SAPIENZA
UNIVERSITÀ DI ROMA

Self-supervised Representation Learning via Contrastive Walks in Videos

Faculty of Information Engineering, Informatics, and Statistics

Master's degree in Data Science

Candidate

Paolo Mandica

ID number 1898788

Thesis Advisor

Prof. Fabio Galasso

Academic Year 2020/2021

Thesis defended on 27 October 2021
in front of a Board of Examiners composed by:

Prof. Tardella Luca (chairman)
Prof. Barbarossa Sergio
Prof. Chatzigiannakis Ioannis
Prof. Fiscon Julia
Prof. Galasso Fabio
Prof. Leonardi Stefano
Prof. Palagi Laura

Self-supervised Representation Learning via Contrastive Walks in Videos
Master's thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: mandica.1898788@studenti.uniroma1.it

A Federico,
chissà come stai lassù ogni notte.

Abstract

Self-supervised representation learning from video is an expanding topic in the field of computer vision which focuses on learning representations from raw video without the use of labeled data. Advancements on the task have been held back for some time due to video being modeled as a simple extension of images in time, thus not taking into account the concept of temporal correspondence. This thesis work builds upon previous research on the topic and investigates a novel approach to learning a representation from video in a self-supervised way. Video is modeled as a space-time graph where nodes are objects present in each frame and edges link objects from adjacent frames. The pairwise similarity between two nodes is expressed as an edge weight and defines the transition probability of a random walker transitioning from one node to the other. A complete walk along the space-time graph allows to obtain the long-range correspondence between objects in the frames. Self-supervision was introduced in the training process through the generation of palindrome sequences, exploiting the concept of cycle-consistency in time, which lead to have as objective the maximization of the likelihood of returning to the initial node after performing a random walk along the palindrome graph. The novel approach proposed in this thesis consists in using objects present in each frame as nodes of the space-time graph. Objects are extracted via superpixel segmentation algorithms, which allow to keep the whole training process unsupervised. Superpixels are groups of pixels whose main advantage consists in the reduction of the number of image primitives for subsequent processing; this should lead to a reduction of training times up to 5 fold and a faster convergence of the loss during training. The learned representation is then used to compute the similarity between frames in the evaluation process, consisting in label propagation over the task of video object segmentation. In fact, the experiments performed show that training models using superpixels as nodes of the graph allows to use larger batch sizes, reduces the per-epoch training duration and causes the loss to converge much faster. In particular, the combination of patches and superpixels produces a model which performs, on the video object segmentation task, almost on par with the original one, while reducing the time needed to train it by more than 30%. Research on the topic of modeling video as an object-based space-time graph using superpixel segmentation is still in progress and the initial results presented in this thesis give optimistic perspectives for future developments.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor, Professor Fabio Galasso, for giving me the opportunity to work on this research project and providing invaluable guidance during the development of this work.

I would like to acknowledge my colleagues, Luca and Anil, for their wonderful collaboration. Without people like you, who work with passion and enthusiasm, this research project would have not come to life as it is. A thank you also goes to all my colleagues working in the PINLab research group.

I would also like to thank all the professors from whom I had the honor to learn valuable skills and lessons throughout the course of the past two years, and all the colleagues I had the fortune to work with.

Last but not least, I am extremely grateful to my sister and my parents for their love, caring and support during my studies. I am also thankful to all the family members and friends who stood by my side in good and bad times. A special thank goes to my friend Giulia, who put me in the right steps for following this academic path and who continuously inspires me both as a student and as a person.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Inspired by the human ability of visualizing and understanding the world around us through the transformation of complex visual scenes into meaningful representations, the field of Artificial Intelligence applied to Computer Vision has been steadily evolving in recent years. One of the most important tasks, if not the most important one, consists in learning visual representations of images and videos in order to transfer the knowledge to other, more specific, downstream tasks, such as instance segmentation, object detection, human pose forecasting. While research has produced, so far, exceptional results related to image representation learning, the task of learning representations from videos is still in an intermediate progress stage. There are multiple causes for this slow progress in the sphere of video representation learning, but two specific reasons can be highlighted as the main ones:

- Supervised representation learning from video requires huge amount of data. The creation of labeled datasets involves a massive human effort in labeling by hand hundreds of images for thousands of videos and, consequently, large amounts of money are need for the development of just a single dataset.
- Video is still treated as a simple repetition of images, thus as a mere extension of image in time. But, modeling video as a XYT dimensional space, where T is just another dimension, leads to not taking into account the concept of *temporal correspondence* and, thus, limiting the ability of learning useful video representations.

A solution to the first problem consists in leveraging *self-supervision* in order

to learn from huge amounts of data without the need of human made labels. In fact, self-supervised learning exploits labels that are naturally part of the input data instead of relying on external ones. Moreover, training a model in a self-supervised fashion demands the choice of an adequate loss function, such as a *contrastive loss*. The framework coming out from the combination of self-supervised learning and the contrastive loss can be defined as *self-supervised contrastive learning*.

In contrast, the second problem is harder to solve. Many approaches have been developed in recent times, but few have obtained good enough results when it comes to representing video. The idea that this work proposes consists in depicting video as a space-time graph where nodes are objects in a frame and weighted edges link objects in adjacent frames through a similarity metric. With this type of formulation, the task of learning a representation comes down to walking through the space-time graph along the time dimension and fitting the transition probability matrix of the walker. Self-supervision is introduced via the generation of palindrome sequences at training time, which allow to define the target of the walk as the actual query node. Moreover, object masks inside each frame are obtained by using *superpixel segmentation* algorithms, which should lead to improvements in terms of training times and convergence rates while also being coherent with the self-supervised paradigm.

To better summarize, this thesis goal is to investigate the possibility of creating a pretrained (BERT-like) neural network for visual representation of videos by exploiting self-supervision, contrastive learning, and a very large dataset. The learned model should then be able to be used as backbone for learning downstream tasks such as object segmentation, label propagation and human pose estimation. Here is a brief description of what each chapter of this work focuses on.

In Chapter 2 I introduce and review the research literature behind this thesis. An in-depth description of the concepts of self-supervised representation learning, contrastive learning, cycle-consistency of time and superpixel segmentation is provided in order to give the reader most of the knowledge needed to follow the development of the work.

In Chapter 3 a formalization of the problem is presented, followed by a brief description of the work from Jabri et al. [17]. Their paper proposes a *patch-based* approach to represent video as a space-time graph in order to implement a contrastive learning framework for learning video representation. This serves as basis for the

presentation of the core idea behind this work, that is the introduction of a *novel object-based contrastive walk* approach whose goal is to solve some problems present in the patch-based approach and to allow the model to learn to link objects between frames instead of square patches.

In Chapter 4 I present the datasets used for training and evaluation, plus the metrics used to measure the performance of the model. After that, the hyper-parameters used for training and testing the model are listed. Finally, I describe the evaluation process of the model over the task of label propagation on video object segmentation, I present the results obtained and I compare them with the ones from the original model.

In Chapter 5 I conclude the thesis by summarizing the work done and the results obtained until now. I present some ideas for future developments, since the project is still under development, and I provide suggestions for practical applications.

Chapter 2

Background

This chapter provides a review of the relevant literature behind the work presented in this thesis. First, an introduction to *self-supervised representation learning* is given in Section 2.1, which includes the concepts of *representation learning* and *self-supervision*. *Self-supervised contrastive learning* is then reviewed in Section 2.2, followed by a brief description of the key concept of *cycle-consistency of time* in Section 2.3. Finally, some state-of-the-art *superpixel segmentation* methods are discussed in Section 2.4.

2.1 Self-supervised Representation Learning in Videos

2.1.1 Representation Learning

The performance of any machine learning or information processing task is heavily dependent on how the data is represented. In classical machine learning, the process of choosing the features that better represent the data is called *feature engineering* and consists in designing data processing pipelines for the evaluation and transformation of features in order to support and improve the subsequent learning task. Different types of data representations are suited for different learning tasks.

With the advent of Deep Learning and the increased use of neural networks in fields like Computer Vision and Natural Language Processing (NLP), classical feature engineering techniques rapidly became obsolete, giving rise to the field of *representation learning via Deep Neural Networks*. Complex data sources such as

images, videos and audio need to be represented in more abstract, and mathematically useful, ways in order to make it easier for deep learning models to extract useful information from them and to leverage it in learning a large variety of downstream tasks such as classification, segmentation, pattern recognition.

In the last few years there have been a lot of advances in representation learning from still images. Deep learning image models based on the ImageNet and COCO datasets [15, 32] have now reached incredible accuracy levels in downstream tasks, such as classification [1, 47] and object detection, thanks to their ability to build representations. This advancements, however, have not been fully translated into the sphere of video representation learning. There are various reasons that lead to this gap: first, creating labeled video datasets is a time consuming task which involves labeling by hand hundreds of frames for thousands of videos. Moreover, videos are still treated as a simple extension of images in the time domain. Modeling videos as XYT spatio-temporal volumes, where time is just an extra dimension, limits the ability of models to learn better representations. The main reason for this is that a pixel in position (x, y) in frame j may not have any relationship with the pixel in the same position in the next frame, $j + 1$. Thus, modeling a video should take into consideration the notion of *temporal correspondence* [43] in order to learn how to represent dynamic scenes and their changes over time.

2.1.2 Self-supervised Learning approaches

Given the problems related to labeling video datasets, a much more practical approach for learning representations consists in using self-supervision in order to leverage large amounts of data without needing human intervention on it. Citing Yann LeCun, Director of Facebook AI Research (FAIR):

> "Most of what we learn as humans and most of what animals learn is in a self-supervised mode, not a reinforcement mode. It's basically observing the world and interacting with it a little bit, mostly by observation in a test-independent way."

Self-supervised learning is a subset of unsupervised learning. Unlike supervised learning, it exploits labels that are naturally part of the input data, or that can be extracted from it, instead of relying on external labels. While self-supervision is,

nowadays, extensively used in Natural Language Processing (NLP), it is actually much less used in computer vision [23, 19].

Self-supervised learning models can be grouped into two categories, *generative* and *discriminative*, which differ statistically on the joint distribution $P(X, Y)$ of the input X and the target Y. Generative models try to model $p(X|Y = y)$ as:

$$p(X|Y = y) = \frac{p(X, Y)}{p(Y = y)} = \frac{p(X, Y)}{\int_x p(Y, X = x)}, \tag{2.1}$$

while discriminative one try to model $p(Y|X = x)$ as:

$$p(Y|X = x) = \frac{p(X, Y)}{p(X = x)} = \frac{p(X, Y)}{\int_y p(X, Y = y)}. \tag{2.2}$$

Generative models have been the center of attention, even for representation learning, for some years thanks to the popularity gained by Generative Adversarial Networks (GANs) [13, 49, 20]. With time, though, the complexity of GANs has lead to an increased interest in discriminative models, which have demonstrated to be able to perform really well in representation learning [18].

When it comes self-supervised learning on videos a common approach consists in solving a *pretext* task which allows the model to learn useful visual features that incorporate temporal consistency [4]. By "useful" features we mean a representation that should be easily adaptable for other tasks, unknown during training time [48]. Subsequently, the learned parameters are transferred to a supervised *downstream* task, such as segmentation or object detection, used to evaluate the quality of features learned by the self-supervised learning process [19].

In the latest years, multiple *pretext* tasks with different self-supervised approaches have been proposed for learning visual representations of videos [11, 24, 45, 27]. Among a lot of different approaches and techniques, *self-supervised contrastive learning* has become one of the most popular, thanks to recent advancements reached with algorithms such as SwAV [6] and SimCLR [7]. In the next section we are going to dive deeper in the concept of self-supervised contrastive learning and how it relates to video representation learning.

2.2 Self-supervised Contrastive Learning

Contrastive learning is a machine learning technique used to learn representations by teaching the model to distinguish similar samples from dissimilar ones, which

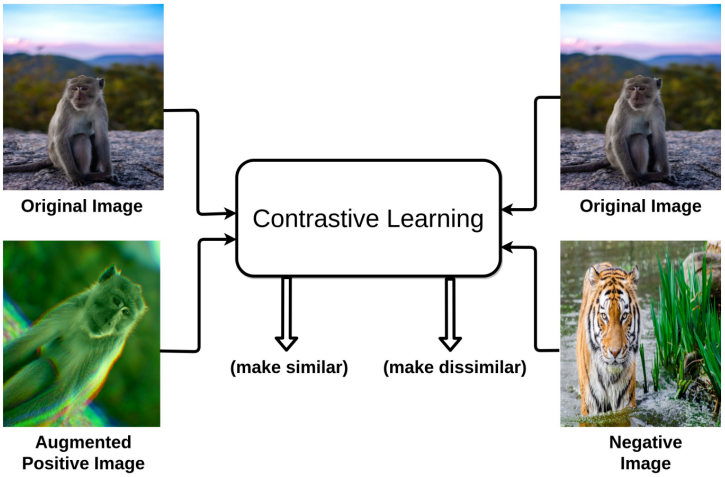


Figure 2.1. Basic functioning of the contrastive learning paradigm [18]

means embedding diverse samples far from each other while similar ones closer. Contrastive learning can be applied in both supervised [22] and unsupervised fashion [18, 41]. In the context of this work the focus is on self-supervised contrastive learning.

In computer vision tasks, the metric used to measure the distance between samples embeddings is a contrastive loss evaluated on the visual representations obtained in output from an encoder. Usually, during training, an augmented version of the original sample is used as *positive* (similar) sample, while all the other samples in the batch are used as *negative* (dissimilar) ones, as shown in Figure 2.1.

When training a model with contrastive learning framework (Figure 2.2) there are four main components needed [7]:

- a *data augmentation* or *label extraction* module which, in the case of videos, extracts query-target pairs that are know to correspond, without human supervision.
- A neural network *base encoder* $f(\cdot)$ that extracts representation vectors from data samples (e.g. ResNet18 or VGG16).
- A pairwise *similarity function* which computes the similarity between the embeddings obtained from the encoder.
- A *contrastive loss function* whose aim is to get similar pairs embeddings close together and diverse ones far from each other.

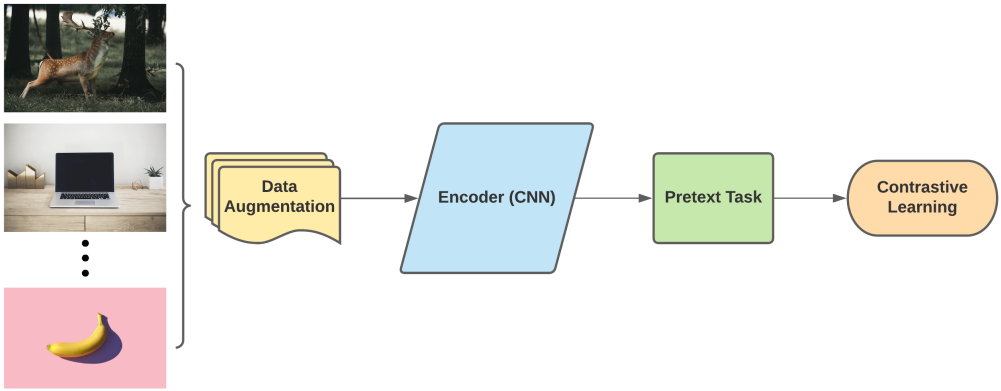


Figure 2.2. Self-supervised contrastive learning training pipeline

A common example of similarity metric is the *inner product*, which applied to two l_2-normalized d-dimensional vectors $\boldsymbol{v}$ and $\boldsymbol{w}$ is defined as $sim(\boldsymbol{v}, \boldsymbol{w}) = \langle \boldsymbol{v}, \boldsymbol{w} \rangle = \boldsymbol{v}^T \boldsymbol{w} / \|\boldsymbol{v}\| \|\boldsymbol{w}\|$ (i.e. cosine similarity). The similarity function is then used as basis for the contrastive loss. The loss function, generally used in recent works [7, 31, 34, 44], for a positive pair of examples (i, j), in a self-supervised framework, is the *normalized temperature-scaled cross entropy* (NT-Xent) loss, defined as follows:

$$\ell_{i,j} = -\log \frac{\exp(\text{sim}(\boldsymbol{z_i}, \boldsymbol{z_j})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq i]} \exp(\text{sim}(\boldsymbol{z_i}, \boldsymbol{z_k})/\tau)} \tag{2.3}$$

where $\mathbb{1}_{[k \neq i]} \in \{0, 1\}$ is an indicator function evaluated 1 iff $k \neq 1$ and τ denotes a temperature parameter. The loss allows the positive pair $(\boldsymbol{z_i}, \boldsymbol{z_j})$ to get closer while repelling the other (dissimilar) items in the batch.
There are also other types of contrastive loss functions, which are used in different self-supervised contexts, such as: *max margin contrastive loss* [14], *triplet loss* [42], *multi-class N-pair loss* [34].

2.3 Cycle-consistency of Time

In their work *"Learning Correspondence from the Cycle-consistency of Time"* [39], Xiaolong Wang et al. introduced the idea of using *cycle-consistency in time* as supervisory signal for learning visual representations from unlabeled video from scratch.

Based on the fact that visual changes in a scene do not happen all of a sudden, there can be made the assumption that there is inherent visual correspondence

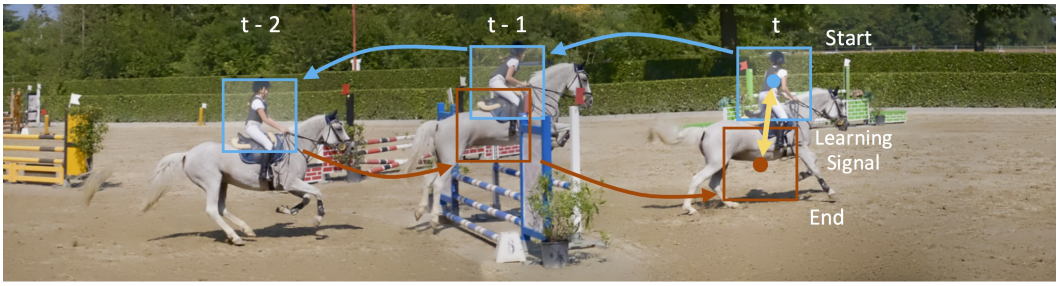


Figure 2.3. A Cycle in Time. Tracking along the sequence formed by a cycle in time where the target is the beginning of the cycle. The yellow arrow between the start and end represents the loss.

between frames adjacent in time, which means that video can be used as a free source of supervision to find this correspondences and learn visual representations. The paper proposes a way of learning to track correspondences and to capture visual invariance, simultaneously.

The key idea of this framework consists in the fact that unlimited supervision can be obtained by tracking correspondences backward and then forward, along a cycle in time (Figure 2.3). The inconsistency between the start and end points is then used as loss function. The aim of the model is to learn visual representations that track correspondences across observations in time in order to minimize the loss (increase *cycle-consistency*).
Finally, shorter cycles may be easier to learn and could compensate for poor correspondence in the early part of training. So, learning happens, simultaneously, from many cycle lengths to improve the learning speed and provide better data.

2.3.1 Training Procedure

During training time (Figure 2.4), the model learns a feature space encoder by ϕ by tracking a patch p_t, extracted from image I_t, backwards and then forwards in time, using a tracker $\mathcal{T}$. The aim of the training is to minimize of the cycle-consistency loss ℓ_θ in order to learn a feature space ϕ for robust visual representations. The cycle-consistency loss ℓ_θ is the euclidean distance between the spatial coordinates of initial patch p_t and the patch found at the end of the cycle in I_t.

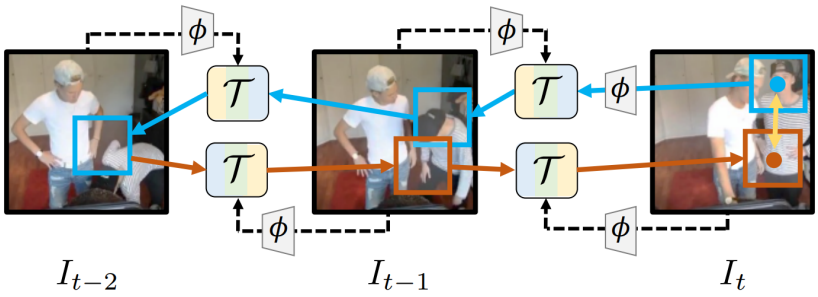


Figure 2.4. Training ϕ by End-to-end Cycle-consistent Tracking

2.4 Superpixel Segmentation

Superpixel segmentation is a category of algorithmic techniques which allow to reduce the number of image primitives for subsequent processing, such as computing local image features. Superpixels are the result of clustering of pixels, which essentially leads to image oversegmentation. They carry more information than single pixels and better represent patterns and objects, or parts of them, inside the image. Due to the lower number of superpixels in an image, usually around the hundreds, with respect to the number of pixels, they allow to improve the computation speed in subsequent processing tasks.

Multiple superpixel segmentation algorithms have been developed in the last couple of decades, but there have been just a few attempts to integrate them into neural networks for various reasons, such as the inefficiency of applying standard convolutions to them [46].

In this section we are going to look at some of the most popular superpixel segmentation algorithms, but before proceeding with the review of the different methods, it's important to define the two macro categories of approaches used in the core of this work: *graph-based* and *gradient-ascent-based*. The following definitions are borrowed from the work of Wang et al., *Superpixel segmentation: A benchmark* [38].

- **Graph-based** methods treat each pixel as a node in a graph where each edge weight corresponds to the similarity between two nodes (pixels). The approach consists in grouping nodes by minimizing a cost function over the graph in order to generate superpixels.

- **Gradient-ascent-based**, or clustering-based, methods group pixels into clusters (i.e., superpixels) and iteratively refine them via gradient ascent methods until some convergence criteria are satisfied.

As evidenced from the previously cited benchmark from Wang et al. [38], graph-based algorithms work better than the clustering-based ones in terms of segmentation accuracy but they require a trade-off between accuracy and speed of segmentation.

2.4.1 Algorithms Review

Here is a review of the different superpixel segmentation algorithms taken into consideration for the development of this work.

SLIC

Ideally, a superpixel method should be fast and produce high quality segmentations. The problem is that most existing superpixel algorithm have high computational cost and/or produce not so good segmentations.
Simple linear iterative clustering (SLIC), proposed in 2010 [2, 3], is a clustering-based algorithm that addresses some of these issues and produces high quality, compact, nearly uniform superpixels more efficiently than most state-of-the-art methods.

Algorithm. It works by first performing a local clustering of pixels in the 5-D space defined by the CIELAB color space and the pixel coordinates, and then using a distance measure to enforce compactness and regularity in the superpixels shapes. In detail, the algorithm performs the following steps:

1. Sample K regularly spaced cluster centers and move them to seed locations based on the lowest gradient position in a $n \times n$ neighborhood.
2. Associate each pixel in the image with the nearest cluster center whose search area overlaps this pixel.
3. For each cluster, compute a new center as the average vector of all the pixels belonging to the cluster.
4. Repeat step 2 and 3 until convergence.

(a) SLIC with 64 components **(b)** SLIC with 256 components

Figure 2.5. An example of superpixel segmentation using SLIC with 64 (a) and 256 (b) components.

The classical *k-mean* algorithm has complexity $O(NKI)$, with N equals the number of pixels in the image, K equals the number of clusters and I corresponds to the number of iterations needed for the algorithm to converge. In comparison, SLIC time complexity is $O(N)$ thanks to the constant number of iterations and to the fact that for each pixel the algorithm needs to compute the distance from a maximum of eight clusters, avoiding a lot of not needed computations.

FH

Felzenszwalb and Huttenlocher (FH), proposed in 2004 [10], is a graph-based algorithm which represents pixels as nodes on a graph where each edge weight corresponds to the dissimilarity between two pixels. Superpixels are then generated by finding the minimum spanning tree of the constituent pixels. The method uses the Dijkstra algorithm to compute the shortest paths in the undirected graph.

(a) FH with scale 500 (59 superpixels) **(b)** FH with scale 100 (256 superpixels)

Figure 2.6. An example of superpixel segmentation using FH with scale set to 500 (a) and 100 (b).

The superpixels computed by the FH algorithm adhere well to the boundaries

of the objects inside an image but, at the same time, they have slightly irregular shapes and sizes (see Figure 2.6). Another problem of this method is that it finds the superpixels in a greedy way, so its computational complexity, $O(N\log N)$, is much higher than other algorithms. Finally, something to consider is the fact that the number of superpixels cannot be controlled but it depends on the scale and minimum size chosen at initialization.

QuickShift

QuickShift, proposed in 2008 [37], is a clustering-based mode-seeking segmentation method. It uses the Parzen density estimate to form a tree of links from each point to the nearest neighbor which increases the density. It is able to cluster N points in $O(dN^2)$, with d being a small constant only if the distance used is Euclidean. The algorithm is a considerable improvement over mean shift which allows it to be much faster in clustering points, but still slower than most other state-of-the-art methods.

(a) QuickShift with ratio 0.5, kernel size 10 and max distance 100 (44 superpixels)

(b) QuickShift with ratio 0.5, kernel size 10 and max distance 10 (162 superpixels)

Figure 2.7. An example of superpixel segmentation using QuickShift with max distance set to 100 (a) and 10 (b).

LSC

Linear Spectral Clustering (LSC), proposed in 2015 [26], is a gradient-ascent-based algorithm which produces compact and uniform superpixels with low computational cost, $O(N)$. It maps pixel values and coordinates inside the image to a high dimensional feature space through a kernel function. The superpixel segmentation is based on an similarity metric (approximated by the kernel function) which measures

color similarity and space proximity between pixels. Thanks to its properties, the LSC algorithm is able to quickly segment an image into superpixels while being memory efficient. The superpixels produced preserve global features of the image and are usually more regular compared to SLIC.

(a) LSC with region size 100 and ratio 0.5 (32 superpixels)

(b) LSC with region size 50 and ratio 0.5 (144 superpixels)

Figure 2.8. An example of superpixel segmentation using LSC with region size set to 100 (a) and 50 (b).

Chapter 3

Self-supervised Representation Learning via Contrastive Walks in Videos

This chapter presents problem formalization, related work and implementation of the model developed for this thesis. In the first section, I describe what are the problems solved by previous research works and I provide the reasons why this work has been proposed. In the second section, a brief description of the work from Jabri et al. [17] is presented in order to provide a basis for my work. Finally, the proposed transition from Patch-based to Object-based Contrastive Walks is covered in an extensive way and further insights on the full architecture and the training procedure are also provided.

3.1 Problem Formalization

As already stated in Section 2.1, self-supervised representation learning from video is still catching up to all the improvements obtained in learning from images. The fundamental problem is that video cannot be treated as a basic sequence of frames with no temporal relation between each other. In most works proposed in recent times, the time variable (T) in video is considered just another dimension which allows to relate just a specific pixel in position (x, y) at time t to the same pixel at time $t + k$. While this approach could make sense in a *high sampling rate*

scenario, in practice the same pixels might have no relation what so ever between each other when going from frame t to $t + k$ due to the low sampling rate and to the movement of objects and camera in a dynamic environment.

The vast majority of self-supervised representation learning models developed in recent times learn visual representations through similarity learning on matching image pairs, where the ground-truth is known and the matching view is the corresponding augmented sample. But when we work with temporal correspondences we have to consider the fact that they are *latent*, which means that they do not manifest explicitly in the data but we have to learn them. The problem is that in order to learn temporal correspondences we rely on the model, yet the model needs temporal correspondences to be trained. The solution to this problem has been proposed by Jabri et at. and it is described in Section 3.2. However, the patch-based approach used in Jabri's work can be quite computationally expensive, plus it needs longer training times in order to converge. For this reason, in Section 3.3 I propose a novel object-based approach which leverages superpixels in order to reduce the complexity of the training process. Objects inside a video are fewer than the number of overlapping patches, so they allow to use much larger batch sizes and lead to a faster convergence of the loss.

To conclude, the final goal behind this work is to propose a novel self-supervised approach to learn a generic video-encoder by leveraging the success of recent self-supervised methods (e.g. Videowalk [17], SimCLR [7]). The aim is to generate a version of the model pretrained on huge amounts of data (e.g. BERT-like [8]) which would then allow to transfer knowledge to other downstream tasks such as instance segmentation and object tracking.

3.2 Patch-based Contrastive Walks in Video

In the paper *Space-Time Correspondence as a Contrastive Random Walk*, by A. Jabri et al. (NeurIPS 2020) [17], the authors propose a new self-supervised approach for learning a representation for visual correspondence from raw video. The idea consists in generating a space-time graph, from video, where image patches are nodes and edges exists only between nodes from adjacent frames. The weight on the edge connecting two nodes corresponds to the similarity between the patches, determined under a learned representation. So, in practice, learning a representation consists in

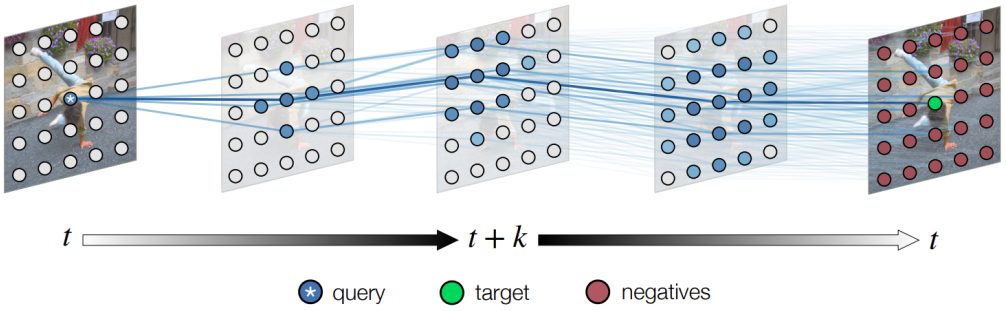


Figure 3.1. Illustration of a random walk though the space-time graph.

fitting the transition probability matrix of a random walker stepping from patch to patch through time along the graph.

The self-supervised approach exploited in this work is based on the idea of *cycle-consistency of time* [39], reviewed in Section 2.3. During training, self-supervision is introduced by generating *palindrome* sequences of frames from each training video, which allow to have a query and a target node that actually correspond to the same patch in the same frame. In this way, each step of the walker going from the starting to the ending point of the sequence can be interpreted as a contrastive learning problem (Figure 3.1).

3.2.1 The Contrastive Learning Framework

Each video is composed by a set of frames $\mathbf{I}$. Each frame $\boldsymbol{I}_t$ is divided into a set $\boldsymbol{q}_t$ of overlapping patches which represent the N nodes of the directed graph. The set of nodes for each frame is encoded by an encoder ϕ into l_2-normalized d-dimensional vectors which together make up an embedding matrix $Q_t \in \mathbb{R}^{N \times d}$. The similarity between two vectors q_1 and q_2 is computed through a pairwise similarity function $d_\phi(q_1, q_2) = \langle \phi(q_1), \phi(q_2) \rangle$. To each pairwise similarity (edge going from a node in frame t to another in frame $t+1$) is applied a softmax function with temperature τ to generate non-negative affinities in order to obtain a stochastic matrix of affinities:

$$A_t^{t+1}(i,j) = \text{softmax}(Q_t Q_{t+1}^T)_{ij} = \frac{\exp(d_\phi(\boldsymbol{q}_t^i, \boldsymbol{q}_{t+1}^j)/\tau)}{\sum_{l=1}^{N} \exp(d_\phi(\boldsymbol{q}_t^i, \boldsymbol{q}_{t+1}^l)/\tau)} \tag{3.1}$$

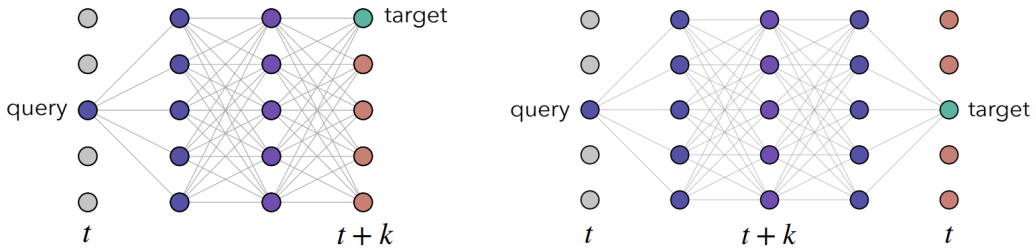


Figure 3.2. Walk on the space-time graph. In (a) the walker goes from the query node to the ideally known target. In (b) a palindrome sequence is used to go from query to target.

The long-range affinity matrix for the whole graph can be formulated as a multiple steps walk in time along the graph (Figure 3.2a):

$$\bar{A}_t^{t+k} = \prod_{i=0}^{k-1} A_{t+i}^{t+i+1} = P(X_{t+k}|X_t), \tag{3.2}$$

where X_t is the state of the walker at time t.

In the ideal scenario, we could suppose to have a ground-truth for the target reached by the walker. In such case, supervision could be exploited to train the encoder by maximizing the likelihood of the walker beginning at a query node and ending up at the target corresponding to the positive label. Computing the supervised loss would correspond to computing the cross entropy loss $\mathcal{L}_{CE}$ between the affinity matrix at time $t+k$ and the corresponding labels Y_t^{t+k}:

$$\mathcal{L}_{sup} = \mathcal{L}_{CE}(\bar{A}_t^{t+k}, Y_t^{t+k}) = -\sum_{i=1}^{N} \log P(X_{t+k} = Y_t^{t+k}(i)|X_t = i) \tag{3.3}$$

By minimizing $\mathcal{L}_{sup}$ we increase the probability of the walker following a path that links query and target nodes.

Self-supervision in training is achieved through the generation of *palindrome* sequences (Figure 3.2b), which means that the second half of the sequence is identical to the first half reversed. So the sequence $(I_t, ..., I_{t+k})$ would become $(I_t, ..., I_{t+k}, ..., I_t)$. In practice, for each query node exists a corresponding target node; this idea leads to the use of the following cycle-consistency objective:

$$\mathcal{L}_{cyc}^k = \mathcal{L}_{CE}(\bar{A}_t^{t+k}\bar{A}_{t+k}^t, I) = -\sum_{i=1}^{N} \log P(X_{t+2k} = i|X_t = i). \tag{3.4}$$

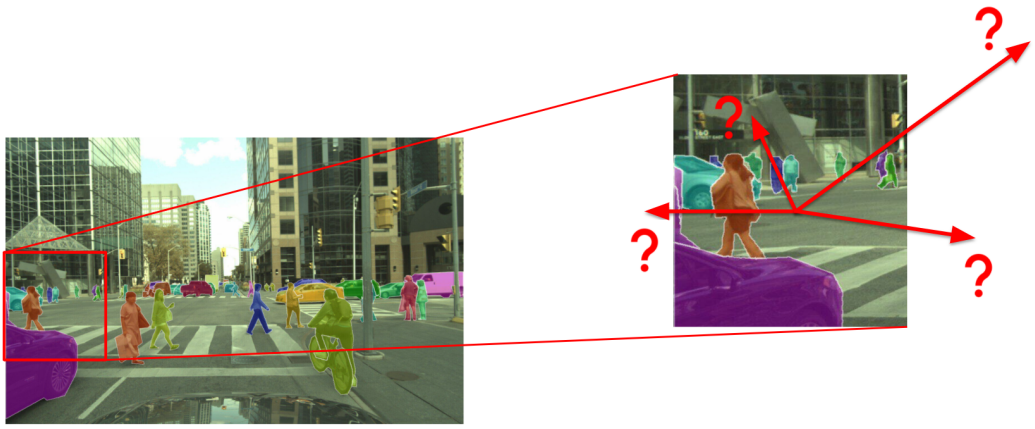


Figure 3.3. In a single square patch there could be multiple objects moving simultaneously in different directions. Such scenario could cause ambiguity in the transition probabilities of the adjacency matrix.

3.3 Novel Object-based Contrastive Walks in Video

3.3.1 From Patches to Objects

The subdivision of a frame into multiple square patches might lead to some problems which would not allow to train the best possible encoder.

- A single patch could contain more than one object. This could create ambiguity in the adjacency matrix (Figure 3.3).
- Different objects in the scene, and usually in a single patch, move in different directions.
- The background is included in (almost) every patch, which again leads to ambiguity when comparing the encoding of two different patches.
- The computational cost of working with a lot of patches is higher than working with just a few objects and the background.

The approach proposed in this thesis consists in substituting the subdivision of each frame into a set of patches to the subdivision of each frame into segmented objects, which are usually less in number and better defined than square patches. Such approach would bring various advantages:

- lower computational cost thanks to the smaller number of objects in a frame, as demonstrated by Henaff et al. [16], when compared to the number of patches needed to train the model;
- larger batch sizes can be used to speed up the learning process and make it converge faster;
- tracking moving objects in the scene avoids ambiguity in the adjacency matrix;
- distinguishing background and foreground solves the problem of camera movement;
- working with objects instead of patches improves the interpretability.

3.3.2 Superpixel Extraction and Embedding

With this thesis, I propose a novel approach to self-supervised contrastive learning in video which uses as basis the work of Jabri et al., reviewed in Section 3.2, but introduces the idea of extracting objects instead of square patches from each frame of a video. Objects from an image are extracted through *superpixel segmentation*, discussed in Section 2.4, and they are used as substitute to patches to learn the transition probability matrix.

In the patch-based approach each $H \times W$ patch is embedded using a ResNet-18 [15] into a $h \times w \times 512$ feature map, which after average pooling becomes a 512-dimensional vector. Subsequently, l_2-normalization is applied to it and the vector is linearly projected into a 128-dimensional space. In practice, q_t^i into Equation 3.1 corresponds to the 128-dimensional vector associated to node i in frame t.

When working with superpixels instead of patches, the operations needed for the embedding of each superpixel change. The following steps are executed:

1. A full size frame, with dimensions $H \times W$, is encoded using the ResNet-18 which produces a $h \times w \times 512$ feature map.
2. N superpixels are computed from the full $H \times W$ frame. Each superpixel corresponds to a $H \times W$ matrix with 0/1 values.
3. For each feature of the feature map the receptive field is computed.

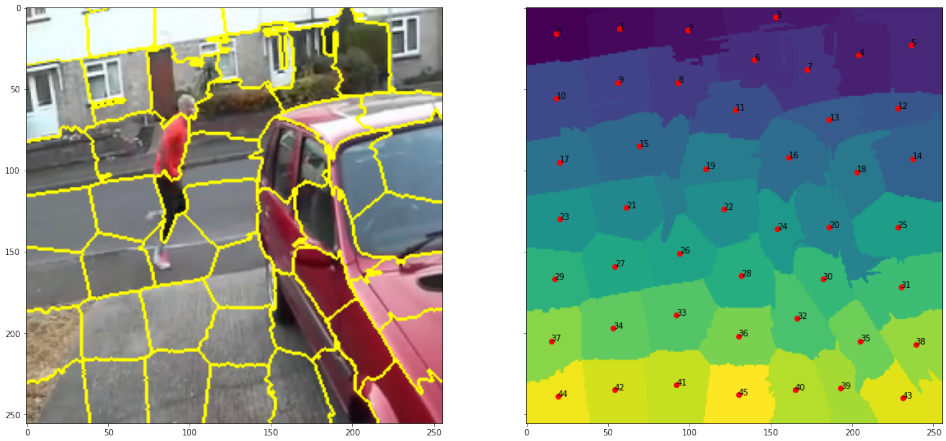


Figure 3.4. Superpixels computed from a frame using the SLIC algorithm

4. Let's call the current superpixel s, where $s \in \{s_1, s_2, ..., s_N\}$. Let's also call r_{ij} the receptive field associated to the feature f_{ij}, where $f \in \mathbb{R}^{h \times w \times 512}$ is the feature map and $f_{ij} \in \mathbb{R}^{512}$ is the feature with coordinates (i, j). In order to compute the vector q_t^s associated to superpixel s we need to weight and sum all the features according to the proportion of overlap of the corresponding receptive field with respect to s. Then, the embedding vector of s in frame t is defined as

$$q_t^s = \sum_{i=1}^{h} \sum_{j=1}^{w} \frac{r_{ij} \cap s}{|s|} f_{ij} \tag{3.5}$$

   The embedding matrix for q_t is denoted as $Q_t \in \mathbb{R}^{N \times 512}$

5. Finally, each 512-dimensional vector is linearly projected into a smaller 128-dimensional one and l_2-normalized.

3.3.3 Implementation

The end-to-end training of the proposed architecture is made up of some fundamental steps which are going to be described as follows.

Superpixel Masks Generation. A superpixel segmentation algorithm is used to extract, from each frame of a video, a variable number of superpixel masks (Figure 3.4), depending on the hyper-parameters of the algorithm.

Encoding. Each superpixel mask is embedded using a CNN, specifically a ResNet-18 [15], which gives in output a 512-dimensional vector, which is $l2$-normalized

immediately afterwards. Each frame will now have shape $(S, 512)$, where S is the number of superpixel masks (one for each superpixel) and is variable from frame to frame. After the generation and embedding of all the masks, *constant padding* is applied to each one of them in order to be able to concatenate multiple frames, with different number of superpixels, into the same video-tensor. The padding value is define as the maximum number of superpixels (M) in a single frame among all the videos in the batch minus the actual number of superpixels found in the current frame. This allows to obtain a batch-tensor with shape $(B, T, M, 512)$, where B is the batch size and T in the clip length. The tensor is then linearly projected into a new one with shape $(B, T, M, 128)$ and l_2-normalized.

Affinity Matrices Computation. During training, different path lengths are used for the walk in order to compute the affinity matrices and, consequently, the loss. Specifically, assuming clip length T, the loss is computed as the sum of the losses across different sub-cycles: $\mathcal{L}_{train} = \sum_{i=1}^{T} \mathcal{L}_{cyc}^{i}$. A sub-cycle is a palindrome sub-sequence of frames from the whole video clip.

Encoder Architecture

The ResNet-18 used for the encoding of input video clips is a modified version of the original one [15], the same used by Jabri et. al in their work. Specifically, the stride of the last two convolutional blocks is reduced from 2 to 1 in order to increase the size of the output by a factor of 4. To obtain as output the full feature map, the average pooling, fully connected and softmax layer are removed (Table 3.1).

Table 3.1. Modified ResNet-18 Architecture

Layer Name	Output Size	ResNet-18
input	$H \times W \times 3$	
conv1	$H/2 \times W/2 \times 64$	$7 \times 7, 64, \text{stride2}$
maxpool	$H/4 \times W/4 \times 64$	stride 2
res1	$H/4 \times W/4 \times 64$	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$, stride 1
res2	$H/8 \times W/8 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$, stride 2
res3	$H/8 \times W/8 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$, stride 1
res4	$H/8 \times W/8 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$, stride 1

Chapter 4

Experiments and Results

4.1 Datasets and Benchmark

4.1.1 Kinetics 400

The model has been trained on the complete *Kinetics 400* dataset. A description of it follows.

Kinetics 400 [21] is a collection of large-scale, high-quality datasets of URL links of up to 306,000 video clips that cover 400 human action classes. The videos include human-object interactions such as playing instruments, as well as human-human interactions such as shaking hands and hugging. Each action class has at 400–1150 clips video clips, each from a unique video. Each clip is human annotated with a single action class and lasts around 10 seconds. For the purpose of this project annotations have not been used.

At training time a batch of video clips is extracted from the dataset and each clip is processed by a *transform* function which performs the following actions:

- *Resize*: resize each frame to 256×256.
- *Normalization*: normalize pixel values.
- *Color Jitter*: randomly change the brightness, contrast, saturation and hue of the video.
- *Random Horizontal Flip*: horizontally flip the given video randomly with a

given probability.

4.1.2 DAVIS 2017

The model has been evaluated on the DAVIS 2017 [30] benchmark. A description of it follows.

The Densely-Annotated VIdeo Segmentation **(DAVIS) 2017** is an initiative that provides a dataset and benchmark for video object segmentation. The dataset contains a total of 150 high-definition sequences with all their frames annotated with multi-object (2-4) masks at pixel-level accuracy. In each frame of each video, the main moving objects in the scene are segmented and divided by semantics (e.g. people, animals, cars). For the purpose of our evaluation, the *val* dataset has been used. It contains 30 segmented videos at 480p resolution.

The evaluation of the model on DAVIS 2017 reports, as performance metrics, the mean and recall over all object instances of the $\mathcal{J}$ and $\mathcal{F}$ measures, already proposed in DAVIS 2016 [29]:

- **Region Similarity** $\mathcal{J}$. It measures the region-based segmentation similarity, i.e. Jaccard index defined as the *intersection-over-union* of the estimated segmentation mask M and the goundtruth mask G.

$$\mathcal{J} = \frac{|M \bigcap G|}{|M \bigcup G|}$$

- **Contour Accuracy** $\mathcal{F}$. Defining $c(M)$ as the set of closed-contours delimiting the mask, we can compute the contour-based precision P_c and recall R_c between the contour point of $c(M)$ and $c(G)$. The $\mathcal{F}$ is a trade-off measure between P_c and R_c.

$$\mathcal{F} = \frac{2P_cR_c}{P_c + R_c}$$

4.1.3 Superpixel Segmentation Benchmark

In order to choose one or more superpixel segmentation methods to be integrated inside the model, a custom benchmark has been developed for the comparison of different methods. The metrics used are the same used in the DAVIS benchmark, which means **Region Similarity** $\mathcal{J}$ and **Contour Accuracy** $\mathcal{F}$. Mean and recall of

Table 4.1. SLIC segmentation evaluation

# of components	compactness	$\mathcal{J}, \mathcal{F}, \mathcal{J}\&\mathcal{F}$ mean	$\mathcal{J}, \mathcal{F}$ recall
5	30	0.131402 0.123832 0.127617	0.0 0.0
5	50	0.143864 0.139346 0.141605	0.0 0.033333
10	30	0.161979 0.139603 0.150791	0.066667 0.0
10	50	0.151404 0.134106 0.142755	0.1 0.0
20	30	0.099141 0.092297 0.095719	0.033333 0.0
20	50	0.100956 0.095027 0.097992	0.0 0.0

the $\mathcal{J}$ and $\mathcal{F}$ measures over all object instances are reported for each segmentation method and for each combination of hyper-parameters.

Tables 4.1, 4.2 and 4.3 illustrate the results obtained from the evaluation of each one of the methods described in Section 2.4 with different hyper-parameters tested:

Due to its high computational complexity, the QuickShift algorithm couldn't have been evaluated.

Table 4.2. FH segmentation evaluation

scale	**sigma**	**min size**	$\mathcal{J}, \mathcal{F}$, $\mathcal{J}\&\mathcal{F}$ **mean**	$\mathcal{J}, \mathcal{F}$ **recall**
2000	0.8	2000	0.206562 0.210053 0.208307	0.300000 0.233333
5000	0.5	1000	0.232154 0.271940 0.252047	0.3 0.5
5000	0.5	2000	0.234891 0.266252 0.250571	0.266667 0.433333
5000	0.8	1000	0.226065 0.276450 0.251258	0.300000 0.466667
5000	0.8	2000	0.227663 0.267074 0.247369	0.300000 0.433333

Table 4.3. LSC segmentation evaluation

region size	**ratio**	$\mathcal{J}, \mathcal{F}, \mathcal{J}\&\mathcal{F}$ **mean**	$\mathcal{J}, \mathcal{F}$ **recall**
100	0.2	0.041184 0.062297 0.051740	0.000000 0.033333
100	0.5	0.033785 0.042699 0.038242	0.0 0.0
300	0.2	0.112628 0.120868 0.116748	0.033333 0.066667
300	0.5	0.128415 0.149314 0.138865	0.033333 0.066667

4.2 Training of the Model

Due to the large amount of time needed for the training of the model, between 9 and 22 hours per epoch depending on the configuration, I decided to perform multiple experiments, with different hyper-parameters, for just one single epoch which, according to the training of the model by A. Jabri et al. [17], named *videowalk* from now on, should allow the model to reach at least 80-85% of the performance it would reach after 8-10 epochs.

The training of the model has been performed with different batch sizes (between 16 and 52), for 1 epoch per configuration, on two Nvidia Tesla V100-SXM2-32GB.

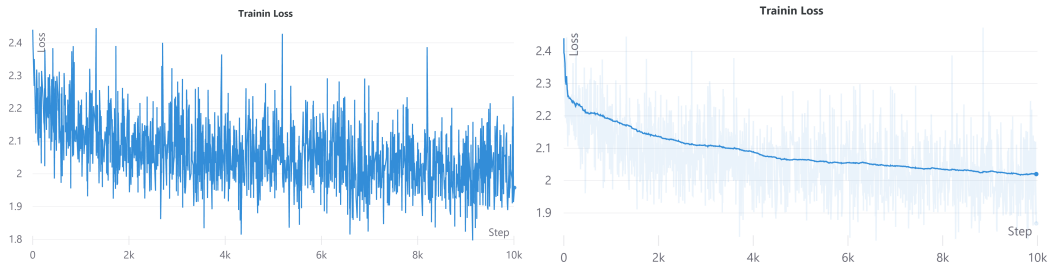
Table 4.4 shows the hyper-parameters used for the training of the model.

Table 4.4. Training hyper-parameters

Hyper-parameter	**Value**
Learning rate	0.0001, 0.0003
Temperature τ	0.05
Frame size	256
Clip length	4
Edge dropout	0.1

The training of the baseline model, *videowalk*, has been executed with a batch size of 36, due to the high memory requirements of the patches. Additionally, 4 different experiments have been performed with the following superpixel segmentation methods and respective hyper-parameters:

- SLIC algorithm with number of components = 10 and compactness = 30 (Figure 4.2);
- SLIC algorithm with number of components = 30 and compactness = 30 (Figure 4.3);
- SLIC algorithm with number of components = 50 and compactness = 30 (Figure 4.4);
- SLIC and FH algorithms with uniform random sampling (Figure 4.5) of the method for each video clip (i.e., clip 1 with SLIC, clip 2 with FH, etc.) where the hyper-parameters are the following:



(a) Loss not smoothed **(b)** Exponential moving average

Figure 4.1. Training loss of *videowalk* model

Figure 4.2. Training loss of model with SLIC and number of components = 10

  - SLIC: number of components = 30 and compactness = 30;
  - FH: scale = 600, sigma = 0.5, minimum size = 400.

- Patches and SLIC segmentation (Figure 4.6) with random sampling with probability 0.7 for patches and 0.3 for superpixels. SLIC with number of components = 30 and compactness = 30.

The main advantages of the object-based approach over the patch-based one is, as already explained, the lower computational cost. In fact, as we can see from Table 4.5, the training runs which used superpixels allowed to exploit a batch size at least 40% larger and completed the epoch in up to 40% less time.

Figure 4.3. Training loss of model with SLIC and number of components = 30

Figure 4.4. Training loss of model with SLIC and number of components = 50

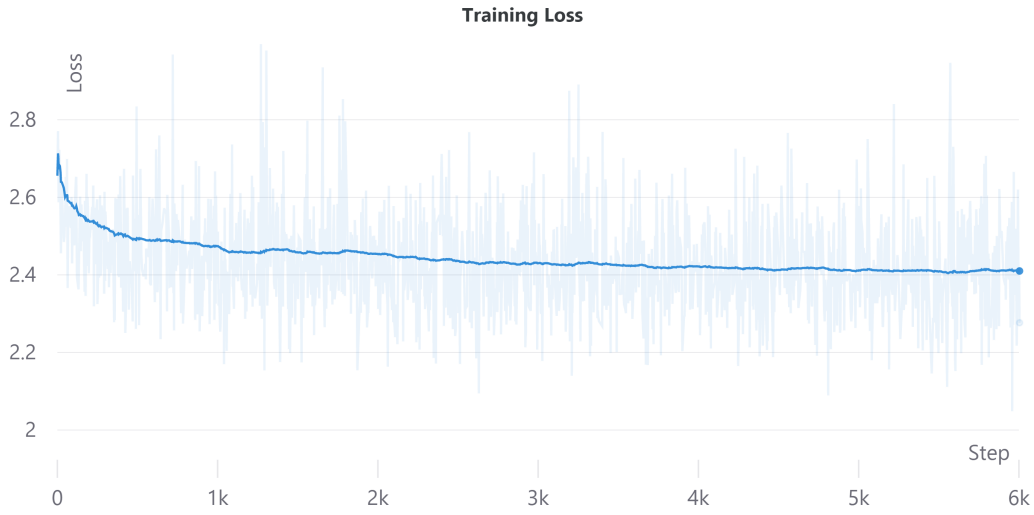


Figure 4.5. Training loss of model with uniform random selection of the method between SLIC and FH

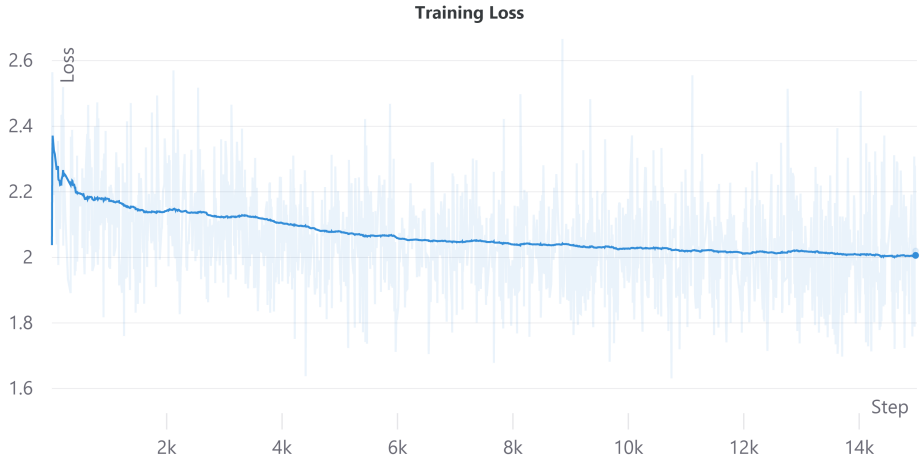


Figure 4.6. Training loss of model with Patches and SLIC segmentation with number of components = 30

Table 4.5. Batch sizes and Training times of training run with different configurations

Method	Batch Size	Training Time h:m
Videowalk	36	16:27
Ours w/ SLIC-10	52	9:14
Ours w/ SLIC-30	52	13:22
Ours w/ SLIC-50	52	14:59
Ours w/ SLIC & FH	52	16:40
Ours w/ patches & SLIC	16	10:52

4.3 Evaluation and Results

The evaluation task consists in performing label propagation on videos given just the label for the first frame. Label propagation means predicting labels for each pixel of each target frame. The representations obtained using the trained encoder are used as a similarity function for the k-nearest neighbors prediction, as already implemented in previous works [25, 39]. The actual implementation of the evaluation algorithm is borrowed from the *videowalk* [17] work.

Table 4.6 shows the hyper-parameters used for the evaluation of the model on the DAVIS benchmark. To allow the comparison with the *videowalk* model I used the same hyper-parameters used in the paper by Jabri et al.

Table 4.6. Evaluation hyper-parameters

Hyper-parameter	**Value**
Temperature τ	0.05
# of neighbors k	10
# of context frames	20
Spatial radius	12

Following the DAVIS guidelines, I evaluated the model on the task of semi-supervised multi-object (i.e. 2-4) segmentation. The images used for the evaluation have resolution 480p. For each model configuration I report mean (m) and recall (r) of standard boundary alignment ($\mathcal{F}$) and region similarity ($\mathcal{J}$) metrics.

Performance of the model, in different configurations, on DAVIS 2017 metrics

Table 4.7. Video object segmentation results on DAVIS 2017 val set Comparison of our method (4 variants) with previous self-supervised approach Videowalk as baseline. $\mathcal{J}$ measures region similarity, $\mathcal{F}$ is a boundary alignment metric.

Method	$\mathcal{J}\&\mathcal{F}_m$	$\mathcal{J}_m$	$\mathcal{J}_r$	$\mathcal{F}_m$	$\mathcal{F}_r$
Videowalk	0.576946	0.566783	0.675131	0.587109	0.662787
Ours w/ SLIC-10	0.14492	0.110819	0.051421	0.179021	0.054988
Ours w/ SLIC-30	0.305837	0.283757	0.286523	0.327916	0.281627
Ours w/ SLIC-50	0.233967	0.203531	0.145167	0.264402	0.190767
Ours w/ SLIC-30 & FH	0.363672	0.354097	0.324767	0.373248	0.328937
Ours w/ patches & SLIC-30	0.552	0.541	0.609	0.564	0.618

are reported in Table 4.7 and compared with *Videowalk* performance.

As shown from the results in Table 4.7, our model is able to perform almost on par with the original one, videowalk, on the DAVIS 2017 benchmark, and it requires a much lower training duration. The configuration that gives the best results overall is the one using both patches and superpixels computed with SLIC with 30 components. The combination of the two approaches gives very good results in terms of $\mathcal{J}$ and $\mathcal{F}$, just 0.02 below *videowalk*, while taking more than 30% less time to complete 1 epoch with a lower batch size.
When training models using just superpixels the training time is again lower compared to videowalk. But, in those cases, we can deduce from the plots of the training losses that the models using just SLIC segmentation (Figures 4.2, 4.3, 4.4) quickly overfit the training dataset. This behavior leads to a lower loss which flattens before the end of the epoch and stays the same even during the rest of the training. On the other side, the model using SLIC + FH segmentation (Figure 4.5) is not able to learn very well, as we can see from the loss decreasing too slowly. One possible solution to this problem could be the increase of selection probability of the SLIC segmentation method which allows the model to learn faster, while keeping the FH segmentation method to introduce some noise into the learning process.

Chapter 5

Conclusions and Future Work

In this work I proposed a novel approach for learning visual representations from raw video in a self-supervised way by tracking segmented objects via a contrastive random walk. The object extraction is performed via superpixel segmentation methods which allow to keep the whole training process self-supervised and to not introduce labeled data. The aim of this novel object-based approach is to substitute, or complement, the patch-based one, previously proposed by Jabri et al. [17].

I successfully integrated objects by means of superpixels into the model of Jabri et al. and achieved the expected improvements in training and convergence speed. However, the new model is still slightly under-performing the original one but, being this an early research stage on the topic, the results seem promising and lots of possible future variations of the model are yet to be explored.

Future developments of the current work include various possibilities.

Firstly, I aim to continue the analysis on the combination of patches and superpixels for the training of the model in order to find the best configuration to balance training speed, convergence speed, and performance.

Moreover, the use of objects inside the graph could allow us to model the interaction between them with the use of Graph Convolutional Networks. One option could be to adapt the recently developed STS-GCN [33] to the object-based space-time graph architecture to suit the process of learning visual representations.

Finally, research has demonstrated that Transformer Networks are a very good choice when it comes to modeling long-term dependencies. The use of Transformers

in Natural Language Processing has lead to state-of-the-art results in different benchmarks [12, 35, 36]. At the same time, Transformers have also shown promising results in multiple computer vision tasks and benchmarks [5, 40, 28, 9], sometimes matching or outperforming other state-of-the-art models. Thus, one possible future development could consists in the use of Transformers for modeling long-range temporal correspondences in our space-time graph structure.

The research presented in this thesis, and its future developments, have the potential to positively contribute in practical applications such as autonomous driving systems, people/objects tracking in surveillance for authorized purposes, motion tracking in sports, and many others.

Bibliography

[1] URL: https://paperswithcode.com/sota/image-classification-on-imagenet.

[2] Radhakrishna Achanta et al. "SLIC superpixels". In: *Technical report, EPFL* (June 2010).

[3] Radhakrishna Achanta et al. "SLIC Superpixels Compared to State-of-the-Art Superpixel Methods". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2274–2282. DOI: 10.1109/TPAMI.2012.120.

[4] Nikolas Adaloglou. "Self-supervised representation learning on videos". In: *https://theaisummer.com/* (2020). URL: https://theaisummer.com/self-supervised-learning-videos/.

[5] Nicolas Carion et al. *End-to-End Object Detection with Transformers*. 2020. arXiv: 2005.12872 [cs.CV].

[6] Mathilde Caron et al. *Unsupervised Learning of Visual Features by Contrasting Cluster Assignments*. 2021. arXiv: 2006.09882 [cs.CV].

[7] Ting Chen et al. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020. arXiv: 2002.05709 [cs.LG].

[8] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. arXiv: 1810.04805 [cs.CL].

[9] Alexey Dosovitskiy et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. arXiv: 2010.11929 [cs.CV].

[10] Pedro F. Felzenszwalb and D. Huttenlocher. "Efficient Graph-Based Image Segmentation". In: *International Journal of Computer Vision* 59 (2004), pp. 167–181.

[11] Basura Fernando et al. *Self-Supervised Video Representation Learning With Odd-One-Out Networks*. 2017. arXiv: 1611.06646 [cs.CV].

[12] *GLUE Benchmark*. URL: https://gluebenchmark.com/leaderboard.

[13] Ian J. Goodfellow et al. *Generative Adversarial Networks*. 2014. arXiv: 1406.2661 [stat.ML].

[14] R. Hadsell, S. Chopra, and Y. LeCun. "Dimensionality Reduction by Learning an Invariant Mapping". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. Vol. 2. 2006, pp. 1735–1742. DOI: 10.1109/CVPR.2006.100.

[15] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[16] Olivier J. Hénaff et al. *Data-Efficient Image Recognition with Contrastive Predictive Coding*. 2020. arXiv: 1905.09272 [cs.CV].

[17] Allan Jabri, Andrew Owens, and Alexei A. Efros. *Space-Time Correspondence as a Contrastive Random Walk*. 2020. arXiv: 2006.14613 [cs.CV].

[18] Ashish Jaiswal et al. "A Survey on Contrastive Self-Supervised Learning". In: *Technologies* 9.1 (2021). ISSN: 2227-7080. DOI: 10.3390/technologies9010002. URL: https://www.mdpi.com/2227-7080/9/1/2.

[19] Longlong Jing and Yingli Tian. *Self-supervised Visual Feature Learning with Deep Neural Networks: A Survey*. 2019. arXiv: 1902.06162 [cs.CV].

[20] Tero Karras, Samuli Laine, and Timo Aila. *A Style-Based Generator Architecture for Generative Adversarial Networks*. 2019. arXiv: 1812.04948 [cs.NE].

[21] Will Kay et al. *The Kinetics Human Action Video Dataset*. 2017. arXiv: 1705.06950 [cs.CV]. URL: https://deepmind.com/research/open-source/kinetics.

[22] Prannay Khosla et al. *Supervised Contrastive Learning*. 2021. arXiv: 2004.11362 [cs.LG].

[23] Alexander Kolesnikov, Xiaohua Zhai, and Lucas Beyer. *Revisiting Self-Supervised Visual Representation Learning*. 2019. arXiv: 1901.09005 [cs.CV].

[24] Hsin-Ying Lee et al. *Unsupervised Representation Learning by Sorting Sequences*. 2017. arXiv: 1708.01246 [cs.CV].

[25] Xueting Li et al. *Joint-task Self-supervised Learning for Temporal Correspondence*. Sept. 2019.

[26] Zhengqin Li and Jiansheng Chen. "Superpixel segmentation using Linear Spectral Clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 1356–1363. DOI: 10.1109/CVPR.2015.7298741.

[27] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. *Shuffle and Learn: Unsupervised Learning using Temporal Order Verification*. 2016. arXiv: 1603.08561 [cs.CV].

[28] Niki Parmar et al. *Image Transformer*. 2018. arXiv: 1802.05751 [cs.CV].

[29] F. Perazzi et al. "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation". In: *Computer Vision and Pattern Recognition*. 2016.

[30] Jordi Pont-Tuset et al. "The 2017 DAVIS Challenge on Video Object Segmentation". In: *arXiv:1704.00675* (2017).

[31] Rui Qian et al. *Spatiotemporal Contrastive Video Representation Learning*. 2021. arXiv: 2008.03800 [cs.CV].

[32] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[33] Theodoros Sofianos et al. *Space-Time-Separable Graph Convolutional Network for Pose Forecasting*. 2021. arXiv: 2110.04573 [cs.CV].

[34] Kihyuk Sohn. "Improved Deep Metric Learning with Multi-class N-pair Loss Objective". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf.

[35] *SQuAD Explorer*. URL: https://rajpurkar.github.io/SQuAD-explorer/.

[36] *SWAG Benchmark*. URL: https://leaderboard.allenai.org/swag/submissions/public.

[37] Andrea Vedaldi and Stefano Soatto. "Quick Shift and Kernel Methods for Mode Seeking". In: *Computer Vision – ECCV 2008*. Ed. by David Forsyth, Philip Torr, and Andrew Zisserman. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 705–718. ISBN: 978-3-540-88693-8.

[38] Murong Wang et al. "Superpixel segmentation: A benchmark". In: *Signal Processing: Image Communication* 56 (2017), pp. 28–39. ISSN: 0923-5965. DOI: https://doi.org/10.1016/j.image.2017.04.007. URL: https://www.sciencedirect.com/science/article/pii/S0923596517300735.

[39] Xiaolong Wang, Allan Jabri, and Alexei A. Efros. *Learning Correspondence from the Cycle-Consistency of Time*. 2019. arXiv: 1903.07593 [cs.CV].

[40] Xiaolong Wang et al. *Non-local Neural Networks*. 2018. arXiv: 1711.07971 [cs.CV].

[41] Xinlong Wang et al. "Dense Contrastive Learning for Self-Supervised Visual Pre-Training". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2021, pp. 3024–3033.

[42] Kilian Q Weinberger, John Blitzer, and Lawrence Saul. "Distance Metric Learning for Large Margin Nearest Neighbor Classification". In: *Advances in Neural Information Processing Systems*. Ed. by Y. Weiss, B. Schölkopf, and J. Platt. Vol. 18. MIT Press, 2006. URL: https://proceedings.neurips.cc/paper/2005/file/a7f592cef8b130a6967a90617db5681b-Paper.pdf.

[43] Josh Wills, Sameer Agarwal, and Serge Belongie. *What went where*. 2003.

[44] Zhirong Wu et al. *Unsupervised Feature Learning via Non-Parametric Instance-level Discrimination*. 2018. arXiv: 1805.01978 [cs.CV].

[45] Dejing Xu et al. "Self-Supervised Spatiotemporal Learning via Video Clip Order Prediction". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2019.

[46] Fengting Yang et al. *Superpixel Segmentation with Fully Convolutional Networks*. 2020. arXiv: 2003.12929 [cs.CV].

[47] Xiaohua Zhai et al. *Scaling Vision Transformers*. 2021. arXiv: 2106.04560 [cs.CV].

[48] Richard Zhang, Phillip Isola, and Alexei A. Efros. *Split-Brain Autoencoders: Unsupervised Learning by Cross-Channel Prediction*. 2017. arXiv: 1611.09842 [cs.CV].

[49] Jun-Yan Zhu et al. *Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks*. 2020. arXiv: 1703.10593 [cs.CV].