}_i^{(\infty)}$
$\mathbf{W} \leftarrow$ affinity matrix as described before
$\mathbf{D} \leftarrow \sum_j W_{ij}$, the diagonal degree matrix
$\hat{Y}^{(0)} \leftarrow (y_1, \ldots, y_l, 0, 0, \ldots, 0)$
while *not converged to* $\hat{Y}^{(\infty)}$ **do**
  $\hat{Y}^{(t+1)} \leftarrow \mathbf{D}^{-1}\mathbf{W}\hat{Y}^{(t)}$
  $\hat{Y}_l^{(t+1)} \leftarrow Y_l$
end

The Algorithm 1 works in iterative manner and requires nodes $1, 2, \ldots, l$ to be labelled with their known label (either $+1$ or -1) and nodes $l + 1, \ldots, n$ to be labelled with 0. For simple illustration of how the algorithm work see Figure 2.2. There are many improvements to this algorithm, for example label propagation inspired by Jacobi iteration algorithm proposed in [19] which is described by Algorithm 2, which adds the regularization parameter ϵ and a parameter μ.

In 2004 Zhou *et al.* [6] introduced improved label propagation algorithm called *Label spreading* which is described in Algorithm 3. This algorithm is also used as inspiration for the transductive approach in the original paper [5] this thesis builds on.

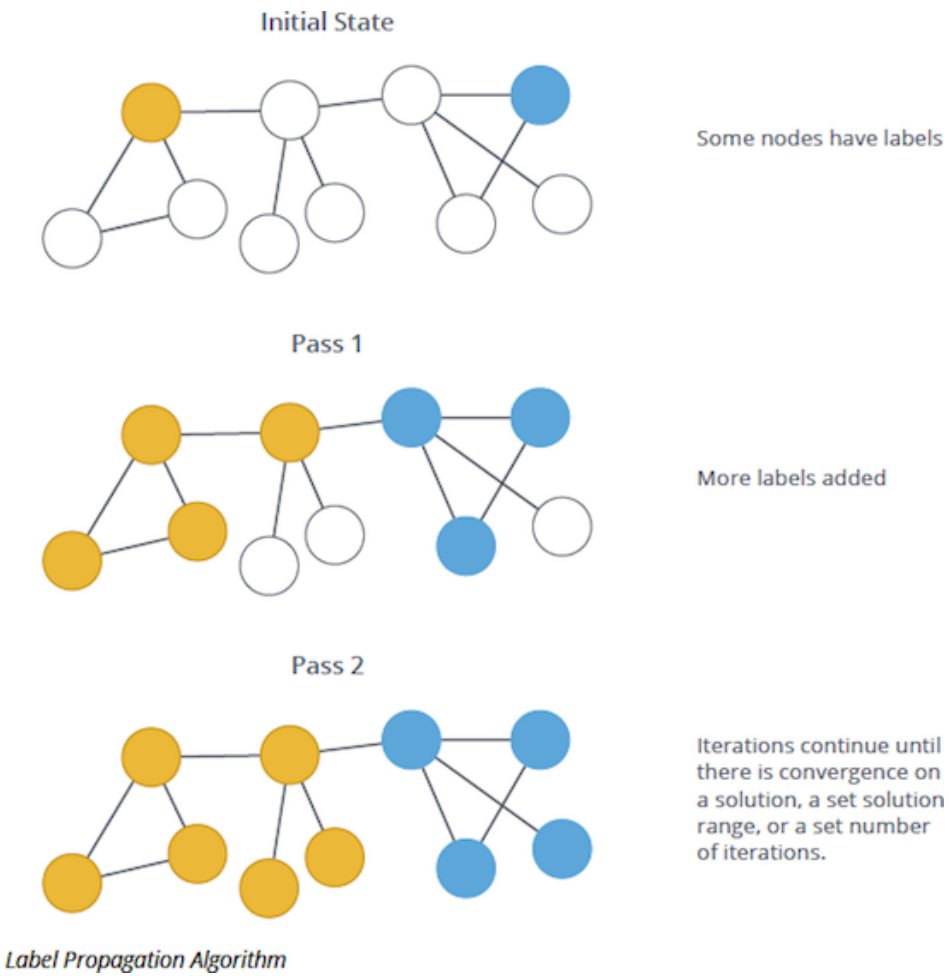

Figure 2.2: Simple illustration of work of the label propagation algorithm. Source: [18]

Algorithm 2: Label propagation (inspired from Jacobi iteration algorithm). Source: [19]

Result: Label point x_i by the sign of $\hat{y}_i^{(\infty)}$
Compute an affinity matrix $\mathbf{W}$ such that $\mathbf{W}_{ii} = 0$
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j W_{ij}$
Choose a parameter $\alpha \in (0, 1)$ and a small $\epsilon > 0$
$\mu \leftarrow \frac{\alpha}{1-\alpha} \in (0, +\infty)$
Compute the diagonal matrix $\mathbf{A}$ by $\mathbf{A}_{ii} \leftarrow I_{[l]}(i) + \mu\mathbf{D}_{ii} + \mu\epsilon$
Initialize $\hat{Y}^{(0)} \leftarrow (y_1, \ldots, y_l, 0, 0, \ldots, 0)$
Iterate $\hat{Y}^{(t+1)} \leftarrow \mathbf{A}^{-1}(\mu\mathbf{W}\hat{Y}^{(t)} + \hat{Y}^{(0)})$ until convergence to $\hat{Y}^{(\infty)}$

Algorithm 3: Label spreading (Zhou *et al.* [6]). Source: [19]

Result: Label point x_i by the sign of $\hat{y}_i^{(\infty)}$
Compute an affinity matrix $\mathbf{W}$ for $i \neq j$ and $\mathbf{W}_{ii} = 0$
Compute the diagonal degree matrix $\mathbf{D}$ by $\mathbf{D}_{ii} \leftarrow \sum_j W_{ij}$
Compute the normalized graph Laplacian $\mathcal{L} \leftarrow \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$
Initialize $\hat{Y}^{(0)} \leftarrow (y_1, \ldots, y_l, 0, 0, \ldots, 0)$
Choose a parameter $\alpha \in (0, 1)$
Iterate $\hat{Y}^{(t+1)} \leftarrow \alpha\mathcal{L}\hat{Y}^{(t)} + (1-\alpha)\hat{Y}^{(0)}$ until convergence to $\hat{Y}^{(\infty)}$

Recently, with the rise of deep learning, semi-supervised learning and label propagation had seen a surge in interest. For example, in Transductive Propagation Networks [20] the authors were motivated by a few-shot learning task, which aims to learn a classifier that is trained on a small number of training examples per class. Such a task usually has a set of known examples (called support and used for training) and a set of unknown examples (called query and used for testing). The paper uses a transductive approach by including query set in the optimization objective. It uses a convolutional neural network to transform examples into embeddings, which are then used as vertices in a graph constructed by the union of support and query sets. Then, label propagation is used to inference labels of unknown examples in the graph. The loss is computed with respect to the embeddings in the graph. Another interesting use is in a paper called Label Propagation for Deep Semi-supervised Learning [21] which also takes the transductive approach to semi-supervised tasks. At first, the neural network is trained using the original labeled samples. Then the iterative process is initialized. A nearest neighbor graph is created for the samples, and unknown labels are inferred using the label propagation algorithm. Newly created labels called pseudo-labels are then together with original labels used to retrain the network with respect to certainty-based weights.

2.3 Metric learning

Many approaches in machine learning require a measure of distance between data points. Traditionally one would use one of the „standard“ distances such as Euclidean, Manhattan, Chebyshev, etc. However, using these distance metrics requires prior domain knowledge of the problem to solve. Actually, it is challenging to design and implement metrics that are tailored to the task at hand and that work well. With high-dimensional data, it is even intractable. The field of metric learning aims to design such distances (or metrics) automatically from given data to solve specific tasks. Such tasks might include ranking, clustering, k-NN classification, information retrieval,

or even data visualization (more exactly lowering dimensions of the input data to make visualizations easier for humans to understand). This section uses one of the best sources on metric learning, which is Bellet *et al.* [22].

2.3.1 Problem settings

Historically there were two main categories of metric learning problems depending on the type of supervision (supervised and weakly supervised) given about the training data. More recently, there are also efforts done in the area of semi-supervised setting especially together with self-supervised learning in works such as Chen *et al.* [23] who presented a simple framework for contrastive learning of visual representations called *SimCLR*.

For simplicity, we assume that the training data consist of vectors that lie in some feature space called $\mathcal{X} \subseteq \mathbb{R}^d$. The training data is in form of a set of inputs which we will denote $X = \{x_1, \ldots, x_n\}$ where each of the $x_i \in \mathcal{X}$. Also, let's formally denote three sets that encode the relation between samples in set X,

$$\begin{aligned} \mathcal{S} &= \{(x_i, x_j) \in X \times X : x_i \text{ is similar to } x_j\}, \\ \mathcal{D} &= \{(x_i, x_j) \in X \times X : x_i \text{ is dissimilar to } x_j\}, \\ \mathcal{R} &= \{(x_i, x_j, x_k) \in X \times X \times X : x_i \text{ is more similar to } x_j \text{ than to } x_k\} \end{aligned} \tag{2.8}$$

The first set $\mathcal{D}$ contains pairs of instances that are known to be similar. The second set called $\mathcal{D}$ contains pairs of instances that are known to be dissimilar to each other. The third and last set $\mathcal{R}$ includes, in this case, triplets (can be, of course, generalized to any tuple) where we know that the first one is more similar to the second one than to the third one. This, as we will further see, can be useful for use in the triplet loss.

2.3.1.1 Supervised setting

The algorithm has access to a set of labelled training samplex $Z = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ where each pair consists of a training sample from the feature space $x_i \in \mathcal{X}$ and a corresponding label $y_i \in \mathcal{Y}$. $\mathcal{Y}$ is finite and discrete set of $|\mathcal{Y}|$ labels. From the set Z usually come the aforementioned sets $\mathcal{S}, \mathcal{D}, \mathcal{R}$ which create constraints for learning algorithm. An example of such setting is given in Figure 2.3.

2.3.1.2 Weakly supervised setting

In the weakly supervised setting, the algorithm has no access to the labels of individual training samples. It is only provided with constraints in the form of $\mathcal{S}, \mathcal{D}, \mathcal{R}$. This is a meaningful setting in various applications where labeled data is costly to obtain. Getting the sets, as mentioned earlier, is easy and cheap such as links between websites, user feedback, etc. This can be seen as having label information only at the pair/triplet level.

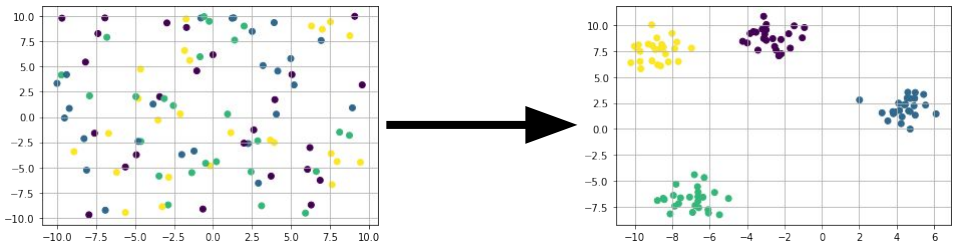

Figure 2.3: Example of supervised setting in metric learning. Labelled points in embedding space (left) are pushed together and points of different classes away from each other (right).

2.3.1.3 Semi-supervised setting

Besides the supervision (either full or weak), the algorithm also has access to a sample of unlabelled instances for which no side information is available. This is useful to avoid overfitting when the labeled data or other information is scarce. As mentioned before, one of the examples of semi-supervised setting is *SimCLR* [23] framework, which uses data augmentations on large batches to create pairs of positive and negative training data automatically.

2.3.2 Mahalanobis distance

Classical metric learning aims at learning linear metrics such as the Mahalanobis distance. Their expressive power is limited, but they are easy to optimize, as they lead to convex formulations and therefore guarantee the existence of a globally optimal solution. Compared to nonlinear metrics, they are also more robust to overfitting [22].

The Mahalanobis distance was first introduced in 1936 as a distance between some point P and a distribution D [24]. Its definition is quite simple. Given a vector $\mathbf{x} = (x_1, x_2, \ldots, x_n)^T$ from a set of observations with mean $\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n)^T$ and a covariance matrix S it is defined as:

$$D_M(\mathbf{x}) = \sqrt{(\mathbf{x} - \boldsymbol{\mu})^T S^{-1} (\mathbf{x} - \boldsymbol{\mu})} \tag{2.9}$$

However, it can also be defined as a dissimilarity measure between two vectors $\mathbf{x}$ and $\mathbf{y}$ on the same distribution again with the covariance matrix S:

$$D_M(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})^T S^{-1} (\mathbf{x} - \mathbf{y})} \tag{2.10}$$

The latter definition is usually used in the field of metric learning. Since the covariance matrix S is always positive semi-definite, we can use the decompo-

sition $M = L^T L$ to rewrite the Equation 2.10 as:

$$\begin{aligned} D_M(\mathbf{x}, \mathbf{y}) &= \sqrt{(\mathbf{x} - \mathbf{y})^T S^{-1} (\mathbf{x} - \mathbf{y})} \\ &= \sqrt{(\mathbf{x} - \mathbf{y})^T L^T L (\mathbf{x} - \mathbf{y})} \\ &= \sqrt{(L(\mathbf{x} - \mathbf{y}))^T (L(\mathbf{x} - \mathbf{y}))} \\ &= \sqrt{(L\mathbf{x} - L\mathbf{y})^T (L\mathbf{x} - L\mathbf{y})} \end{aligned} \tag{2.11}$$

In other words, Equation 2.11 shows that the Mahalanobis distance is a Euclidean distance after a linear transformation L. Actually, if we take L to be the identity matrix, we get the standard Euclidean distance.

Furthermore, since the matrix M is always positive semi-definite, the distance in the form of Equation 2.10 always holds properties required for a valid metric:

1. $d(\mathbf{x}, \mathbf{y}) \geq 0$
2. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x})$
3. $d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z})$
4. $d(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$

If the fourth condition is dropped, then $d(\mathbf{x}, \mathbf{y})$ is called a pseudometric instead of a metric [25].

2.3.3 Deep learning and metric learning

Even though the Mahalanobis distance is widely used and studied for decades, its major drawback is that it is very limited in capturing non-linear dependencies in the training data. With the rise of deep learning, neural networks have become state-of-the-art in many machine learning tasks. For example, in the field of face recognition, one of the „hot“ topics of these days, the advances would hardly be possible without the usage of convolutional neural networks. In works like Schroff *et al.* [26] the convolutional neural networks are used to extract features from the images of faces and to map the features to a Euclidean embedding space of dimension 128.

While the Mahalanobis distance's goal was to learn the transformation matrix L or M in the case of deep metric learning it is to learn network's parameters θ to provide a mapping function f which would map the inputs into a d-dimensional vector in some feature space. As being optimization of neural network's parameters θ it becomes a non-convex optimization problem of minimizing the loss $l \in \mathcal{R}$ which si given by a loss function. Unfortunately, the classic losses used with neural networks such as cross-entropy loss or mean-square loss are not as suitable for deep metric learning as others specifically

designed for metric learning purposes. Some examples are contrastive loss as proposed by Chopra *et al.* [27] or triplet loss proposed by Hoffer *et al.* [28]. In the last couple of years, there has been tremendous progress in designing such losses [29], [30] some even surpassing the traditional cross-entropy loss on the task of image classification [29].

2.3.3.1 Contrastive loss

First introduced in Chopra *et al.* [27] is defined as:

$$L(\mathbf{x}_a, \mathbf{x}_b, y) = yd(\mathbf{x}_a, \mathbf{x}_b) + (1 - y)\max(0, m - d(\mathbf{x}_a, \mathbf{x}_b)) \tag{2.12}$$

where $\mathbf{x}_a$ is the anchor embedding, $\mathbf{x}_b$ is embedding of the other training sample, $y \in \{0, 1\}$ is a label denoting whether training samples are from dissimilar ($y = 0$) or similar ($y = 1$), $m \geq 0$ is a margin parameter and $d(\cdot, \cdot)$ is the parametrized distance function. The m term is used to „tighten“ the constraint: if the two embeddings in a pair are dissimilar, then their distance should be at most m, or the loss of the given pair is ignored (in other words, there is no need to worry about negatives that are far enough from the first sample). We will call such negative samples *easy negatives* other negatives obeying the $d(\mathbf{x}_a, \mathbf{x}_b) \leq m$ condition will be called *hard negatives* (these terms will be useful in the next subsection). The intuition behind the contrastive loss is shown in Figure 2.4.

2.3.3.2 Triplet loss

As the name mentions, the triplet loss uses triplets for training instead of pairs, such triplet $(\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n)$ contains an anchor, positive and negative sample. The main idea of triplet loss is again to pull similar samples together and dissimilar apart. Still, we don't want to push the embeddings of each label to collapse into tiny clusters. The only requirement is that given two positive examples of the same class and one negative example. The negative should be farther away than the positive by some margin $m \geq 0$. The loss is defined as:

$$L(\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n) = \max(0, d(\mathbf{x}_a, \mathbf{x}_p) - d(\mathbf{x}_a, \mathbf{x}_n) + m) \tag{2.13}$$

where $\mathbf{x}_a$ is the anchor embedding, $\mathbf{x}_p$ is embedding of the positive sample, $\mathbf{x}_n$ is embedding of the negative sample, $m \geq 0$ is a margin parameter and $d(\cdot, \cdot)$ is parametrized distance function. In this case, the parameter m forces the negative sample to be farther away than the positive by some margin. The intuition behind the triplet loss is shown in Figure 2.5. In the contrastive loss case, if the distance of dissimilar samples was higher than margin m, the loss would become 0, in the case of triplet loss, there is a similar issue, but as there are three components in the calculation, it gets a little bit more tricky. There are not only easy and hard negatives but also semi-hard negatives. The intuition of the three types is shown in Figure 2.6. If we analyze the three negatives types, we get:

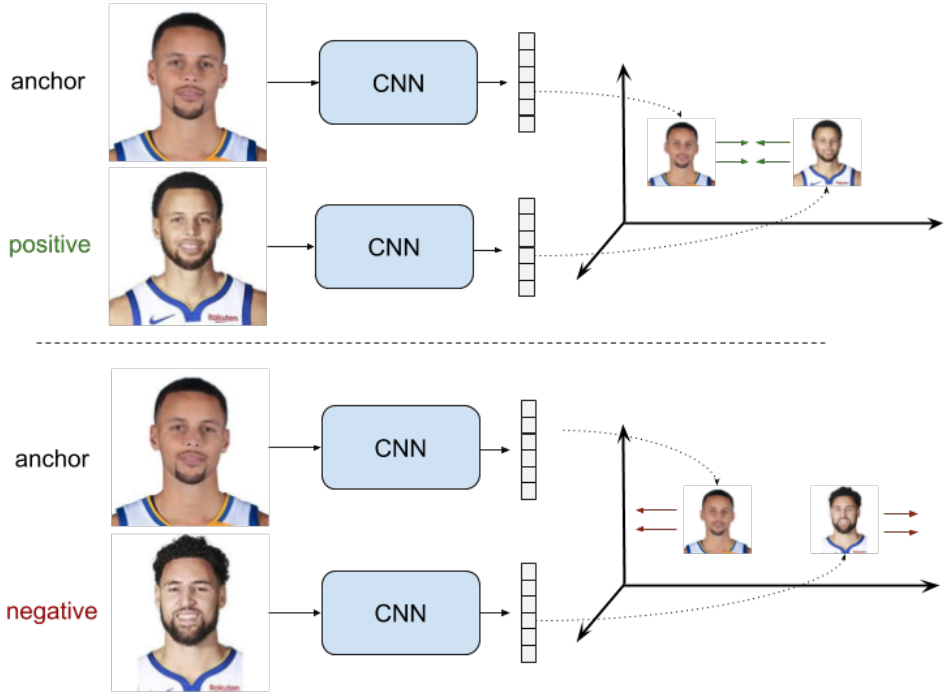

Figure 2.4: Example of contrastive loss used for image recognition. Embeddings of similar faces are pulled together, while dissimilar are pushed away from each other. Source: [31].

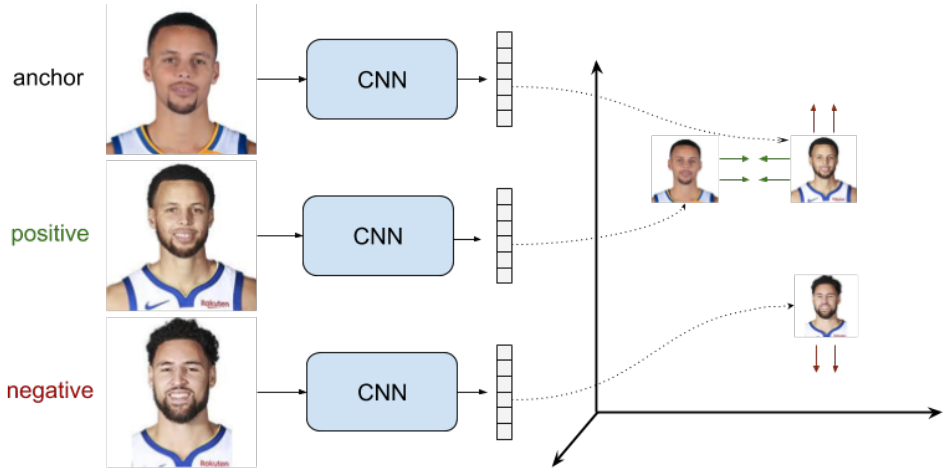

Figure 2.5: Example of triplet loss used for image recognition. Embeddings of anchor and positive sample are pulled together, while embedding of negative sample is pushed away. Source: [31].

1. easy triplets: $d(\mathbf{x}_a, \mathbf{x}_n) \geq d(\mathbf{x}_a, \mathbf{x}_p) + m$, in this case the negative is far

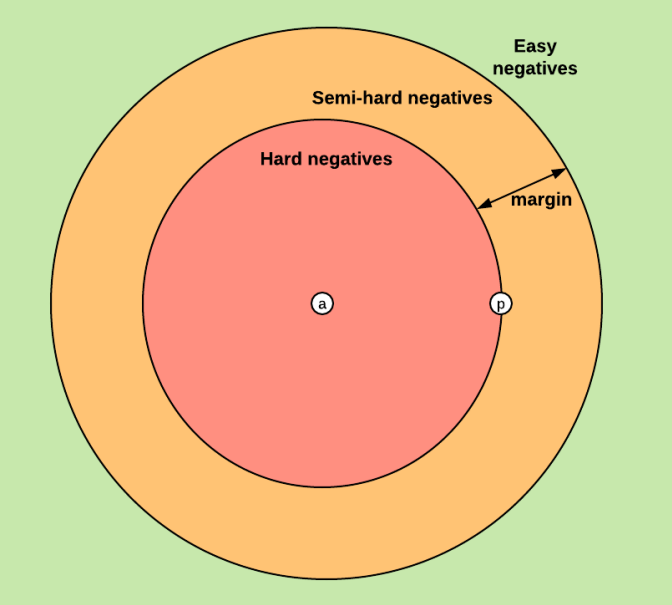

Figure 2.6: Representation of the 3 types of negatives, bubble with a represents the anchor and bubble with p is the positive sample. Source: [31].

enough so the loss becomes 0

2. hard tripets: $d(\mathbf{x}_a, \mathbf{x}_n) \leq d(\mathbf{x}_a, \mathbf{x}_p)$, here the negative sample is closer to the anchor than the positive, the loss is positive

3. semi-hard triplets: $d(\mathbf{x}_a, \mathbf{x}_p) \leq d(\mathbf{x}_a, \mathbf{x}_n) \leq d(\mathbf{x}_a, \mathbf{x}_p) + m$, the negative is farther than the positive, but closer than the margin, loss is still positive

As can be seen in the list, the selection of training triplets highly affects the performance of triplet loss. In the worst-case scenario, all sampled triplets could be only *easy*, so the loss wouldn't converge and stay constant. Also, if we are given n training samples, there are $O(n^3)$ possible triplets, so we face another problem, and that is whether we want to select all triplets prior to training (this approach is called *offline mining*) or during the training (called *online mining*). Both methods have their pros and cons, and choosing the right one is highly dependent on the given task and available hardware.

2.3.4 Applications

There are many possible applications of metric learning (both traditional and deep). This thesis will list a couple of the most used and notable [22, 32].

1. **Computer Vision** – in the field of computer vision, there is a vast usage of metric learning (deep metric learning specifically), from visual search

systems working over large databases of products [33] to face identification and verification that is present in our phones and computers [26]

2. **Information retrieval** – metric learning in the case of information retrieval can be used to create inputs, for example, for recommendation systems, search engines, or anywhere where it is crucial to be able to gather the most relevant documents to given query (document, in this case, can be anything from a piece of text to a song or a video)

3. **Nearest neighbors models** – the learned metric can be used to enhance the performance of the nearest neighbors algorithms for tasks such as classification, anomaly detection, etc.

4. **Clustering** – the learned metric can provide a way to bias the clusters found by clustering algorithms to offer more sensible results

5. **Dimensionality reduction** – the learned metric L can map inputs from higher dimension m to a lower dimension n, and this can be beneficial for tasks like visualization of high dimensional data, or generally as preprocessing the data for another machine learning algorithm

Chapter 3

Related work

This chapter goes over the notable existing methods for solving the video object segmentation problem. It is further divided into couple of sections based on the approach of the methods.

3.1 Single frame models

In the past few years, most models have been based on finetuning the model on a single input image and then inferencing on individual frames. These methods usually only learn the objectness prior and spatial continuity without even considering the temporal information. Their effectiveness shows that optimizing a domain-specific spatial smoothness term dramatically enhances performance. Unfortunately, due to their nature, it takes a long time to process a single video (about tens of seconds) which renders them unfeasible for online applications.

3.1.1 One-Shot Video Object Segmentation [34]

One of the first works in single frame models area. It leverages the use of transfer learning, where it starts with a pre-trained base CNN trained for image labeling on the ImageNet dataset. This network is then trained on binary masks obtained from the DAVIS dataset. And finally, during inference, the network is further finetuned on the first annotated frame from the test video, making it able to target a specific object in a single frame. This approach has some drawbacks, though. The first one is the architecture used – the paper uses VGG [35] architecture (it is not specified which one), which is somewhat slow compared to state-of-the-art networks used nowadays. Another drawback is the ability of the model to track only one object. This is given that the paper uses the DAVIS 2016 dataset, which contains only one annotated object per frame. The network is not modified to perform multilabel segmentation. Drawbacks aside, the model surpassed its predecessors by

whopping 11.8 points, scoring 79.8 points on the DAVIS 2016 validation set. Even though being slow in today's standard, it also performed much faster than its predecessors.

3.1.2 Online Adaptation of Convolutional Neural Networks for Video Object Segmentation [36]

Builds on top of the OSVOS approach mentioned in the preivous subsection 3.1.1. Apart from finetuning on the first frame, it performs online adaptations during the processing of the video. It means that for every other frame than the first one, the network gets finetuned for that particular frame as well as the first frame again. Despite its name, the method is significantly slower than the previous (because of the finetuning for each frame) but performs much better in terms of qualitative results. It surpassed the OSVOS by 5.9 points, scoring 85.7 points. But again, this approach uses slow VGG architecture and can track one mask only.

3.1.3 Video Object Segmentation Without Temporal Information [37]

Also called OSVOS-S by its authors is yet another model building on top of the original OSVOS. This model treats each frame independently and thus ignores the temporal information from a video. Compared to the previous model, this one is robust to object occlusions, lost frames, etc. The model has three main components: a base network acting as a feature extractor and three classifiers with shared features: the first round of foreground estimator and two conditional classifiers to model the appearance likelihood. Compared to the original OSVOS, the improved version scored 86.5 points and was better by 7.7 points.

3.1.4 Proposal-generation, Refinement and Merging for Video Object Segmentation [13]

Or short, PReMVOS is entirely different from the previously mentioned models. As already stated in the name, the model first generates coarse object proposals using the Mask R-CNN [38], then followed by a refinement network producing accurate pixel masks for each proposal. These proposals are then merged over time using optical flow warping and a Re-ID feature embedding vector. The complexity of the model can be seen in Figure 3.1. Despite the complexity, it was the best model on the DAVIS 2017 benchmark scoring 71.6 points. It achieved first place in both DAVIS 2018 VOS Challenge and YouTube-VOS 1st Large-scale Video Object Segmentation Challenge.

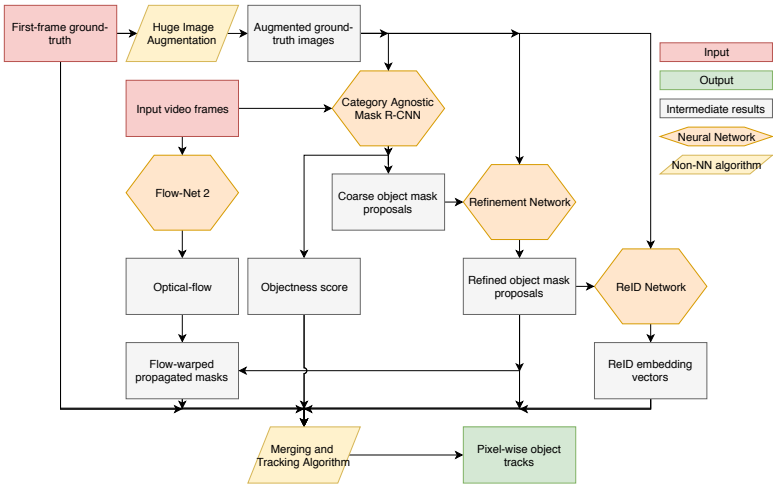

Figure 3.1: PReMVOS model diagram clearly shows the complexity of the proposed model. Source: [13]

3.1.5 CNN in MRF: Video Object Segmentation via Inference in A CNN-Based Higher-Order Spatio-Temporal MRF [11]

This model also uses an entirely different approach to tackle the VOS problem. The model exploits the spatial-temporal properties of a video by modeling the video as a Markov Random Field defined over pixels of a video. Specifically, for a given object, a CNN trained for this specific object can predict the probability of labeling to a set of spatially neighboring pixels. As a result, higher-order, richer dependencies among pixels in the set can be implicitly modeled by the CNN. Also, the model learns the optical flow of a video. However, given the nature of MRFs, the inference is very hard. So the paper came up with an algorithm to perform approximate inference in the MRF. Without any additional modules or ensembling, the model outperformed all other entries in the DAVIS 2017 Challenge.

3.2 Propagation-based models

While the models in the previous Section 3.1 finetuned on the first frame, the models mentioned in this section don't perform any finetuning on the frames whatsoever. Instead, they embed image pixels into a feature space and utilize pixel similarity to guide propagation between the frames. As there is no finetuning involved, the propagation-based models run much faster than the single frame models mentioned in the previous section. However, the lack of domain-specific finetuning leads to much worse performance in terms of quality.

3.2.1 Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning [39]

Even though the model results were significantly worse than the state-of-the-art (77.4 compared to OnAVOS's 85.5), it proposed a novel way to video object segmentation problem using metric learning. The model uses a slightly modified triplet-loss to train a feature extractor which puts pixels into separable embedding space. The inference is made using extracting embeddings from the first annotated frame. For each subsequent frame, the embeddings are extracted and classified using a simple nearest neighbor's classifier. The predicted labels are applied to the input image, and thus segmentation is obtained. Unfortunately, the model totally ignores any temporal information which could be helpful.

3.2.2 VideoMatch: Matching based Video Object Segmentation [40]

The model uses provided ground truth mask to obtain the foreground and background features which are then soft matched with the foreground and background features of all other inferred frames (one by one, though) to get the similarity score with the reference (the first) frame. Then the similarities are concatenated and softmaxed to get the final prediction. This model brings an exciting concept of soft matching later. It works in the following manner: first, we take two sets of features and compute the pairwise cosine similarity between all pairs of features. The final matching score is then produced by calculating the average of top K similarity scores. This model proved to be state-of-the-art on the Youtube-Objects and JumpCut datasets and to be competitive on DAVIS 2016 and DAVIS 2017 (being significantly worse than the best models) but with the computational time of at least one order of magnitude better than current state-of-the-art models.

3.3 Long-range spatio-temporal models

There are two main branches of the models tackling the video object segmentation problem using this approach. The first uses a recurrent neural network that uses the previous frame estimate to predict the object segmentation in the current frame. The model is then trained using backpropagation through time. These models are unfortunately prone to estimate errors from previous frames. The other branch is based on Markov Random Fields over the spatio-temporal domain of a video. This approach was popular before the rise of deep learning and is computationally expensive and cannot compete with deep learning-based models. As they are not able to compete with modern approaches, none of them will be mentioned in this thesis.

3.3.1 Fast Video Object Segmentation by Reference-Guided Mask Propagation [41]

This model uses a deep Siamese encoder-decoder network. The network has two parallel encoders which encode the target frame with the previous frame's mask and reference frame and its mask. These latent representations are then concatenated, passed through global convolution block, couple refine modules which handle upsampling and reconstruction and finally passed through a convolutional layer with softmax to get the target mask. But the exciting thing about the model is its training. The model uses two-stage training where in the first stage, the model is pre-trained on simulated samples (from datasets such as Pascal-VOC [42], ECSSD [43], and MSRA10K [44]) using two different sampling strategies. One uses various augmentations to select a pair of images to train on, and the other one uses augmentations on a pair of foreground and background objects. The second stage of training is done by fine-tuning on video data. This model scored 81.5 points on DAVIS 2016, thus not being better than the state-of-the-art but was much faster on inference than any state-of-the-art models with only approximately $130ms$ needed to process one frame.

3.3.2 MaskRNN: Instance Level Video Object Segmentation [45]

The model combines the recurrent component with segmentation and localization nets to take advantage of the temporal information and the location before improving the results. Basically, the model combines binary segmentation for each object together with effective tracking using bounding boxes. To get a bounding box proposal, the model uses optical flow estimation followed by binary segmentation. After the combination of masks from all objects in a video is used recurrently to improve the bounding box proposals even more and exploit the temporal information. To improve performance even further, online finetuning is employed in a semi-supervised setting. On DAVIS 2016 dataset, the model scored 80.38 points.

3.4 Other relevant models

3.4.1 Video Object Segmentation using Space-Time Memory Networks [15]

This is the most relevant work to the original paper. It uses very similar exploitation of dense long-term information as the original. The original paper has a much simpler implementation and doesn't require any additional datasets, and infers all objects simultaneously. The model uses past frames with object masks as encoded key, value pairs that are then so-called space-

time memory read, which means the memory gets combined with the query using a dot-product and then exponentiated power of e. This then gets put through a decoder to get the final segmentation for the query frame. The model produces a mask of scale $\frac{1}{4}$ to the size of the input image. As mentioned before, the model is unable to infer more objects simultaneously. Each object must be inferred on its own, and then the predictions get merged by soft aggregation operation. The model also requires two-stage training. First is pretrained on a simulation dataset generated from static images and is further finetuned on real-world videos. The trained model got 72.2 points in DAVIS 2017 Challenge.

Chapter 4

Proposed approach

This chapter provides an overview of the proposed approach by the thesis' author. It is divided into two subsections. The first one describes the inference-based techniques. These approaches, as the name suggests, only exploit the transductive inference procedure. Improvements proposed in this section are mainly based on combining multiple models or a combination of preprocessed video frames. The other describes improvements to training procedure which were believed to improve the embeddings produced by the model. The improvements are mainly based on using deep metric learning to train a better feature extractor than the one proposed by the original paper [5]. This chapter does not provide any benchmark results as they have their own dedicated Chapter 5.

4.1 Inference-based improvements

Most of the inference-based improvements described in this section exploit combining predictions of the original model but in different settings such as flipped video sequence as the input, etc. Only the improvement suggested in Section 4.1.1 changes the prediction flow of the original paper [5]. Due to limitations of the hardware the author had access to (NVIDIA Tesla T4 with 16GB memory), only a combination of at most two predictions (or models) could have been done.

The inference-based improvements should help build a better graph to propagate the labels on, for example, by making it bigger as proposed in Subsection 4.1.3 or by flipping the input images both horizontally or vertically.

4.1.1 Propagation of probabilities instead of one-hot encoded labels

Before describing the approach let us recollect the inference Equation 2.4 which reads as,

$$\hat{\mathbf{y}}(t+1) = \mathbf{S}_{1:t \to t+1} \hat{\mathbf{y}}(t) \tag{4.1}$$

it is important to note, that the result $\hat{\mathbf{y}}(t+1) \in \mathbb{R}^{(N_O+1)\times(m*n)}$, where N_O stands for the number of objects in the video sequence plus one for background, n, m stand for input withd and height respectively. This matrix denotes the probability of a given pixel to be one of the possible classes. During inference the prediction for frame $t+1$ is obtained by

$$\hat{\mathbf{Y}} = \arg\max \hat{\mathbf{y}}(t+1) \tag{4.2}$$

which is then transformed to one-hot encoded vector and fed into next round of prediction as $\hat{\mathbf{y}}(t+1)$. The prediction $\hat{\mathbf{Y}}$ is again in shape of $(N_O+1)\times(m*n)$.

As shown in previous equations, we might simply omit the arguments of the maxima part (the Equation 4.2) and propagate it in the next round as $\hat{\mathbf{y}}(t)$ instead. However, in order to work properly then we need to change the similarity measure from Equation 2.6 from,

$$w_{ij} = \exp\left(f_i^T f_j\right) \exp\left(-\frac{\|\mathrm{loc}(i) - \mathrm{loc}(j)\|^2}{\sigma^2}\right) \tag{4.3}$$

to

$$w_{ij} = \exp\left(f_i^T f_j\right) \tag{4.4}$$

therefore using only frame features as similarities with omitting the spatial similarity term. It is unclear why it is needed to skip the spatial term, but it was experimentally discovered that it is required in order to omit the spatial weight to work correctly.

Using probability propagation also provides an opportunity to use various fusion operations to merge the predictions. With one-hot encoded labels, one must usually use operations such as maximum, while with probabilities, one could use functions like summation, difference, or mean, for example. However, using probabilities instead of one-hot encoded labels brings potential problems with numerical instability in the propagation operation (which is matrix multiplication).

4.1.2 Test-time flip augmentations

The approach introduced as second combines predictions generated from the original video sequence with predictions generated from a horizontally mirrored video sequence. The rationale behind this approach is simple. It is believed that the embedding model could capture different relations for differently oriented images. Prediction generated from the horizontally flipped

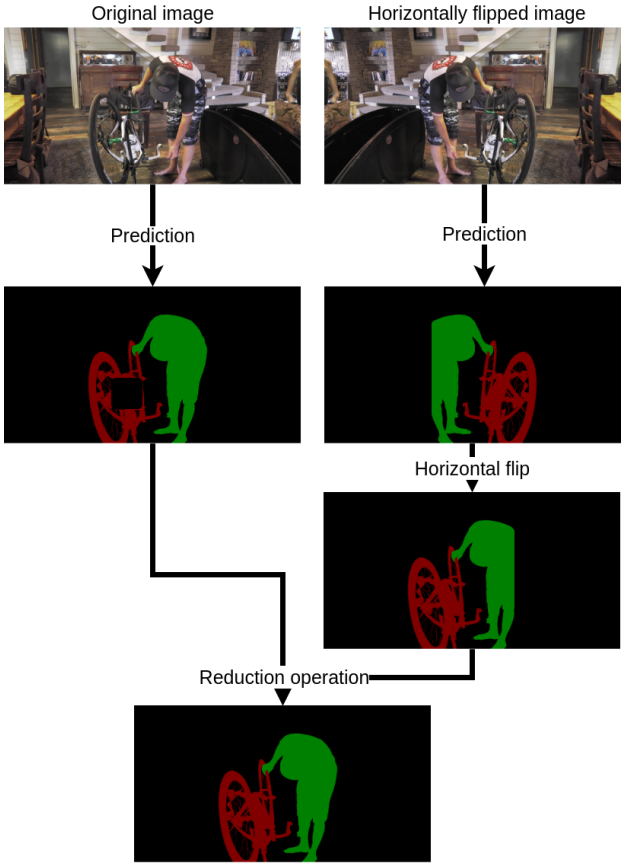

Figure 4.1: Diagram showing the horizontal test-time augmentation approach. Prediction is done using same model for both images.

video is flipped horizontally again to align with the original video. Then both predictions are merged to get the final prediction. At first, the operation used to merge the predictions was simple max as it works fine with the one-hot encoded predictions. After converting the inference code to probability propagation, three different operations were used (independently of each other) to get the final prediction. Operations used to obtain final fused prediction p from predictions p_o and p_f were:

1. $p = \max(p_o, p_f)$
2. $p = \min(p_o, p_f)$
3. $p = \frac{p_o + p_f}{2}$ – a simple arithmetic mean

The final prediction was then obtained from p by applying the argmax operation. Mean was not used as one of the fusion operations since there are only two predictions to combine, and mean would not change the outcome

of the argmax operation. The flow diagram of this approach can be seen in Figure 4.1

Additionally we propose usage of not only horizontally but also vertically flipped images, which could also help with better inference.

4.1.3 Test-time rescale augmentation

After laboring with horizontally and vertically flipped videos, it only made sense to shift attention to scaled ones. Getting more detailed and better predictions from scaled images makes sense since the model might capture more details when inferencing using different resolution inputs. The flow of the prediction was chosen to be very much the same as in previous sections dealing with flipped images. Only this time, the other input to the model would be of either smaller or higher resolution by some scale $N \geq 0$ and $N \neq 1$. However, due to the technical limitations mentioned in Section 4.1 parameter, N had to be capped at $N = 1.15$ as higher values would quickly deplete the GPU memory. Unfortunately, at the time of writing this thesis, PyTorch's support for sparse matrices was quite bad, so optimization using sparse matrices for the computations would be almost impossible to implement.

Both the original label propagation and probability propagations were implemented for this approach, together with previously mentioned fusion operations.

Even with the previously mentioned limitations, this approach performed the best of all proposed methods being significantly better than the original paper's results – more on this in Chapter 5.

4.1.4 Backbone combinations

After experimenting with combining multiple outputs from the same model, it was decided to try the combination of predictions provided by multiple different backbone models. It is believed that using predictions by two independently trained models could help to capture finer ones. The other model chosen for this approach is using ResNet-50 architecture provided by Yalniz *et al.* [46]. The model was trained on a large collection of images (up to 1 billion) in a semi-weakly supervised manner using the teacher/student paradigm. Performance of the model is 81.2 top-1 accuracy on ImageNet benchmark.

However, the model had to be slightly modified to produce embeddings of required dimension 256. To be compatible with embeddings from the thesis' model, it had to be changed somewhat. It also uses the first 8 layers as the original paper's model, but to provide same number of embeddings, the stride of layers 6 and 7 had to be set to 1 (otherwise, the model would produce only one-quarter of the required number of embeddings). Furthermore, to get embeddings of dimension 256 two convolutional layers had to be added as dimension adjustment.

Two versions of label propagation (both one-hot encoded and probability-based) were implemented, and, interestingly, the probability-based one performed better. Also, couple different fusion operations (maximum, mean, etc.) were tried to combine the predictions – more on the results in Chapter 5.

4.2 Training-based improvements

Unlike the improvements mentioned in the previous Section 4.1, improvements mentioned in this section exploit various learning techniques to improve the pixel embeddings produced by the model. In theory, training-based improvements should help enhance the pairwise similarity by learning better embeddings, which is much needed to build correctly the similarity matrix **S** mentioned in Equation 2.4.

Suggested improvements use metric learning described in Section 2.3. Using metric learning in this setting can be beneficial to learn better embeddings than the ones learned by the original cross-entropy loss proposed by [5]. Naturally, it makes sense to try one of the most straightforward techniques in the deep metric learning field – contrastive loss. It is straightforward to implement yet very powerful loss function used to model the embedding space. But it has certain limitations, so its usage can be mostly justified as a proof-of-concept before moving on more complex algorithms. Triplet loss is the next (and one of the most popular) loss function that comes to mind. Using the notion of triplets instead, it can model the embedding space even better by being aware of both positive and negative data points as well as the anchor at each update step. But as mentioned in Section 2.3, metric learning is quite demanding in terms of selected tuples of samples for learning. Therefore it makes sense to try a couple of different mining algorithms to obtain such tuples (triplets in this case).

Finally, weighed combination of cross-entropy loss with triplet loss is proposed as it should help the most to learn proper embeddings for the pixels. Triplet loss, in this case, should help build better feature space for the embeddings, while cross-entropy loss should help the model classify the pixels better.

4.2.1 Contrastive loss

The simplest idea was to use the contrastive loss described in Section 2.3.3.1 to push pixel embeddings of the same classes together and of different away from each other. While being very simple, the contrastive loss has its limitations. If two points are different, the contrastive loss pushes both data points in the opposite direction. However, this solution is not optimal if one of those points is already at its cluster center.

4.2.2 Triplet loss

Triplet loss tackles the limitations of the aforementioned contrastive loss by calculating the loss on triplets instead of pairs. The formal description of triplet loss and types of triplets is given in Section 2.3.3.2. Unfortunately, triplet loss also has some limitations. For example, in the case when all triplets are „easy“ (e.g., their loss is 0), the network's weights do not update at all. Therefore the network doesn't learn. Since for N embeddings there are $O(N^3)$ possible triplets, it would be computationally very expensive to evaluate all of them (not to mention, that majority will be either „easy“ or violate the necessity of being in form $(\text{anchor}, \text{positive}, \text{negative})$). It is generally difficult to sample meaningful triplets from training data – That's why several negative sampling (or mining) algorithms are proposed in the following subsections.

4.2.3 Triplet miners

As mentioned in the previous Section 4.2.2, the selection of triplet is crucial for triplet loss' performance. In the following subsection we propose various algorithms (called miners), that given a set $\mathcal{S}$ of N embeddings as input provide a set of triplets (e_a, e_p, e_n), such that $e_a, e_p, e_n \in \mathcal{S}$ and class of e_a is the same as e_p and different from class of e_n. Furthermore the returned triplets should have an added information value, thus not being „easy“ (as mentioned in Section 2.3.3.2).

In the case of this thesis, the set of embeddings $\mathcal{S}$ is given in a tabular form, which allows us to propose both spatially and temporally based improvements.

4.2.3.1 Spatially nearest negative miner

This miner has got its name from leveraging the spatially nearest negatives for each anchor embedding. To quickly get positions of the nearest negative for each anchor, the distance transformation operation is used (more exactly by using `SciPy`'s distance transformation function we can quickly obtain distances of the nearest negatives as well as their indices). The distance transformation is an operation applied to binary images (where 0 denotes background and 1 foreground). The operation transforms the image so that background pixels still have value 0 and the value of each foreground pixel is the distance to the nearest background pixel. One can use many different distance metrics to calculate the resulting distance. For example (with their respective formulas):

1. **Euclidean** – $d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$
2. **Manhattan** – $d(\boldsymbol{x}, \boldsymbol{y}) = |x_1 - y_1| + |x_2 - y_2|$
3. **Chessboard** – $d(\boldsymbol{x}, \boldsymbol{y}) = \max\left(|x_1 - y_1|, |x_2 - y_2|\right)$

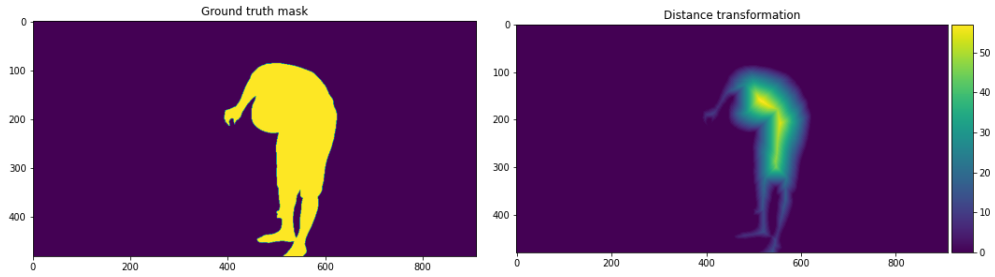

Figure 4.2: Example of distance transformation applied on a binary image (on the left). Result is on the right and shows distance of each foreground pixel to the nearest background pixel (using a color scale on the right). Metric used in this case is Manhattan.

Example application of the operation is in Figure 4.2. The implemented miner uses the Manhattan distance to calculate the distance to the nearest background. Note that the miner calculates the distance for each class in the training mask separately. This is due to the limitation of the operation to binary images only. Embeddings of all pixels that belong to objects (thus not being background) are used as anchors. For each anchor, the positive is sampled as the least similar (using cosine similarity) pixel having the same class as the anchor. Negative for each pixel is selected using the calculated distance as the spatially closest pixel of a different class.

4.2.3.2 Skeleton anchors miner

As the name suggests, the skeleton miner uses morphological skeletonization to obtain the anchors for training triplets. Simply put, skeletonization is an operation that reduces binary objects to 1 pixel-wide representations. Using this representation can be helpful for feature extraction or representing an object's topology, for example. An example of how the operation works is given in Figure 4.3. Pixels that are on the skeleton are roughly in the middle of segmented objects. Therefore we can suppose that they should be the best representation of an object and thus are used as anchors. For each object in the training mask, the skeleton is calculated, and pixel embeddings on the skeleton are selected as anchors. We then choose positive for each anchor as the least cosine similar embedding of a pixel not being on the skeleton and having the same class as the anchor. Negatives are sampled from pixel embeddings having different labels from the anchor and being most cosine similar to the anchor. Again, this way of selecting positives and negatives should minimize the probability of sampling an „easy“ triplets.

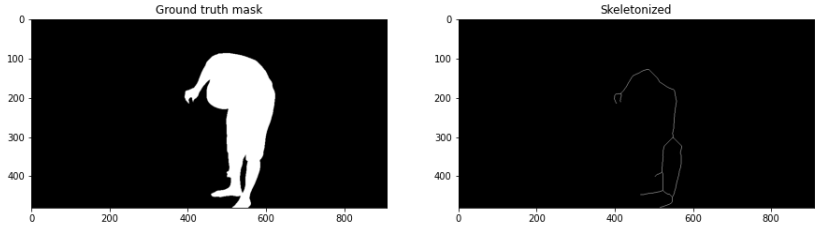

Figure 4.3: Skeletonization performed on semgentation mask given in the left picture. The right picture shows finished skeletonization.

4.2.4 Combining cross entropy loss with triplet loss

When the miner does not discover any triplet that would contribute to the loss (or would contribute very little), it is still desirable to update the network's weight nevertheless. It would only make sense to combine the proposed triplet loss with the paper's originally proposed cross-entropy. For this purpose, a simple addition of the two losses would do the trick. However, we do not want both losses to contribute with the same weight. Ideally, we want triplet loss to contribute more than the cross-entropy loss (as it is only some sort of „backup“ if the miner does not return meaningful triplets). Therefore we could replace the loss with the weighted sum of the two losses given by the following formula,

$$\mathrm{L} = \frac{w_1 * CL + w_2 * TL}{w_1 + w_2} \tag{4.5}$$

where CL is the cross-entropy loss, TL is triplet loss and w_1, w_2 are non-negative weights of the respective losses. The weighted sum is divided by the sum of the weights to rescale the loss to a more desirable range so the loss would not be too high so that the weights would be updated too much, and we would „overshoot“ the ideal weights. For simplicity we could set $w_1 = 1.0$ so we only need to find optimal w_2. This can be done using a simple grid search, for example.

4.2.5 Experimental miners

This subsection contains miners that were used for experimenting. They either restrict the spatial volume in order to sample positives from anchor's neighborhood or they extend the sampling over temporal dimension of the input video. Some of them combine approaches suggested by miners mentioned in Subsection 4.2.3. These miners are not as important as the ones mentioned previously, they are just results of experimenting with various ideas which were believed to improve the embeddings.

4.2.5.1 Kernel miner

The first experimental miner is called Kernel. The name comes from using an $n \times n$ mask M similar to a kernel known from image processing. The mask „slides“ over the embeddings (just like in 2-dimensional convolutions) to select the anchors and positives. At each step, the anchor is selected as embedding at $M_{\lfloor \frac{n}{2} \rfloor, \lfloor \frac{n}{2} \rfloor}$, then cosine similarity between the anchor and all other embeddings in the mask is calculated. The positive is then selected as the least similar embedding from the mask. After that, we calculate the cosine similarity between the anchor and all embeddings not in the mask and having a different class than the anchor. We select the negative as the most similar embedding from the set of the embeddings not in the mask and having another class.

This miner works under the assumption that most similar samples (set of positives) will always be close to the anchor (thus using the $n \times n$ mask).

However, it is essential to note that this miner does not exploit the temporal volume of the volume of input video at all. It uses only the spatial volume. Furthermore, it restricts the spatial volume to sample positives from.

4.2.5.2 Temporal miner

As mentioned, the Kernel miner does not exploit the temporal volume of the input video at all. The temporal miner, on the other hand, uses only the temporal volume and not the spatial.

It works as follows. The embeddings of the current n-th frame are all used as anchors. Embeddings from the previous four frames are then used to sample positives and negatives. Then we calculate the cosine similarity between anchors and embeddings from the previous frames. For each anchor, we sample its corresponding positive as the least similar embedding from the previous frames and the negative as the most similar embedding from the previous frames. Choosing in this manner again minimizes the probability of selecting „easy“ triplets.

4.2.5.3 One-back one-ahead miner

Just like the previous one, this miner exploits only the temporal volume of the input video. And just like the previous one, this miner also uses the embeddings from the n-th frame as anchors, but this time positives and negatives are chosen from embeddings of $(n-1)$-th and $(n+1)$-th frames. Otherwise, the sampling process remains the same as with the previous miner. The idea behind this miner is that sampling from both previous and future frames should exploit temporal volume in a better way than just sampling from previous frames.

4.2.5.4 Skeleton with nearest negative miner

Combining skeletonization operation with the nearest negative miner could yield even better results. This is where the skeleton with the nearest negative comes to the play. First, the skeleton is calculated for each object in the input image. Then the input image (containing only the current object) gets distance transformed. Embedding of each pixel of the skeleton is considered to be an anchor, and for each anchor, the positive is sampled by getting the least similar pixel embedding that has the same label but is not present on the skeleton. Negative for each anchor is obtain using the precalculated distance transformation by selecting the spatially closest pixel (again, its embedding) on the object boundary. Selecting in this manner yields fewer triplets than both skeleton and nearest negative miners on their own, but the triplets should be more exhaustive in terms of participation in the loss.

4.2.5.5 Skeleton temporal miner

This miner combines the previously mentioned Skeleton miner but this time with adding temporal volume from the Temporal miner. Functionality is basically the same as Skeleton miner but with added extra frames to sample triplets from. This miner was added solely as an experiment.

Chapter 5

Benchmarks

This chapter has two main goals. The first is to give an overview of metrics typically used to measure performance on video object segmentation tasks and datasets used for training and evaluation of the task. The other goal is to present the measured results for all proposed approaches, put them into context with the original paper's results, and discuss the results. The chapter also offers some possible directions for future work.

5.1 Metrics

In order to know how well the proposed improvements perform, we need to evaluate their performance. Typically we are given a ground truth mask G and a predicted mask M.

One of the most intuitive metrics to use would be a simple pixel accuracy. But it has some significant drawbacks that eliminate it from using in a general setting. Even if the accuracy is very high (even close to 100%), it does not necessarily mean that the approach performs well as shown in Figure 5.1, even with relatively high accuracy ($\sim$ 83% in this case), it is an unusable and awful result. So in order to trustworthily measure performance, we need to

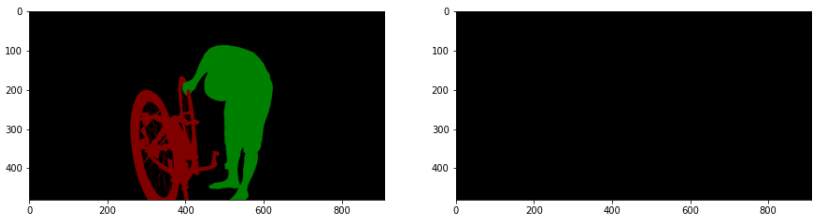

Figure 5.1: Ground truth mask G and predicted mask M. Accuracy in this case is $\sim$ 83%.

use a different metric than just accuracy. The most widely used metrics were proposed by Perazzi *et al.* in [2]. They proposed using three metrics to use to evaluate video object segmentation. The first two are based on individual images using two different points of view, region-based and contour-based. The third metric considers the temporal dimension of a video and measures the temporal stability of the predictions.

5.1.1 Region similarity $\mathcal{J}$

The first metric also referred to as the Jaccard index, is defined as the area of overlap between the predicted segmentation M and the ground truth G divided by the area of union between the predicted segmentation M and the ground truth G. Its formula is written as $\mathcal{J} = \frac{|M \cap G|}{|M \cup G|}$. A rough illustration of the calculation is shown in Figure 5.2. In layman's terms, the metric tells

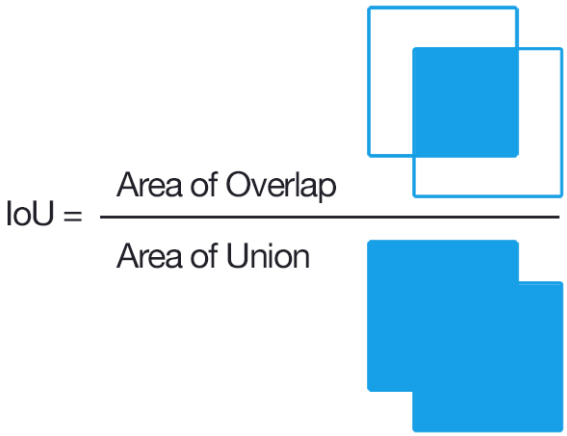

Figure 5.2: Illustration of how Jaccard index is calculated. Source: [47].

us how well is the object of interest covered by the prediction mask. In the case of multi-class evaluation, the final value is calculated as a mean of the Jaccard index for each object in the image. By design, the value of the metric is in the range $[0, 1]$, where 0 is a totally incorrect prediction, and 1 is a perfect match. It is often interpreted as percents and therefore multiplied by 100. It is one of the most widely used metrics for both image detection and image segmentation [42].

5.1.2 Contour accuracy $\mathcal{F}$

The contour accuracy $\mathcal{F}$ is also often called Sørensen–Dice coefficient. In the case of this metric, the mask is treated as a set of closed contours delimiting the spatial extent of the mask. For the ground truth mask G, we will consider a set of closed contours $c(G)$, and for predicted mask M, we will consider

another set $c(M)$. Having these two sets, we can calculate the contour-based precision P_c and recall R_c between the contour points of the finite sets using a bipartite graph matching in order to be robust to small inaccuracies. Once we have calculated P_c and R_c we can simply calculate the contour score. We define the $\mathcal{F}$ as $\mathcal{F} = \frac{2P_cR_c}{P_c+R_c}$ (note that it is similar to $F1$ metric). Again in layman's terms, this metric tells us how well the prediction copies the contours of the object (or objects) of interest. According to *Perazzi et al.* [2] and Xu *et al.* [3] this is another widely used metric for video object segmentation task.

5.1.3 Temporal stability $\mathcal{T}$

Many of the video object segmentation methods also use temporal stability to measure inaccuracy and turbulence of the predicted contours. It is measured by the dissimilarity of the target shape descriptors that describe the contour pixels between two adjacent video frames. However, DAVIS 2017 [4] doesn't use the $\mathcal{T}$ metric as the dataset contains many occlusions for which the metric is not suitable. Thus the metric is described only for completeness and is not measured in the following sections.

We show all of the used metrics – $\mathcal{J}$, $\mathcal{F}$ and $\mathcal{J}\&\mathcal{F}$ with examples in Figure 5.3. These examples come from DAVIS 2017 validation set, and the predictions are actual predictions calculated by the baseline method.

Figure 5.3: Examples of used metrics shown together with the video frame, ground truth and predicted segmentation mask.

5.2 Datasets

The original paper [5] measures performance on two different major datasets, DAVIS 2017 [4] and YouTube-VOS [3]. However, due to long training and testing times and some issues of the YouTube-VOS described in Subsection 5.2.2, this thesis measures results only on DAVIS 2017 dataset. YouTube-VOS is still described for completeness.

5.2.1 DAVIS 2017

The DAVIS dataset first appeared in 2016, but at that time, each video had only one object. Since 2017 each video sequence contains at least two objects. The videos are more complex with more distractors, smaller objects, and smaller structures, more occlusions, fast motions, etc. Overall the dataset consists of 150 sequences containing a total of 10459 annotated frames and 376 objects. Each sequence has 24 FPS. The dataset is provided in 2 resolutions, 480p and the other original (which depends on video, some are 4K, some Full HD, etc.). Due to the mentioned occlusions and difficulties in the dataset, the temporal stability - $\mathcal{T}$ metric was discarded since the 2017 edition. One of the annotated frames from the dataset is shown in Figure 5.4 it shows that compared to YouTube-VOS (in Figure 5.5), the annotation is much finer, and the video resolution is much better.

Figure 5.4: Sample frame from the DAVIS 2017 dataset. Source: [4]

5.2.2 YouTube-VOS

As claimed on their website, YouTube-VOS is the first large-scale benchmark that supports multiple video object segmentation tasks. It is directed mainly at semi-supervised video object segmentation and video instance segmentation tasks. It also has the following features:

1. 4000+ high-resolution YouTube videos
2. 90+ semantic categories

3. 7800+ unique objects
4. 190k+ high-quality manual annotations
5. 340+ minutes duration

As shown in the Figure 5.5 the quality of videos is not high though. Furthermore the annotations are worse than in the DAVIS daset. Also the framerate at 5 FPS is much lower compared to DAVIS dataset with 24 FPS.

Figure 5.5: Sample frame from the YouTube-VOS dataset. Source: [3]

5.3 Evaluation

For all approaches, both $\mathcal{J}$ and $\mathcal{F}$ metrics are calculated. Additionally, $\mathcal{J}\&\mathcal{F}$ is also calculated as mean of the $\mathcal{J}$ and $\mathcal{F}$ to provide quick and easy high-level metric. In all cases, the higher scores, the better.

All evaluations are done on DAVIS 2017 validation dataset. Training is done on DAVIS 2017 train dataset. In all cases for inference-based improvements, the model used was pretrained on DAVIS 2017 for 240 epochs using SGD optimizer with a learning rate of 0.02 and cosine annealing as learning rate scheduler. The model was trained in the same way [5] suggests. Furthermore, the models for training-based improvement were all trained for 30 epochs in order to show whether or not they improve the original model.

Measurements (in both cases of inference-based and training-based approaches) were done on *AWS g4dn.2xlarge* instance with NVIDIA Tesla T4 graphics card with 16 GB of graphics memory, 8 core Intel Xeon Platinum 8259CL CPU, and 32 GB of RAM.

5.3.1 Inference-based approaches

First, we evaluate the probability propagation, as it requires finding the optimal temperature $T \geq 1$, so we do not need to find optimal T for each of the following approaches, which uses probability propagation. Tuning of the T is

shown in Table 5.1. The optimal value is $T = 1.375$. The selection was made using a simple pseudo-binary search. Note that if the value of parameter T is set high enough (in our case, it was experimentally set as $T = 1000.0$), the probability propagation becomes very close to the original 1-hot encoded label propagation.

T	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$
0.5	0.085	0.015	0.045
1.0	0.802	0.802	0.802
2.0	0.805	0.809	0.807
5.0	0.770	0.745	0.757
10.0	0.745	0.699	0.722
1000.0	0.735	0.729	0.733
1.5	0.811	0.819	0.815
1.25	0.811	0.816	0.814
1.375	**0.824**	**0.819**	**0.821**

Table 5.1: Selecting the optimal parameter T using pseudo-binary search.

After finding the optimal temperature, we need to find the best fusion operation for combining predictions of approaches using two models or two sets of features generated by the same model (but again only for probability propagation). The available fusion operations are already mentioned in Subsection 4.1.2. Obtained results are shown in Table 5.2. Measuring was done using the horizontal flip test-time augmentation. As can be seen from Table 5.2, the best fusion operation to use is the mean with 0.81 $\mathcal{J}\&\mathcal{F}$ score. From now on when referring to probability propagation one can assume that the $\mathbf{T = 1.375}$ and used fusion operation is **mean**.

Fusion operation	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$
maximum	0.750	0.784	0.767
minimum	0.689	0.651	0.670
mean	**0.831**	**0.789**	**0.810**

Table 5.2: Selecting the best fusion operation for combining multiple predictions.

Additionally, before selecting the best inference, we need to select the optimal (or maximum feasible) scale for the rescale test-time augmentation. Table 5.3 shows various scales and their performances. Note that for a scale of 1.0, the performance is identical to the baseline approach. The best scale showed to be **1.15**. However, larger values could not be tested due to the hardware limitations mentioned at the beginning of Section 5.3. Furthermore, for

the sake of simplicity, the evaluation was done only for the 1-hot encoded label propagation and not for the probability propagation version of the algorithm.

Scale	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$
0.75	0.7830	0.7490	0.7660
0.9	0.7836	0.7565	0.7700
1.0	0.7400	0.7342	0.7371
1.05	0.7990	0.7798	0.7894
1.1	0.8164	0.8038	0.8101
1.12	0.8133	0.8005	0.8069
1.15	**0.8254**	**0.8243**	**0.8248**

Table 5.3: Selecting the best scale for the second image in rescale test-time augmentation.

Finally, after selecting optimal T for probability propagation and the best fusion operation for probability propagation, we need to select the best test-time agumentation. The available test-time augmentations were already mentioned in Section 4.1. Measured results are shown in Table 5.4, which shows results for each test-time agumentation with both probability propagation and original label propagation. The first measured test-time agumentation is only following the paper's provided baseline and is only mentioned for reference. It is interesting to see that probability propagation with mean as reducing operation performed better in all strategies but the last one. In the case of the last test-time augmentation, its poor results might be due to the fact, that as already mentioned in Subsection 4.1.4 the Facebook's model architecture was slightly different from the one used by the original model and by the fact that Facebook's model was trained with image recognition task in mind, and not for video object segmentation. However, in the original label propagation setting, the Facebook model combination performed better in terms of $\mathcal{J}$ metric than the baseline, the $\mathcal{F}$ score was worse than the baseline, mainly due to the fact that the model was not trained with the notion of contours in mind. Another interesting observation can be seen in the case of the vertically flipped image test-time agumentation. In this case, the model performed much worse compared to the horizontally flipped image test-time agumentation, which is because of a simple reason. The original model was not trained on randomly vertically flipped images (while it was trained on randomly horizontally flipped images). The overall best test-time agumentation showed to be the **scaled images** proposed in Subsection 4.1.3. It scored 0.830 combined mean score, which is the overall best of all test-time augmentations. The best-performing version was using probability propagation. But the version with original label propagation scored comparably well (being only 0.006 points overall worse).

The ideal scale for the second image in the case of this test-time agumentation was 1.15 for both probability propagation and label propagation versions.

Test-time agumentation	Probability propagation	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$
original (baseline)	✓	0.824	0.819	0.821
	✗	0.740	0.734	0.737
horizontal flip	✓	**0.831**	0.789	0.810
	✗	0.811	0.801	0.806
vertical flip	✓	0.651	0.596	0.623
	✗	0.606	0.541	0.573
rescale images	✓	0.827	**0.833**	**0.830**
	✗	0.825	0.824	0.824
backbone combination	✓	0.683	0.577	0.630
	✗	0.764	0.713	0.738

Table 5.4: Selecting the best test-time agumentation using both probability propagation and original label propagation.

Figure 5.6 shows examples of inference-based improvements on example sequence from the DAVIS 2017 validation set (called india). The first line shows predictions by baseline approach. The second and third lines show horizontal flip and resscale test-time augmentation, respectively. See, that predictions on the last line are even better and more detailed than the baseline predictions.

5.3.2 Training-based approaches

With the training-based approaches, the crucial task is to evaluate which of the proposed losses (and possibly triplet miners) is performing the best.

For brevity, we only discuss the improvements that surpassed the baseline approach. Performance of the others which were tried as experiments is mentioned in Section 5.5. The overall best mining algorithm for the triplet loss was the one using the spatially nearest negatives. The best performing version of it uses Manhattan distance as the metric for the underlying distance transformation algorithm. Even though the miner supports other metrics (such as Euclidean and Chebyshev), the Manhattan distance showed to perform best in both $\mathcal{J}$ and $\mathcal{F}$ metrics. The performance increased by 5.2% and 3.8% for the respective metrics compared to the baseline approach. Overall improvement was by 4.5%. This is not by a big margin, but in combination with the inference-based methods, the improvement could be more significant.

After finding the best miner for the triplet loss, which performed the best of all proposed losses, we need to also find the best weight for the triplet loss for the combining cross-entropy loss with triplet loss as mentioned in

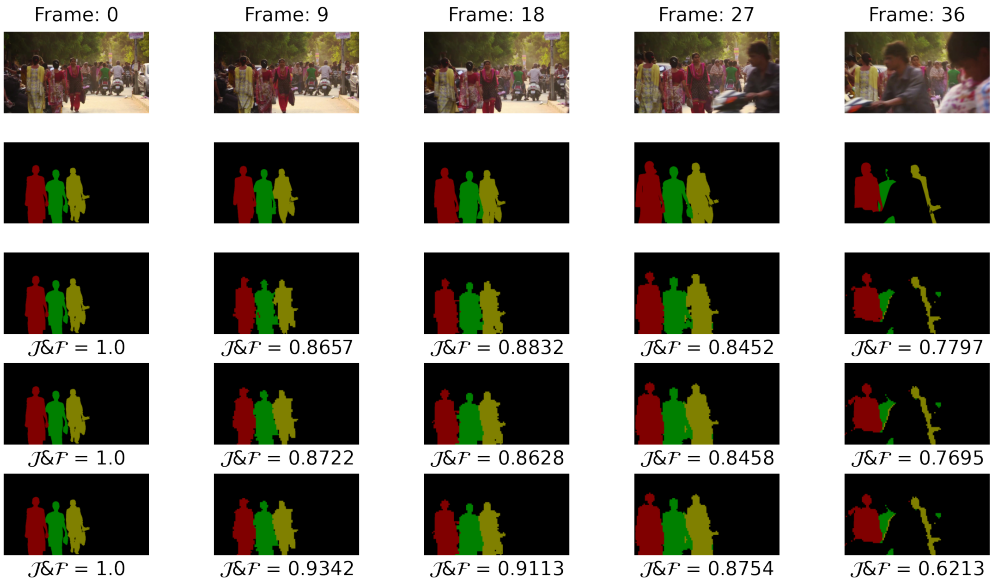

Figure 5.6: Examples of inference-based improvements, the first line shows original RGB image from the dataset, the second line shows hand-created annotation of the image and the next line contains prediction by the baseline approach. The fourth line shows predictions by horizontal flip test-time agumentation without prob. propagation. On the last line are scaled images test-time agumentation with probability propagation.

Loss	Miner	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	Loss value
Cross-entropy (baseline)		0.740	0.734	0.737	0.0812
Triplet	sp. near. negatives	**0.7806**	**0.7635**	**0.7721**	0.0262
	skeleton anchors	0.7696	0.7272	0.7484	0.0289

Table 5.5: Comparison of baseline cross-entropy loss with triplet loss using different miners.

Subsection 4.2.4. The kernel miner was used to find the weight because since it performed the worst in the finding ideal miner function, we can make sure that any improvements in the watched metrics will be due to the combination of losses and not by the miner function. Results of finding are shown in Table 5.6. If we recollect the Equation 4.5, we set $w_1 = 1.0$ (tied with the cross-entropy loss) and therefore simplify the problem only to finding only weight w_2 belonging to the triplet loss part of the equation. As the Table 5.6 shows the best weight, w_2 appears to be 8.0, which outperformed our baseline of kernel miner by almost 74%. It also outperformed the other selected weight candidates by a significant margin. Note that the weight candidates were

chosen arbitrarily.

Weight w_2	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$
1.0	0.5540	0.5287	0.5414
1.5	0.5823	0.5616	0.5719
4.0	0.6558	0.6344	0.6451
6.0	0.6712	0.6685	0.6699
8.0	**0.7027**	**0.6911**	**0.6969**
9.0	0.7003	0.6832	0.6918

Table 5.6: Comparison of different weights for the combined cross-entropy and triplet loss.

Additionally, we show examples of predicted frames. Figure 5.7 shows examples of training-based improvements on example sequence from the DAVIS 2017 validation set (called motocross-jump). The first line shows predictions by baseline approach. The second and third lines show predictions by models trained using the nearest negative miner and skeleton miner, respectively. Predictions produced by the model trained by the nearest negative miner actually better capture the details of the motorcycle and better distinguish between the motorcycle and the rider. The model trained using skeleton miner also very well distinguishes between the motorcycle and the rider, but does not cover the objects as good as the previous model.

5.4 Combining best inference-based and training-based improvements

After finding both the best inference and training-based improvements, it only makes sense to measure their combined performance. Table 5.7 shows the combination of scaled test-time agumentation with a model trained using the nearest negative miner. The table shows both versions with probability propagation and with the original 1-hot encoded label propagation. As expected, both versions performed as good as the proposed improvements on their own, but again the probability propagation version showed to be slightly better than the original 1-hot encoded label propagation.

Probability propagation	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$
✗	0.8143	0.8090	0.8116
✓	0.8386	0.8261	0.8323

Table 5.7: Comparison of different weights for the combined cross-entropy and triplet loss.

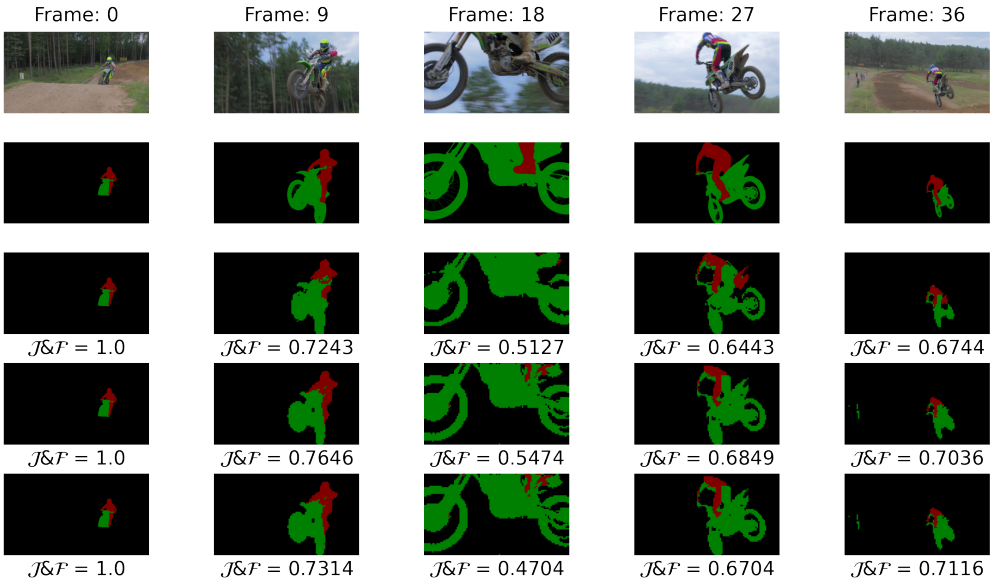

Figure 5.7: Examples of training based improvements. The first line shows original RGB image from the dataset, the second line shows hand-created annotation of the image and the next line contains prediction by the baseline approach. The next line shows predictions by a model trained using the nearest negative miner. The last line depicts predictions produced by model trained using the skeleton miner.

Figure 5.8 shows predictions obtained by the combination of the best methods. It can be clearly seen that with the probability propagation, the model was able to capture much more details, and there would not be any blank spots in the middle of objects, unlike with the original label propagation. Both versions handled very well disappearance of the objects from the video for a short while.

5.5 Experiments

In this section, we evaluate the experimental losses and miners. All of the mentioned losses and miners performed worse than the baseline, yet they provide an area to focus on in possible future work.

One of the losses tried at the very beginning was the focal loss [48]. The focal loss is supposed to help learn better classifier of pixels as it is more suitable for learning on highly imbalanced datasets, which a segmentation mask certainly is (most of the segmentation mask is background, usually shown in black color in the segmentation masks). Focal loss is simply an extension of the cross-entropy loss adding a modulation factor $(1-p_t)^\gamma$ making the formula

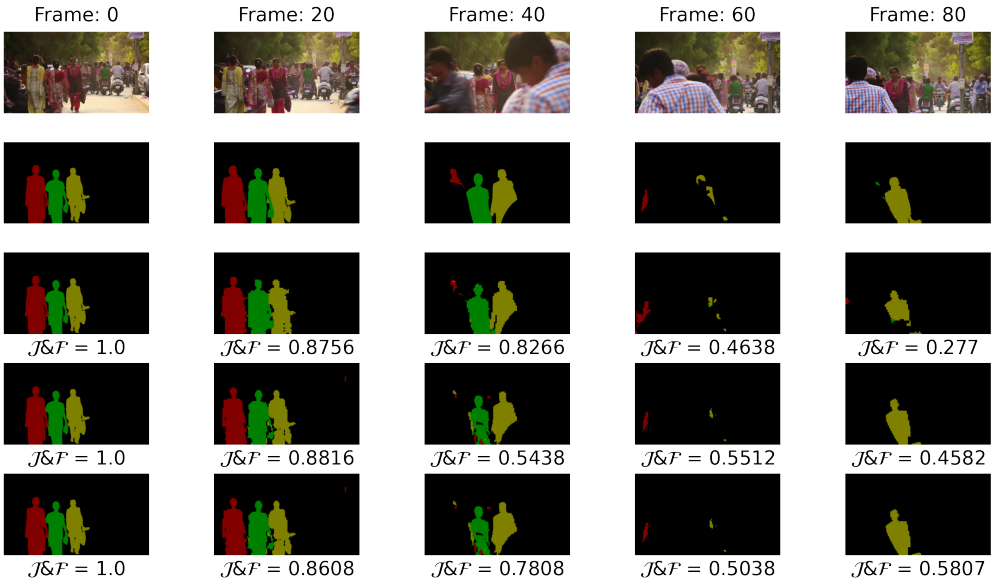

Figure 5.8: Examples of predictions obtained by combination of best proposed improvements. The first line shows original RGB image from the dataset, the second line shows hand-created annotation of the image and the next line contains prediction by the baseline approach for reference. The other two lines show predictions obtained by combining scaled images test-time agumentation with model trained using the nearest negative miner. Additionally both versions with original label propagation and with probability propagation are shown.

of the loss as follows,

$$\mathrm{FL}(p_t) = -\alpha_t(1-p_t)^\gamma \log(p_t) \tag{5.1}$$

where $\alpha_t \in [0,1]$ is a weighting factor for class t and $\gamma \geq 0$ is focusing parameter. The modulating factor causes the loss to give hard samples (false negatives) more weight than easy samples (true negatives). However, as shown in Table 5.8 it did not perform even as good as the baseline. This could be caused by the incorrectly set parameter γ, which could be more explored as a part of future work.

Another of the experimental losses used was the contrastive loss, which performed better in terms of the $\mathcal{J}$ metric than the baseline, but was much worse in terms of the contour accuracy $\mathcal{F}$.

Additionally mentioned here are some of the experimental miners that were tried as a part of an effort to learn better pixel embeddings by exploiting either temporal volume of input videos, restrict spatial volume to sample from, or combined multiple approaches from simpler miners. Neither of them performed even as good as the baseline.

Loss	Miner	$\mathcal{J}$	$\mathcal{F}$	$\mathcal{J}\&\mathcal{F}$	Loss value
Cross-entropy (baseline)		0.740	0.734	0.737	0.0812
Focal		0.740	0.680	0.710	0.0921
Contrastive		0.753	0.675	0.714	0.1108
Triplet	kernel	0.507	0.295	0.401	0.5670
	temporal	0.7619	0.6964	0.7291	0.0419
	1-back-1-ahead	0.7480	0.6981	0.7230	0.0417
	skeleton dist. trans.	0.7607	0.7164	0.7385	0.0268
	skeleton temporal	0.7398	0.6944	0.7171	0.0284

Table 5.8: Comparison of different performance on validation set of losses (and miners if applicable).

5.6 Discussion and future work

Most of the evaluation of the inference-based approaches dealt with finding optimal settings for probability propagation. It turned out that the best temperature for it is $T = 1.375$, which showed the best $\mathcal{F}$ and $\mathcal{J}\&\mathcal{F}$ metrics. Regarding the best fusion operation for a combination of more predictions (or models) in the probability propagation setting, the best operation by far turned out to be simple arithmetic **mean** with $\mathcal{J}\&\mathcal{F} = 0.8$. Selecting the best test-time agumentation (shown in Table 5.4) demonstrated that in all strategies but the last one, the probability propagation outperformed the baseline approach from [5]. The best test-time agumentation turned out to be the scaled images with a scale of 1.15. It is possible that a larger scale could yield better results, but it was not possible to evaluate larger scale due to hardware limitations.

In terms of future work for the inference-based improvements, there's an area to explore the usage of scaled images. Using scaled images allows the model to focus on both high-level interpretation (with the original „small“ image) as well as finer and finer details with the upscaled images. With GPU having more memory than 16 GB (or being able to inference on multiple GPUs), more upscaled images could be used (both in terms of numbers of images and scales). Another improvement might be switching over to sparse tensors. As of writing this thesis, `Pytorch 1.8.1` doesn't support enough sparse tensors operations to infer using only them and save some GPU memory.

With the training-based approaches, the main task was to select the best loss and, if that loss would be the triplet loss, also select the best mining algorithm for getting training triplets from available embeddings. It turned out that the contrastive loss did not even perform comparably to the baseline.

However, the triplet loss showed that it is capable of learning better embeddings than the original approach. Especially with the nearest negative miner (see Subsection 4.2.3.1), it outperformed the baseline approach by almost 5% in the mean $\mathcal{J}\&\mathcal{F}$ metric with a score of 0.7721 vs. 0.737.

As mentioned in the previous section, the training-based approaches relied solely on different training losses. For future work, there are still some areas of metric learning worth exploring, for example, novel Circle Loss [49], which claims to find better embeddings than the triplet loss. Another area of interest could be finding optimal hyperparameters of the triplet loss and the miners (such as margin m, limited number of triplets, using other similarity metric than cosine similarity, etc.). Another area worth exploring is using different architecture for extracting the embeddings of input images like recent DINO model [50], which claims to achieve the-state-of-the-art performance on DAVIS 2017 in video object segmentation task using self-supervised learning, leveraging the recent Vision Transformers [51].

Just as expected, the best results were obtained by the combination of proposed improvements using the scaled images test-time agumentation and model trained using the nearest negative miner. It scored $\mathcal{J}\&\mathcal{F}$ of 0.8323, which is better than the model trained using the aforementioned miner on its own, and it is also a slight improvement over the scaled images test-time agumentation.

Another important discussion is about the $\mathcal{J}$ and $\mathcal{F}$ metrics. In certain cases, for example, with the contrastive loss, it happens that even though the $\mathcal{J}$ rises, the $\mathcal{F}$ falls. This is due to the fact that models trained using some of the miners or losses fail are able to correctly predict the pixels inside the objects but have problems with object boundaries (they are usually quite jaggedy). Opposite situation interestingly never occurred during the measurements for this thesis.

Conclusion

The goals of this thesis were to summarize the Zhang *et al.*'s paper *A Transductive Approach for Video Object Segmentation*, provide an overview on background knowledge needed to understand it, reproduce the paper's results, and suggest possible improvements to the approach introduced by the original paper.

The thesis summarizes the original paper and provides an introduction to label propagation and metric learning. Both of these topics are important for understanding the proposed improvements. Furthermore, the thesis provides an overview of the current methods of solving the video object segmentation problem, including the current state-of-the-art methods.

Two streams of improvements are proposed. The first stream focuses on inference-based approaches to be able to build better and exploit the proximity graph that is created to propagate the labels. The other focuses on training-based techniques to improve the model for generating pixel embeddings. These approaches are based on training using the triplet loss with using various triplet mining algorithms to obtain better triplets for training the model.

Lastly, the results of the original paper were reproduced, together with measurements of the suggested improvements. Inference-based improvements were able to improve the $\mathcal{J}$ metric by 12% and the $\mathcal{F}$ by 13%. Training-based improvements were able to improve the metrics by 5% and 4%, respectively. The best inference-based improvement propagates probabilities rather than 1-hot encoded labels to improve the label propagation and additionally uses multiple scaled input images instead of one as the original approach. The best training-based improvement replaces the original cross-entropy loss with triplet loss and adds novel triplet mining algorithms to improve performance. In the end, the measured results are discussed, and possible topics for future works are provided.

Bibliography

1. RUSSELL, Stuart; NORVIG, Peter. *Artificial Intelligence: A Modern Approach*. 3rd ed. Prentice Hall, 2010.
2. PERAZZI, F.; PONT-TUSET, J.; MCWILLIAMS, B.; VAN GOOL, L.; GROSS, M.; SORKINE-HORNUNG, A. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 724–732. Available from DOI: 10.1109/CVPR.2016.85.
3. XU, Ning; YANG, Linjie; FAN, Yuchen; YUE, Dingcheng; LIANG, Yuchen; YANG, Jianchao; HUANG, Thomas. *YouTube-VOS: A Large-Scale Video Object Segmentation Benchmark*. 2018. Available from arXiv: 1809.03327 [cs.CV].
4. PONT-TUSET, Jordi; PERAZZI, Federico; CAELLES, Sergi; ARBELÁEZ, Pablo; SORKINE-HORNUNG, Alex; GOOL, Luc Van. *The 2017 DAVIS Challenge on Video Object Segmentation*. 2018. Available from arXiv: 1704.00675 [cs.CV].
5. ZHANG, Yizhuo; WU, Zhirong; PENG, Houwen; LIN, Stephen. *A Transductive Approach for Video Object Segmentation*. 2020. Available from arXiv: 2004.07193 [cs.CV].
6. ZHOU, D.; BOUSQUET, O.; LAL, TN.; WESTON, J.; SCHÖLKOPF, B. Learning with Local and Global Consistency. In: *Advances in Neural Information Processing Systems 16*. Cambridge, MA, USA: MIT Press, 2004, pp. 321–328.
7. HE, Kaiming; ZHANG, Xiangyu; REN, Shaoqing; SUN, Jian. *Deep Residual Learning for Image Recognition*. 2015. Available from arXiv: 1512.03385 [cs.CV].
8. RUSSAKOVSKY, Olga et al. *ImageNet Large Scale Visual Recognition Challenge*. 2015. Available from arXiv: 1409.0575 [cs.CV].

9. WANG, Limin; XIONG, Yuanjun; WANG, Zhe; QIAO, Yu; LIN, Dahua; TANG, Xiaoou; GOOL, Luc Van. *Temporal Segment Networks: Towards Good Practices for Deep Action Recognition*. 2016. Available from arXiv: 1608.00859 [cs.CV].

10. LOSHCHILOV, Ilya; HUTTER, Frank. *SGDR: Stochastic Gradient Descent with Warm Restarts*. 2017. Available from arXiv: 1608.03983 [cs.LG].

11. BAO, Linchao; WU, Baoyuan; LIU, Wei. *CNN in MRF: Video Object Segmentation via Inference in A CNN-Based Higher-Order Spatio-Temporal MRF*. 2018. Available from arXiv: 1803.09453 [cs.CV].

12. LI, Xiaoxiao; LOY, Chen Change. *Video Object Segmentation with Joint Re-identification and Attention-Aware Mask Propagation*. 2018. Available from arXiv: 1803.04242 [cs.CV].

13. LUITEN, Jonathon; VOIGTLAENDER, Paul; LEIBE, Bastian. *PReMVOS: Proposal-generation, Refinement and Merging for Video Object Segmentation*. 2018. Available from arXiv: 1807.09190 [cs.CV].

14. VOIGTLAENDER, Paul; CHAI, Yuning; SCHROFF, Florian; ADAM, Hartwig; LEIBE, Bastian; CHEN, Liang-Chieh. *FEELVOS: Fast End-to-End Embedding Learning for Video Object Segmentation*. 2019. Available from arXiv: 1902.09513 [cs.CV].

15. OH, Seoung Wug; LEE, Joon-Young; XU, Ning; KIM, Seon Joo. *Video Object Segmentation using Space-Time Memory Networks*. 2019. Available from arXiv: 1904.00607 [cs.CV].

16. ILG, Eddy; MAYER, Nikolaus; SAIKIA, Tonmoy; KEUPER, Margret; DOSOVITSKIY, Alexey; BROX, Thomas. *FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks*. 2016. Available from arXiv: 1612.01925 [cs.CV].

17. ZHU, Xiaojin; GHAHRAMANI, Zoubin. *Learning from Labeled and Unlabeled Data with Label Propagation*. 2002. Technical report.

18. NEEDHAM, Mark; HODLER, Amy E. *Graph Algorithms in Neo4j: Label Propagation*. 2019. Available also from: https://neo4j.com/blog/graph-algorithms-neo4j-label-propagation/.

19. CHAPELLE, Olivier; SCHLKOPF, Bernhard; ZIEN, Alexander. *Semi-Supervised Learning*. 1st. The MIT Press, 2010. ISBN 0262514125.

20. LIU, Yanbin; LEE, Juho; PARK, Minseop; KIM, Saehoon; YANG, Eunho; HWANG, Sung Ju; YANG, Yi. *Learning to Propagate Labels: Transductive Propagation Network for Few-shot Learning*. 2019. Available from arXiv: 1805.10002 [cs.LG].

21. ISCEN, Ahmet; TOLIAS, Giorgos; AVRITHIS, Yannis; CHUM, Ondrej. *Label Propagation for Deep Semi-supervised Learning*. 2019. Available from arXiv: 1904.04717 [cs.CV].

22. BELLET, Aurélien; HABRARD, Amaury; SEBBAN, Marc. *A Survey on Metric Learning for Feature Vectors and Structured Data*. 2014. Available from arXiv: 1306.6709 [cs.LG].

23. CHEN, Ting; KORNBLITH, Simon; NOROUZI, Mohammad; HINTON, Geoffrey. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. Available from arXiv: 2002.05709 [cs.LG].

24. MAHALANOBIS, Prasanta Chandra. On the generalized distance in statistics. In: 1936.

25. GRAY, Alfred; ABBENA, Elsa; SALAMON, Simon. *Modern Differential Geometry of Curves and Surfaces with Mathematica, Third Edition (Studies in Advanced Mathematics)*. Chapman & Hall/CRC, 2006. ISBN 1584884487.

26. SCHROFF, Florian; KALENICHENKO, Dmitry; PHILBIN, James. FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015. ISBN 9781467369640. Available from DOI: 10.1109/cvpr.2015.7298682.

27. CHOPRA, Sumit; HADSELL, Raia; LECUN, Yann. Learning a similarity metric discriminatively, with application to face verification. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, vol. 1, pp. 539–546.

28. HOFFER, Elad; AILON, Nir. *Deep metric learning using Triplet network*. 2018. Available from arXiv: 1412.6622 [cs.LG].

29. KHOSLA, Prannay; TETERWAK, Piotr; WANG, Chen; SARNA, Aaron; TIAN, Yonglong; ISOLA, Phillip; MASCHINOT, Aaron; LIU, Ce; KRISHNAN, Dilip. *Supervised Contrastive Learning*. 2021. Available from arXiv: 2004.11362 [cs.LG].

30. CHEN, Weihua; CHEN, Xiaotang; ZHANG, Jianguo; HUANG, Kaiqi. *Beyond triplet loss: a deep quadruplet network for person re-identification*. 2017. Available from arXiv: 1704.01719 [cs.CV].

31. *Understanding Ranking Loss, Contrastive Loss, Margin Loss, Triplet Loss, Hinge Loss and all those confusing names* [online] [visited on 2021-04-08]. Available from: https://gombru.github.io/2019/04/03/ranking_loss/.

32. KULIS, Brian. Metric Learning: A Survey. *Foundations and Trends® in Machine Learning*. 2013, vol. 5, no. 4, pp. 287–364. ISSN 1935-8237. Available from DOI: 10.1561/2200000019.

33. ZHAI, Andrew; WU, Hao-Yu; TZENG, Eric; PARK, Dong Huk; ROSENBERG, Charles. Learning a Unified Embedding for Visual Search at Pinterest. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Anchorage, AK, USA: Association for Computing Machinery, 2019, pp. 2412–2420. KDD '19. ISBN 9781450362016. Available from DOI: 10.1145/3292500.3330739.

34. CAELLES, Sergi; MANINIS, Kevis-Kokitsi; PONT-TUSET, Jordi; LEAL-TAIXÉ, Laura; CREMERS, Daniel; GOOL, Luc Van. *One-Shot Video Object Segmentation*. 2017. Available from arXiv: 1611.05198 [cs.CV].

35. SIMONYAN, Karen; ZISSERMAN, Andrew. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. Available from arXiv: 1409.1556 [cs.CV].

36. VOIGTLAENDER, Paul; LEIBE, Bastian. *Online Adaptation of Convolutional Neural Networks for Video Object Segmentation*. 2017. Available from arXiv: 1706.09364 [cs.CV].

37. MANINIS, Kevis-Kokitsi; CAELLES, Sergi; CHEN, Yuhua; PONT-TUSET, Jordi; LEAL-TAIXÉ, Laura; CREMERS, Daniel; GOOL, Luc Van. *Video Object Segmentation Without Temporal Information*. 2018. Available from arXiv: 1709.06031 [cs.CV].

38. HE, Kaiming; GKIOXARI, Georgia; DOLLÁR, Piotr; GIRSHICK, Ross. *Mask R-CNN*. 2018. Available from arXiv: 1703.06870 [cs.CV].

39. CHEN, Yuhua; PONT-TUSET, Jordi; MONTES, Alberto; GOOL, Luc Van. *Blazingly Fast Video Object Segmentation with Pixel-Wise Metric Learning*. 2018. Available from arXiv: 1804.03131 [cs.CV].

40. HU, Yuan-Ting; HUANG, Jia-Bin; SCHWING, Alexander G. *VideoMatch: Matching based Video Object Segmentation*. 2018. Available from arXiv: 1809.01123 [cs.CV].

41. OH, S. W.; LEE, J.; SUNKAVALLI, K.; KIM, S. J. Fast Video Object Segmentation by Reference-Guided Mask Propagation. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2018, pp. 7376–7385. Available from DOI: 10.1109/CVPR.2018.00770.

42. EVERINGHAM, Mark; GOOL, Luc; WILLIAMS, Christopher K.; WINN, John; ZISSERMAN, Andrew. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision*. 2010, vol. 88, no. 2, pp. 303–338. ISSN 0920-5691. Available from DOI: 10.1007/s11263-009-0275-4.

43. SHI, J.; YAN, Q.; XU, L.; JIA, J. Hierarchical Image Saliency Detection on Extended CSSD. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2016, vol. 38, no. 4, pp. 717–729. Available from DOI: 10.1109/TPAMI.2015.2465960.

44. CHENG, M.; ZHANG, G.; MITRA, N. J.; HUANG, X.; HU, S. Global contrast based salient region detection. In: *CVPR 2011*. 2011, pp. 409–416. Available from DOI: 10.1109/CVPR.2011.5995344.

45. HU, Yuan-Ting; HUANG, Jia-Bin; SCHWING, Alexander G. *MaskRNN: Instance Level Video Object Segmentation*. 2018. Available from arXiv: 1803.11187 [cs.CV].

46. YALNIZ, I. Zeki; JÉGOU, Hervé; CHEN, Kan; PALURI, Manohar; MAHAJAN, Dhruv. *Billion-scale semi-supervised learning for image classification*. 2019. Available from arXiv: 1905.00546 [cs.CV].

47. ROSEBROCK, Adrian. *Intersection over Union (IoU) for object detection*. 2021. Available also from: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

48. LIN, Tsung-Yi; GOYAL, Priya; GIRSHICK, Ross; HE, Kaiming; DOLLÁR, Piotr. *Focal Loss for Dense Object Detection*. 2018. Available from arXiv: 1708.02002 [cs.CV].

49. SUN, Yifan; CHENG, Changmao; ZHANG, Yuhan; ZHANG, Chi; ZHENG, Liang; WANG, Zhongdao; WEI, Yichen. *Circle Loss: A Unified Perspective of Pair Similarity Optimization*. 2020. Available from arXiv: 2002.10857 [cs.CV].

50. CARON, Mathilde; TOUVRON, Hugo; MISRA, Ishan; JÉGOU, Hervé; MAIRAL, Julien; BOJANOWSKI, Piotr; JOULIN, Armand. *Emerging Properties in Self-Supervised Vision Transformers*. 2021. Available from arXiv: 2104.14294 [cs.CV].

51. DOSOVITSKIY, Alexey et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2020. Available from arXiv: 2010.11929 [cs.CV].

Appendix A

Implementational details

The project is implemented using `Python 3.8.5` and with `PyTorch 1.8.1`. Other dependencies are listed in the `requirements.txt` file. The original code accompanying the Zhang *et al.* [5] can be found on Microsoft's GitHub repository (`https://github.com/microsoft/transductive-vos.pytorch`).

Source codes of the project are available on the author's GitHub repository (`https://github.com/hynekdav/semi-supervised-VOS`).

Some of the pretrained models are available on Google Drive (`https://drive.google.com/drive/folders/1MTbu9-6tfPvF9pbBxByQmUrqcyxs5sTW?usp=sharing`).

A.1 Prerequisites

First of all `Python` version at least 3.8 must be installed (`https://www.python.org`). Then install `pip` from `https://pip.pypa.io/en/stable/`, which is required to install the project's dependencies. Run following command to install the dependencies:

```
pip install -r requirements.txt
```

A.2 Usage

The main application entrypoint `main.py` supports 4 basic commands:

- `train`
- `inference`
- `validation`
- `evaluation`

The application is written using `Click` library, so every command has automatically generated help pages. Each can be invoked by running:

```
python main.py <command> --help
```

Examples of running each command are in file `example.sh`.

Additionally, the project offers various visualizations of predicted frames. Available commands for visualizations are:

- `overlay`
- `side-by-side`
- `prediction-only`

The visualizations are invoked same way as the main entrypoint:

```
python visualization.py <command> --help
```

Appendix B

Acronyms

SSL Semi-supervised learning

CNN Convolutional neural network

VOS Video object segmentation

FPS Frames per second

SGD Stochastic gradient descend

TVOS Transductive video object segmentation

DAVIS Densely Annotated Video Segmentation

IoU Intersection over union

Appendix C

Contents of attached CD

- `models` ..... directory containing pretrained models
- `src`
  - `impl` ..... directory containing project's source code in Python
  - `thesis` ..... directory containing thesis' text source code in LaTeX
- `text`
  - `thesis.pdf` ..... the thesis in PDF format