Automated Activity Discovery and Object Detection With Computer Vision: Towards Unsupervised Learning for Breakfast to Surgery

Citation

Permanent link
https://nrs.harvard.edu/URN-3:HUL.INSTREPOS:37364735

Terms of Use
This article was downloaded from Harvard University’s DASH repository, and is made available under the terms and conditions applicable to Other Posted Material, as set forth at http://nrs.harvard.edu/urn-3:HUL.InstRepos:dash.current.terms-of-use#LAA

Share Your Story
The Harvard community has made this article openly available. Please share how this access benefits you. Submit a story.

Accessibility
Automated Activity Discovery and Object Detection with Computer Vision: Towards Unsupervised Learning for Breakfast to Surgery

A THESIS PRESENTED
BY
MICHAEL ZHANG
TO
THE DEPARTMENTS OF STATISTICS AND COMPUTER SCIENCE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
BACHELOR OF ARTS WITH HONORS IN THE JOINT CONCENTRATION OF
STATISTICS AND COMPUTER SCIENCE

HARVARD UNIVERSITY
CAMBRIDGE, MASSACHUSETTS
MAY 2020
Automated Activity Discovery and Object Detection with Computer Vision: Towards Unsupervised Learning for Breakfast to Surgery

Abstract

While deep learning in computer vision has enjoyed tremendous success in recent years, much progress remains to be made in important tasks. Many state-of-the-art vision models follow a supervised-learning paradigm, making them dependent on a large amount of labeled training data. This limits both their potential learning capabilities, as well as their application domains. In this work, we present contributions to expand the state-of-the-art in computer vision along both these dimensions, focusing on how to both perceive activities and objects in the video domain.

We first tackle the problem of unsupervised activity discovery, where given a collection of untrimmed and unlabeled videos, we wish to learn a semantically meaningful embedding of the data which can be used to segment the data into “discovered” activities. To expand the learning capabilities of computer vision models, we avoid the intensive bottleneck of amassing a large annotated dataset, and seek to do so without requiring explicit training labels. We consider unsupervised activity discovery from the perspective of the inherently hierarchical nature of activities, e.g. a single complex activity may be modeled as a sequence of smaller sub-activities. Recognizing dependencies across this spectrum of complexities may therefore lead to greater video understanding. Accordingly, we introduce a hyperbolic embedding representation for video data to simultaneously capture hierarchical and semantic relationships in video data. While motivated by their prior success in modeling explicitly hierarchical data found in language, here we show how to leverage hyperbolic representations for the implicitly hierarchical nature of video data. We demonstrate that our hyperbolic video embeddings approach learn representations that significantly outperform the previous state-of-the-art for unsupervised activity segmentation on the Breakfast and 50Salads datasets, and that our hierarchical embeddings naturally allow discovery of activities at multiple levels of complexity.

Following this, we next consider the second challenge of expanding the application domains of state-of-the-art computer vision models. We focus on open, or non-laparoscopic surgery, which represents the vast majority of all operating room procedures. Despite this prominence, few tools exist to objectively evaluate these techniques at scale, and current efforts involve human expert-based visual assessment. We therefore leverage a state-of-the-art convolutional neural network architecture for object detection to detect operating hands in open surgery videos. Automated assessment was expanded by combining model predictions with a fast object tracker to enable surgeon-specific hand tracking. To train our model, we used publicly available videos of open surgery from YouTube and
annotated these with spatial bounding boxes of operating hands. Our model’s spatial detections of operating hands significantly outperform the detections achieved using pre-existing hand-detection datasets, and allow for insights into intra-operative movement patterns and economy of motion.

Finally, we consider how to combine both advances in capability and application domain to focus on automatic activity discovery in surgery videos. Using our developed unsupervised learning algorithm, we demonstrate that we can discover multiple surgical activities in various operating procedures from just a collection of untrimmed and unlabeled YouTube surgery videos. We leverage a state-of-the-art deep learning architecture to extract base features, which we then encode into temporal and hierarchically-aware hyperbolic embeddings. We can then discover an arbitrary number of activities based on our clustering and decoding procedure. Through qualitative evaluation we show how our segmentations align with changes in video activity over time. Based on the alignment of our video segmentations, we present further insights and evidence into the hierarchical nature of activities. In all, our contributions present a step towards fully automated video activity understanding and discovery in real world domains.
## Contents

1 **Introduction**  
1.1 Learning without Labels ........................................ 4  
1.2 Understanding Activities at Multiple Complexities .......... 6  
1.3 Hierarchy with Hyperbolic Geometry ......................... 7  
1.4 Computer Vision Applications to Surgery .................... 9  
1.5 Roadmap for the Remainder .................................. 10  

2 **Deep Learning Background**  
2.1 Artificial Neural Networks .................................... 12  
  2.1.1 Fully-Connected Layers ................................... 14  
  2.1.2 Convolutional Neural Networks ............................ 15  
2.2 Embeddings and Loss Functions ................................ 17  
2.3 Unsupervised and Self-Supervised Learning .................. 19  

3 **Context for Computer Vision**  
3.1 Classical Computer Vision Features ........................... 22  
  3.1.1 Bag of Visual Words ...................................... 22  
  3.1.2 Fisher Vectors ............................................ 23  
3.2 Feature Extraction and Video Processing Pipelines .......... 24  
3.3 Computer Vision Target Tasks ................................ 25  
  3.3.1 Object Detection ......................................... 25  
  3.3.2 Activity Segmentation and Discovery .................... 25  

4 **Unsupervised Activity Discovery with Hyperbolic Embeddings**  
4.1 Related Work .................................................. 30  
  4.1.1 Unsupervised Activity Segmentation in Videos ............ 30  
  4.1.2 Hyperbolic Embeddings ................................... 31  
4.2 Background: How to Learn in Hyperbolic Space ............... 32  
4.3 A Method for Hierarchical Activity Discovery ................ 33  
  4.3.1 Sub-video Segment Sampling ............................... 34  
  4.3.2 Self-supervision with Temporal and Hierarchical Ranking Losses . 36
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.3</td>
<td>Decoding Embeddings and Label Assignment</td>
<td>37</td>
</tr>
<tr>
<td>4.4</td>
<td>Methods</td>
<td>38</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Datasets</td>
<td>38</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Implementation Details</td>
<td>39</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Segmentation Evaluation Setup</td>
<td>40</td>
</tr>
<tr>
<td>4.5</td>
<td>Experiments: Embedding Performance</td>
<td>40</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Hyperbolic Embedding Results</td>
<td>40</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Temporal Loss Ablation</td>
<td>41</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Euclidean vs Poincaré Embeddings</td>
<td>42</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Qualitative Segmentation Examples</td>
<td>42</td>
</tr>
<tr>
<td>4.6</td>
<td>Experiments: Further Embedding Analysis</td>
<td>42</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Structural Analysis of Poincaré Embeddings</td>
<td>42</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Multi-level Activity Comparisons</td>
<td>44</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Multi-level Activity Segmentation</td>
<td>45</td>
</tr>
<tr>
<td>4.7</td>
<td>Additional Hyperbolic Embedding Visualizations</td>
<td>47</td>
</tr>
<tr>
<td>4.8</td>
<td>Discussion</td>
<td>47</td>
</tr>
<tr>
<td>5</td>
<td>Computer Vision for Hand Detection and Tracking in Open Surgery Videos</td>
<td></td>
</tr>
<tr>
<td>5.1</td>
<td>Background: Surgery Motivation</td>
<td>50</td>
</tr>
<tr>
<td>5.2</td>
<td>Related Work</td>
<td>52</td>
</tr>
<tr>
<td>5.3</td>
<td>Methods</td>
<td>53</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Youtube Dataset Curation</td>
<td>54</td>
</tr>
<tr>
<td>5.3.2</td>
<td>RetinaNet Detection Model</td>
<td>55</td>
</tr>
<tr>
<td>5.3.3</td>
<td>Transfer Learning From Pre-existing Datasets</td>
<td>57</td>
</tr>
<tr>
<td>5.3.4</td>
<td>Tracking Hands Across Frames</td>
<td>57</td>
</tr>
<tr>
<td>5.4</td>
<td>Experiments and Results</td>
<td>58</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Implementation Details</td>
<td>58</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Spatial Hand Detection in Open Surgery</td>
<td>59</td>
</tr>
<tr>
<td>5.4.3</td>
<td>Assessing Surgical Performance with Automated Hand-tracking</td>
<td>62</td>
</tr>
<tr>
<td>5.5</td>
<td>Discussion</td>
<td>64</td>
</tr>
<tr>
<td>6</td>
<td>Automated Discovery of Activities in Surgery Videos</td>
<td></td>
</tr>
<tr>
<td>6.1</td>
<td>Methods</td>
<td>67</td>
</tr>
<tr>
<td>6.1.1</td>
<td>Youtube Action Recognition Dataset</td>
<td>67</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Frame-wise Feature Extraction with 3D CNNs</td>
<td>68</td>
</tr>
<tr>
<td>6.1.3</td>
<td>Hyperbolic Embedding Network</td>
<td>69</td>
</tr>
<tr>
<td>6.2</td>
<td>Experiments and Results</td>
<td>70</td>
</tr>
<tr>
<td>6.2.1</td>
<td>Implementation Details</td>
<td>70</td>
</tr>
<tr>
<td>6.2.2</td>
<td>Qualitative Activity Discovery</td>
<td>70</td>
</tr>
<tr>
<td>6.3</td>
<td>Discussion</td>
<td>73</td>
</tr>
</tbody>
</table>
# Listing of figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Visual challenges with activity recognition</td>
<td>5</td>
</tr>
<tr>
<td>1.2</td>
<td>Hierarchical activity understanding</td>
<td>6</td>
</tr>
<tr>
<td>1.3</td>
<td>Poincaré space examples</td>
<td>8</td>
</tr>
<tr>
<td>2.1</td>
<td>Feedforward neural networks</td>
<td>13</td>
</tr>
<tr>
<td>2.2</td>
<td>Convolutional neural networks and components</td>
<td>16</td>
</tr>
<tr>
<td>3.1</td>
<td>Object detection and activity segmentation</td>
<td>26</td>
</tr>
<tr>
<td>4.1</td>
<td>Hyperbolic embeddings for activity segmentation</td>
<td>28</td>
</tr>
<tr>
<td>4.2</td>
<td>Overview of proposed hyperbolic embedding framework</td>
<td>35</td>
</tr>
<tr>
<td>4.3</td>
<td>Hyperbolic performance on Breakfast dataset</td>
<td>41</td>
</tr>
<tr>
<td>4.4</td>
<td>Hyperbolic embedding video activity segmentations</td>
<td>43</td>
</tr>
<tr>
<td>4.5</td>
<td>Video segment embeddings in Poincaré space</td>
<td>44</td>
</tr>
<tr>
<td>4.6</td>
<td>Structural analysis of learned Poincaré embeddings for the Breakfast dataset</td>
<td>46</td>
</tr>
<tr>
<td>4.7</td>
<td>Comparison of Poincaré and Euclidean embeddings across various Breakfast activity classes</td>
<td>48</td>
</tr>
<tr>
<td>5.1</td>
<td>Overview of model for hand detection and tracking in surgery videos</td>
<td>51</td>
</tr>
<tr>
<td>5.2</td>
<td>Dataset collection and characteristics</td>
<td>54</td>
</tr>
<tr>
<td>5.3</td>
<td>RetinaNet architecture</td>
<td>55</td>
</tr>
<tr>
<td>5.4</td>
<td>Surgery hand detection examples</td>
<td>61</td>
</tr>
<tr>
<td>5.5</td>
<td>Surgery hand tracking examples</td>
<td>63</td>
</tr>
<tr>
<td>6.1</td>
<td>Surgery actions: cutting, suturing, tying</td>
<td>67</td>
</tr>
<tr>
<td>6.2</td>
<td>3D ResNet feature extractor components</td>
<td>68</td>
</tr>
<tr>
<td>6.3</td>
<td>Unsupervised activity discoveries in surgery videos</td>
<td>72</td>
</tr>
<tr>
<td>A.1</td>
<td>Example unsupervised activity segmentations in the Breakfast dataset</td>
<td>79</td>
</tr>
<tr>
<td>A.2</td>
<td>Surgery annotation tool, bounding box interface</td>
<td>81</td>
</tr>
<tr>
<td>A.3</td>
<td>Surgery annotation tool, hand keypoint interface</td>
<td>82</td>
</tr>
</tbody>
</table>
In Chapter 5, dataset curation was additionally coordinated by Xiaotian Cheng, Daniel Copeland, Arjun Desai, Melody S. Guan, Gabriel Brat, and Serena Yeung. Xiaotian Cheng, Gabriel Brat, and Serena Yeung also contributed to the draft.

In Chapter 6, dataset curation was coordinated by Krishna Patel, William Locke, Yilun Zhang, Arjun Desai, Melody S. Guan, Gabriel Brat, and Serena Yeung.
Acknowledgments

More than just a project to wrap up senior year, this thesis represents a culmination of my academic experiences here at Harvard, and a bit of who I’ve become along the way. Accordingly, I’d like to extend my gratitude to all those that have given me the opportunity to be where I am today.

To Prof. Serena Yeung, for being a wonderful research mentor and advisor, and for introducing me to an exciting world of research, a piece of which is presented in this thesis. To Prof. David Parkes, for being an amazing undergraduate academic advisor. Was it not for your advice to reconsider my previous thesis topic while pursuing the research that would end up here, who knows where I would be. To Prof. Susan Murphy, for providing another academic home, and for helping me rediscover an interest for research and the many exciting problems in the world. And to Prof. Joe Blitzstein, for being another amazing undergraduate advisor and classroom professor. While I have not always been the best student, your overall patience, gracious accommodation in helping me figure out life between too many commitments, and willingness to give me a chance will always be incredibly valued.

Beyond these advisors, I am also indebted to countless others. To Prof. Boaz Barak, whose gracious classroom accommodation has saved me on multiple occasions from juggling too many research and class commitments. To Dr. Alex D’Amour, for being yet another inspiring research figure and mentor in life.

To my blockmates, both pseudo and real. Thank you for your friendship and support, for going through the challenges and trials of college life together, and for not doing worse things to me when I would finally sleep after too many endless nights in the dhall.

And of course to my family.
1

Introduction

Ever since the re-emergence of the humble neural network as the workhorse and poster-child of deep learning in 2012, computers have amazed humans with their capabilities to recognize visual patterns. These advances have lead to multiple breakthroughs in computer vision, where at a high level it is of great interest to develop algorithms that can learn on their own how to detect what is going on in images or videos. Despite recent developments demonstrating above-human performance in many of these tasks, computer vision is far from a solved problem. The standard approach to achiev-
ing such state-of-the-art computer vision systems is greatly dependent on a large amount of diverse and labeled data. While collecting this data is certainly doable for a wide range of visual domains, the process can be highly cumbersome. Models trained on one set visual domain may additionally fail to generalize to another. Accordingly, deep learning approaches are currently limited both with regard to learning capability and visual application.

In this thesis I discuss original contributions in both of these dimensions. In particular, I focus on the video domain, where our goals include teaching computer vision models to recognize when different activities occur in various videos (Chapter 4), as well as detecting and tracking specific objects of interest (Chapter 5). With respect to images, videos provide a much richer sense of information and potential machine perception courtesy of the additional time dimension. Additionally, dealing with time introduces the opportunity for methods and modeling assumptions grounded in statistics and probability that play a greater role outside of strictly “deep learning” problem formulations. However, the temporal aspect of videos also lends its own set of additional challenges that make video perception a much more challenging task.

To get around the heavy dependency of requiring labeled data to train a computer vision model in a supervised fashion, I first introduce work that advances the state-of-the-art in unsupervised activity discovery in videos (Chapter 4). My work introduces new methods that enable algorithms to automatically detect when different activities occur in various videos without any training labels, which can be costly to acquire. I then focus on extending computer vision for detecting and tracking specific objects of interest to the surgical domain (Chapter 5). Much of the current state-of-art work is trained and evaluated on datasets depicting humans in everyday scenarios, or activities that an ordinary human could observe. While reproducing human vision-based intelligence and perception in machines for these contexts is certainly an impressive feat, compared to this healthcare remains one of the most exciting yet less explored application areas tying computer vision with positive human impact. Furthermore, the almost completely visual nature of surgery, large availability of
unlabeled surgical videos, and high potential to aid human practice with automated methods make the domain particularly promising. I finally combine these contributions to enable a platform for automated activity discovery in surgery videos. Beyond training a model to merely detect and track multiple operating hands during a surgical procedure (already a non-trivial task), I extend my previous unsupervised learning methods to enable the detection of recognized activities throughout the procedure and potential discovery of new ways to organize surgical behavior.

Accordingly, this thesis seeks to highlight contributions for both improving the state-of-the-art in an open computer vision problem and demonstrating practical advances in a specific real-world application domain. In doing so, we keep in mind three larger themes:

1. **One size may not fit all** — Whether this be realizing that a pre-trained model on one dataset does not generalize well to another—or more drastically, training a model that operates in an alternative to standard and textbook Euclidean space may be more desirable—integral parts of the provided contributions lie in noticing the specific benefits provided by one solution over another in specific problem contexts.

2. **An end to end-to-end?** — In an end-to-end learning approach, one (typically large and deep) model is trained using the rawest form of input to generate a desired output, in our case going from raw input image pixels to an object detection or activity segmentation. Depending on the task and the amount of data available, such setups may lead to performance that exceeds a manually designed pipeline. However, such approaches may encounter challenges in the work presented, which may involve a complex output (figuring out the specific start and stop points of various activities throughout a video) or a new domain (surgery both looks very different and progresses differently compared to everyday activities).

3. **Labels are hard to live with and without** — Supervised training involves amassing a large collection of diverse training examples, with additional work required to annotate each sam-
ple with the desired model output information. Creating training datasets can then be man-
ually intensive, error-prone especially with noisy data, and hard to organize. To escape this
however, unsupervised methods may require additional creativity to come up with a valid
training signal from the input data alone. They are difficult in their own way. In both cases
successful deep learning models are more than just interacting with a black box.

For the rest of this chapter I now introduce central concepts and challenges dealt with in this
work. The first three sections preview topics pertaining to Chapter 4, while the last is more perti-
nent to Chapter 5.

1.1 Learning without Labels

The central premise of our first major contribution lies in training a deep learning model to rec-
ognize differences in the activities present in a video from just the video data alone. This challenge
relates to the heart of unsupervised learning, where we do not have access to a label to supervise the
iterative trial-and-error learning process typical in deep learning. We can essentially view this prob-
lem as trying to infer a latent variable sequence from some given observed data. The input video
data (as a sequence of RGB images, or other pre-extracted features discussed in Section 3.1) makes
up our observed variables, where we typically segment the video itself into separate chunks corre-
sponding to individual elements of the sequence, and the names of the activities we wish to identify
are the latent or hidden variables. Our deep learning model acts as a function to map the observed
variables to the latent variables.

Visual input alone is often not enough to distinguish between activities, as any quick Google
Images search can show. Two separate videos that depict the same activity may visually look very dif-
ferent from each other. However, two clips from the same video may seem much closer in appear-
ance, but actually depict different activities (Figure 1.1). Given a ground-truth annotation of what
is actually going on in a video, a deep learning model can learn to reconcile these differences in a supervised manner. Regardless of what the initial input videos look like, the network will be punished if it outputs the same activity for two videos that are visually similar but actually depict different activities, and over time will learn more useful characteristics from the given video information to make more accurate distinctions. These characteristics go into creating a learned representation of the data, often called an “embedding”, which contains all the necessary information to distinguish input data pertaining to one activity from another.

Without supervision, this process can be very challenging. Instead of having a label that allows the model to discover on its own the important (and often complex and nonlinear) characteristics that make videos depicting the same action more similar than those that do not, as researchers we need to provide additional guidance. In the absence of a label, we need to come up with an alternative training signal from the data alone. Distinguishing from all unsupervised learning methods at large, this process more specific to deep learning contexts is called “self-supervised learning”. This frequently involves creating an alternative label that can be generated from the data itself or solving an intermediate task that still allows our deep learning model to generate useful embeddings, and we discuss existing methods further in Section 2.3.
One such training signal involves paying specific attention to the context of video clips to understand activities from data alone. Intuitively, it is more likely that video clips closer to each other in time are part of the same sequence depicting the same activity than those further apart. Accordingly, video clips exhibit a highly temporal dependency, and closeness in time can serve as a proxy training signal for closeness in actual video content.

This concept is not new, having informed various existing unsupervised learning approaches. However, I extend this line of work by discussing how activities are not only inherently temporal, but also naturally hierarchical (Figure 1.2a). For example, a seemingly mundane activity such as “making cereal” can actually be broken down into various subsequent sub-activities, such as “pouring cereal into a bowl” and “pouring milk into a bowl”, and each of these may themselves be broken into even more fine-grained sub-activities such as “grabbing a bowl” or “opening a box of cereal”. This process can continue all the way down to some atomic unit of activity. There is still a temporal relationship at each level, e.g. “pouring cereal into a bowl” comes before “pouring milk into a bowl”, and this relationship can be used to classify the videos that correspond to them as closer in similarity than videos corresponding to another activity in a breakfast context, such as “pulling out a
chair”. However, now we additionally recognize another dimension where we can rank certain video clips as closer to each other than others.

Leveraging both the temporal and hierarchical aspects of activities then lends a more effective unsupervised training signal than previous approaches. When trying to decipher the actual underlying activities from a video, the cutoffs between activities at any one granularity can be somewhat arbitrary. However, with an additional level of similarity that shows that certain video segments “belong” to another segment over another, we get an intuitively stronger observation that a specific timestamp is indeed the end of one activity and the start of another (Figure 1.2b). Besides lending greater structure to organize our embeddings, training on multiple levels of video activity granularity also provides the ability to discover activities at various complexities. This expands the perception capabilities of computer vision models. How we can capture this intuition into something an unsupervised learning model can pick up through a loss function is further discussed in Section 4.3. I then show in Section 4.4 how this contribution empirically leads to state-of-the-art results.

1.3 Hierarchy with Hyperbolic Geometry

Now that we have gone over some notion of learning with hierarchies, I next briefly introduce hyperbolic geometry as a means to actually capture these dependencies in a deep learning model (Figure 1.3). This contribution supports our first underlying theme, where we demonstrate that despite the vast majority of deep learning work being done in Euclidean space, we are not actually limited to one such geometry. Depending on the task at hand, exploring alternatives can actually lead to improved results. Through our hierarchical comparisons between video segments, we can acknowledge an implicit tree-like structure where sub-activity representations of more complex activities make up the child nodes of their more complex parent activity nodes. Encoding such tree-like structures in traditional Euclidean space however either requires higher dimensional embeddings or compli-
Figure 1.3: Different examples of Poincaré space. (a) Embedding a tree-like structure (original source: Nickel and Kiela. 2017⁶⁰). (b) M.C. Escher, Circle Limit I (1958). In Poincare space tessellations that appear smaller and smaller in locations further and further away from the origin actually maintain the same area. (c) Visualization of feature representations of videos in hyperbolic space, a preview of work to come in Chapter 4.

cated graph neural network architectures⁹⁴. As an alternative, hyperbolic space (and in particular the Poincaré ball or disk model) involves a geometry particularly effective at embedding tree-like structures. While the mathematical derivations of doing deep learning in this space are not the main focus of this work, the central premise for their comparative efficiency lies in their geometry. A disk centered at the origin grows exponentially in area as its radius increases, as opposed to quadratically in area in Euclidean space. Intuitively, at much lower dimensions hyperbolic embeddings can more easily “fit in” all the elements of a hierarchical structure such as a tree, where the number of nodes grows exponentially with depth, e.g. traversing all nodes in a binary tree takes $O(2^n)$ time.

While increasingly popular throughout the field, previous work in natural language processing (NLP)¹⁵,¹¹,⁴⁸,⁶⁰,⁷⁹ introduced hyperbolic embeddings to deep learning. Their superiority in embedding hierarchical word relationships with fewer dimensions has lead to a rapidly growing area of work in the deep learning community. Unlike text however, videos are not represented as pre-defined discrete elements (i.e. words) that exhibit explicit hierarchies (e.g. is-a relationships). Accordingly Section 4.3 discusses further how we can extend hyperbolic embeddings to videos, being the first to do so.
1.4 Computer Vision Applications to Surgery

While the previously described considerations highlight our algorithmic contributions for expanding the greater capabilities of computer vision models, we next focus on how to extend such methods specifically to the surgery domain. Our work focuses on open or non-laparoscopic surgeries, where surgeons operate and interact with patient tissue directly with their hands. Differences in patient outcome are significantly linked to surgeon skill and conduct during an operation\footnote{7}, however current attempts to quantify and assess these capabilities involve human-based subjective assessment\footnote{8,4}, which are error-prone and fail to scale. With open surgeries making up over 90\% of all operations in place today, there is a large incentive to develop automated alternatives.

With this motivation, towards automatic action discovery in surgery videos I introduce the first computer vision-based approach for detecting and tracking operating hands in surgery videos. The work demonstrates that such methods can effectively provide quantifiable data capture and eventual automated assessment. In contrast to our previous contribution, here we show that one dataset cannot lead to strong performance for all potential deep learning applications. Our work highlights the important limitation of domain shift present in many machine learning models today, where a large model pre-trained on a comprehensive dataset may still fail to generalize well to specific application domains if the underlying data distributions are too distinct. The results suggest that existing state of the art object detectors trained on current hand detection datasets are unable to perform well in the surgical domain, limiting the conception of a one-size-fits-all plug-and-play deep learning solution. Accordingly, to train our models we systematically curate a dataset of open surgeries from open-source YouTube videos, which are labeled with spatial bounding boxes containing operating hands. Training with this dataset leads to beating all comparable datasets by a wide margin, and this work is further highlighted in Chapter 5. We then move beyond object detection towards the more complicated task of activity discovery in surgery videos in Chapter 6.
1.5 Roadmap for the Remainder

The remaining chapters of this thesis are organized as follows. Chapter 2 introduces relevant background information and describes fundamental concepts for the deep learning aspects of this work, while Chapter 3 provides further context with respect to computer vision and our problem formulations. Chapters 4 and 5 then detail my main contributions expanding learning capabilities and application domains of computer vision. In Chapter 4 we discuss new algorithms for unsupervised learning and activity discovery. To first test these methods we evaluate on established video datasets depicting humans doing various everyday activities, e.g. making breakfast. We then move to the potentially more complicated domain of surgery videos, and in Chapter 5 first show that computer vision methods can be applied to such tasks with our provided contributions. Building on these works, we finally seek to combine our advances in Chapter 6, which presents new unpublished work for activity discovery in surgery videos. Chapter 7 concludes with final remarks and directions for future work.
Deep Learning Background

Computer vision has demonstrated a great deal of promise in recent years. Whether it be for automated object detection in images or recognition and indexing of activities in videos, a variety of methods have been developed to the point of practice in a variety of video-based real-world applications. Supporting these advances and organizing this progress lie multiple developments. These exist with regard to the methods and models that allow computers to process visual
information better, as well as the task formulations that expand what computer vision entails. While for each of my main contributions I give further context in their chapters’ sections, here I discuss the relevant background particularly pertinent and fundamental to the various aspects of our research.

The foundation of this work perhaps best lies with artificial neural networks, computing systems inspired by their biological counterparts designed to recognize various inputs patterns. In Section 2.1 I give a brief overview of two of their most fundamental building blocks in computer vision, fully connected (Section 2.1.1) and convolutional (Section 2.1.2) layers. I further describe the training process and how computer vision models learn useful features in Section 2.2, which is particularly relevant for underlining our contributions in teaching our models how to learn new understandings in novel manners. As previously discussed, collecting the relevant data to train computer vision models is often a limiting factor, and accordingly Section 2.3 presents a deeper dive into unsupervised learning methods.

2.1 Artificial Neural Networks

In laying the foundation of our work, we start with artificial neural networks (ANNs). As an algorithmic concept, they refer to a broad class of algorithms with loose semblance to how a set of biological neurons function. Every ANN is composed of individual neurons, also called perceptrons, that each take in some set of input values, calculate a weighted linear combination of these values, and output either a 0 or 1 depending on if this sum exceeds a certain threshold determined by a nonlinear activation function. As a mathematical function, a perceptron is given by

\[ y = \Phi \left( \sum_{i=1}^{n} w_i x_i + b \right) \]  

(2.1)

where \( y_i \in \{0, 1\} \) is the output, \( x_i \) refers to a specific input, \( w_i \) is the weight associated with each value, \( n \) denotes the number of inputs, \( b \) is a bias term, and \( \Phi \) represents some nonlinear activation function.
function typically differentiable on all positive values in its domain. In a multi-layer model, the output of a perceptron in one layer becomes one input for another perceptron. While not always the case, in this work we only consider “feedforward” networks, where calculations from input to output flow between layers in a single direction. Given this building block, we can then build larger neural network architectures by combining multiple perceptrons together into a single layer and including multiple layers in a single model.

A very typical feedforward ANN could then be composed of: (1) a single input layer whose input values come from the input data and thus has its number of neurons determined by the nature of the dataset, (2) an output layer that acts as the final layer of the model, and (3) one or more hidden layers that in aggregate perform a series of nonlinear combinations as in Equation 2.1 (Figure 2.1). Furthermore, the output layer often depends on the model’s task. In the case of classification, where each data point has some assigned label or class, and we want to assign the correct label to a given data input, we have a neuron for each class, and a final softmax function applied to the vector of all outputs in this layer to normalize the vector to a distribution. If we instead want to output some numerical value in the case of regression, the output layer contains a neuron for each value we
want to predict. Finally, we can also choose to output an encoding of the data or “embedding” of the input data, where the vector of the output layer’s outputs usually serves as some information-preserving representation of the data. Accordingly the number of output neurons is chosen based on the desired dimensionality of the embedding. This last usage is particularly pertinent for unsupervised settings, where for a given data input there is not a specific vector value we wish to produce, and we discuss this further in Section 2.2.

The final model output is thus affected by any outputs of perceptrons in previous layers that eventually feed into the final layer. By extension then, the weights and biases of the individual perceptrons affect this output, and a model “learns” by iteratively adjusting the values of its weights and biases given the initial values of a sample data input such that their aggregate nonlinear combination across all layers produces an output that matches the associated label of the data input. A loss function determines the closeness of this match, which helps direct how the weights and biases update through a process called “backpropagation”\textsuperscript{27}.

We next discuss two particularly important components of the models presented in this thesis: fully-connected layers and convolutional neural networks.

### 2.1.1 Fully-Connected Layers

A fully-connected (FC) or dense layer is simply a layer where all of its neurons are connected to all neurons in the previous layer. In other words, for every neuron in an FC layer, each of its inputs is the output of a neuron from the previous layer (such as in Figure 2.1a). While conceptually mundane, because FC layers are connected to all of these outputs, they perform the important task of consolidating this vector of outputs, or the previous layer’s “feature map”, which contains all the information the ANN has computed to represent a given data input up until then. Accordingly, FC layers are frequently situated in the last layers of a deep neural architecture, where they consolidate the model’s full representation of the data into either a final output vector that results in
a model’s output or another encoded vector representation, i.e. our aforementioned embeddings (Section 2.1). Embedding vectors themselves are often useful, as they are a more efficient representation of the input data. By training a model we wish for the FC layer to output particularly useful embeddings that summarize the input data well. For example, the embeddings of two data inputs that belong to the same class could be close in some distance metric such as cosine similarity, but the embeddings of two data inputs that do not belong to the same class are further apart.

If a neural network is only composed of FC layers, it is frequently referred to as a fully-connected network (FCN). FCNs are typically shallow (as opposed to “deep”, in the sense of not having many layers) and can be “attached” or trained “on top of” larger pre-trained architectures. In this case an already trained model through a previous deep learning task is used to compute an information-rich feature map which is fed into the FCN. Training the FCN with a new loss function can then produce additional useful embeddings of the original data which may highlight possibly different distinctions in the data.

2.1.2 Convolutional Neural Networks

Convolutional neural networks (CNNs) are a broad class of deep neural networks more uniquely situated for computer vision tasks than standard FCNs. CNNs typically assume inputs with some spatial structure (e.g. images, with dimensions for width, height, and color channel), and make use of special layers that perform location-specific operations called “convolutions” (Figure 2.2a). Each convolution involves calculating a reduced representation of the spatial input by sliding a “kernel” over two dimensions of the input, where we take the sum of the element-wise multiplication between the kernel and its overlapping region. In the case of images this helps detect specific visual patterns. In addition to convolutional layers, CNNs also contain pooling layers, which can help streamline computation by reducing the input dimensions of later layers and make the network less sensitive to noisy variations in rotation and position (Figure 2.2b). The layers are stacked on top
of each other such that the feature maps calculated at each layer correspond to iteratively higher-level features and concepts based on the learned aggregation of simpler patterns recognized in the layers below (Figure 2.2c). Along with FC layers and activation layers present in other neural networks, the convolutional and pooling layers in CNNs make them particularly well-suited for visual processing. While also used in non-visual domains such as natural language processing (NLP)\(^\text{16}\), CNNs have enjoyed wide success across various computer vision tasks including image classification\(^{19,73,26}\), object detection\(^{22,68,53}\), character recognition\(^{49}\), semantic segmentation\(^{24}\), activity recognition\(^{14,72,81,10}\), and activity segmentation\(^{16,66}\).
2.2 Embeddings and Loss Functions

As the last layers of a neural network contain all the information required to produce the final output, the effectiveness of a model can essentially be mapped to how well these last feature maps encode the relevant information of a given data input. Accordingly, we would like the neural network to approximate a function from inputs to outputs where the codomain of this mapping, referred to as the “embedding space”, organizes the vector representations of the various data inputs well. In our particular setting, the embeddings of images and videos depicting similar activities should be “close” to each other with regard to some distance metric in the embedding space. Performing a downstream task such as clustering on the embedding space then allows us to separate data into distinct classes.

How we arrive at this depends on the context and nature of the problem. In supervised settings, we can directly compare the final output with a ground-truth label with a loss function, and then in backpropagation update the weights of the model such that the calculated loss is minimized. Doing so changes the vector values of the final feature map. The loss functions in our work include categorical cross-entropy loss for classification, where for \( c \) classes, a single input \( x \), and network’s computation denoted by \( f(\cdot) \) we first normalize the direct outputs \( f(x)_1, \ldots, f(x)_c \) of our neural network with a softmax function to get normalized probabilities

\[
p_i = \frac{e^{f(x)_i}}{\sum_{j=1}^c e^{f(x)_j}} \tag{2.2}
\]

and then plug these into the loss function such that for a single data point

\[
L_{CE}(p_i, y_i) = - \sum_{i=1}^c y_i \log(p_i) \tag{2.3}
\]

where \( y_i \in \{0, 1\} \) denotes the ground-truth label (0 for incorrect class and 1 for correct) and \( p_i \) is
the network’s probability for predicting the $i$-th class. We also use mean squared error for regression, where our loss is akin to minimizing the L2-norm for a single point:

$$L_{\text{reg}}(x, y) = (y - f(x))^2$$

(2.4)

Finally, it may also be desirable to learn embeddings such that multiple outputs downstream of the feature map reduce multiple loss functions. This is the case in object detection where some of our outputs are used to predict the correct class and others are used to calculate the bounding box coordinates that mostly tightly describe where a target object is in a given input image. In Chapter 5 I present such a use-case when training an object detector to identify operating hands.

In the unsupervised setting, because we do not have a final output label $y_i$ to compare against, we more frequently use model architectures that output whole vector embeddings. In this case the last layer’s feature map is itself the desired output. To update our weights such that this output is desirable, a standard approach that we further describe in Section 4.3.2 involves obtaining weights that directly correspond to our previous property of desirable embeddings: the distance between representations produced by similar inputs is smaller than the distance between representations of less similar inputs. For three inputs $x_t, x_p, x_n$ we can do this with a margin ranking loss:

$$L_{\text{ranking}}(x_t, x_p, x_n; m) = \max(0, m + d(f(x_t) - f(x_p)) - d(f(x_t) - f(x_n)))$$

(2.5)

where $x_p$ represents a positive sample that is closer to a target sample $x_t$ in some distance metric $d(\cdot)$ than a negative sample $x_n$, and $m$ represents how different we want these distances to be. Although we do not know the explicit labels tied to our data points, the data input itself contains useful information that allows us to reason about the relative similarity between any two data points, e.g. pick a triplet $(x_t, x_p, x_n)$. Especially when directly working with model feature maps, designing a proper
loss function is fundamental for shaping how our models learn.

2.3 Unsupervised and Self-Supervised Learning

Armed with knowledge on both the fundamental building blocks and learning concepts of deep learning, we close this chapter by discussing the related literature on unsupervised learning, and in particular self-supervised learning. As a subfield, such methods are immensely important with regard to expanding deep learning capabilities. Accordingly, this section aims to connect the previous background to a modern open problem. I provide a brief literature review highlighting self-supervised learning methods in computer vision to give further context to our own work.

Self-supervised methods have recently enjoyed various successes in training deep neural networks to learn useful “understandings” of the input data without explicit or hand-annotated training labels. This understanding is often characterized by the network’s embeddings or feature maps of the data, which can be probed for useful properties (are similar data points close in the embedding space?). Alternatively the network can be used as a feature extractor, where these embeddings are fed to some downstream task such as classification or segmentation, and high performance also indicates a discriminative embedding and the model’s understanding of the data.

While unsupervised learning can refer to any algorithm that makes inferences from data without requiring labels, self-supervised approaches preclude popular unsupervised learning methods such as K-Means clustering by specifically still trying to train deep learning models in ways similar to a standard supervised approach. The intuition is that in the absence of an actual ground-truth annotation, a training signal can still be crafted from the implicit information present in the data itself—which combined with the right loss function and training task—can still allow a model to learn the weights for a useful embedding space.

In vision, many of these proxy tasks stem from intuitions on how humans might process and
understand visual information. For example, to train a model to learn relevant features for associating an image’s pixels to the different objects in an image, one classic approach involves learning to colorize grayscale images\textsuperscript{91-47}. In this approach, the full-color image dataset is available, but images are first converted to grayscale before being fed as inputs to the neural network. Importantly, there is no access to the groundtruth associated with each pixel, e.g. pixel $x$ depicts part of an apple. Instead, the network tries to classify the correct color for each pixel in the image, which can be checked against the groundtruth full-color image in a supervised manner. Accordingly there is no actual label provided in the dataset, but because doing this task successfully involves learning representations that are correlated with their semantic meaning (e.g. an apple can be red or green, but rarely blue or gray), the feature maps produced would be expected to convey some understanding of the data. Another technique involves “inpainting”, or teaching a model to generate the correct pixels for images that purposely have patches removed as part of the training data\textsuperscript{5}. In doing so, a neural network can learn useful patterns through the spatial dependencies between surrounding pixels, where certain relationships may be more indicative of certain semantic contents over others. Finally, the temporal aspect of videos offers another dimension to come up with proxy training tasks. Past work has included sorting the shuffled frames of a video clip into the correct order\textsuperscript{57,93}, correctly identifying which pair of frames appear closer to each other in time\textsuperscript{66,70}, and predicting the relative time with respect to the source video that a frame occurs at\textsuperscript{45}. 
Context for Computer Vision

We now go over material more specific to our context of computer vision and automated understanding from image and video inputs. Despite the popularity of end-to-end deep learning methods in computer vision, an important basis for the field lies in these handcrafted features, which provide semantically meaningful representations of RGB channel information in an arguably more interpretable way. Section 3.1 gives an overview of the relevant techniques
used in this work. Additionally, I give some background context to a typical deep learning vision pipeline in Section 3.2. Finally, in section 3.3 I provide more context for the specific computer vision tasks tackled with in this work: objection detection (Section 3.3.1) and activity segmentation (Section 3.3.2).

3.1 Classical Computer Vision Features

Despite the rise of end-to-end methods for computer vision, more classical techniques providing computed features without deep learning are still in use today. These methods may lend useful representations of an image beyond its inherent RGB channel information, which can then be fed into a deep learning pipeline for better performance. In Chapter 4 we do exactly this by leveraging one such method, reduced Fisher vectors. Although outside the scope of this contribution alone, this section then seeks to provide additional information behind extracting relevant features.

3.1.1 Bag of Visual Words

Since 2004 bag-of-visual-words (BOVW) models have represented images in more semantically meaningful ways than their given RGB pixel representations. First described in Csurka et al. as a “bag of keypoints” approach, BOVW methods take their cue from document-based bag-of-words models in NLP. One way to summarize a document involves counting the number of times each word in the document appears, and then calculating a frequency histogram over all words. Documents containing similar content or semantic meanings can then be identified by similarity measures calculated over their histograms. BOVW models do the same, but swap image features or “keypoints” for the words.

To obtain these keypoints, we can use a number of non-deep learning feature extraction algorithms, such as Scale Invariant Feature Transforms (SIFT). Similarly to the kernels in a convolu-
tional layer, these algorithms essentially split an image into a collection of recognizable patches or local regions, and associate to each region a set of visual features or “descriptors” calculable from the pixel information within it. The local regions can then be quantized into keypoints or “words” traditionally with K-means clustering, where each local descriptor is grouped into a cluster and the K-means center is used the keypoint. These keypoints are then amased into a visual “vocabulary”, and the frequency of these keypoints present in an image is used as its BoVW representation. Doing so can abstract away some of the noise that might appear from a pixel-by-pixel perspective, where the goal is for images similar in semantic content to end up with similar keypoint frequencies. In practice, multiple features can be associated with a single local region to capture different properties. Additionally, when dealing with video data, sequences of frames can be used to calculate local features such as Space Time Interest Points (STIPs) and Improved Dense Trajectories (iDTs) that track pixel changes over time, adding an additional layer of motion information.

3.1.2 Fisher Vectors

Fisher vectors serve as an alternative method to encode additional information compared to BOVW representations. Instead of quantizing the aforementioned features or descriptors through K-means clustering, when calculating Fisher vectors we choose to fit a Gaussian Mixture Model over the descriptors. The Fisher vector is then computed as the concatenation of the gradients of the log-likelihoods for each cluster distribution’s mean and variance. This usually results in a high-dimensional vector (adding two dimensions for each additional descriptor), which can then be expressed more efficiently through dimensionality reduction techniques to obtained reduced Fisher vectors. Intuitively, since we then model a distribution for each cluster without much more computational cost, we can capture additional information regarding the features assigned to each cluster. In cases where the feature-derived keypoint counts between two images are the same, the position and variance of the features within each cluster may still vary, providing additional informa-
tion to more easily discriminate images over the more lossy histogram-based BOVW approach. In practice, Fisher vectors have been shown to beat BOVWs in image classification performance across a variety of datasets \(^{64}\). While modern-day feature maps from pre-trained deep learning methods have become more standard, Fisher vectors retain some popularity and are much more efficient to compute. We accordingly use a form of reduced Fisher vectors in our work on unsupervised activity discovery in Chapter 4.

3.2 Feature Extraction and Video Processing Pipelines

Now we briefly describe some of the additional practical work required to handle videos in computer vision pipelines. While specific steps may depend on the computational task, two near-universal actions that we also use include video-to-frame processing, and image-based feature extraction.

The first action involves converting a given video input into a sequence of image frames. While a growing number of methods are motivated by directly processing videos, such as tubelet proposal networks \(^{33}\) and 3D convolutional neural networks \(^{31,81,89}\), many convolutional approaches still involve processing static channel-by-width-by-height inputs. Accordingly, videos are first converted to image sequences according to a frames-per-second (fps) ratio.

After obtaining this collection of images, for many pipelines it is not necessary to train a model completely from the pixel-based RGB channel image input. Representations using the features described in Section 3.1 can present the same image information in a more efficient manner while preserving spatial dimensionality. Additionally, deep neural network architectures pre-trained on large datasets such as ImageNet \(^{14}\) are now widely available, such that after feeding an image frame into the model, the feature map calculated at its penultimate layer can provide an alternative useful vector representation as described in Section 2.2. Both deep learning and more classical feature extractors can then be used to convert input images into more useful representations. When used to
train our own models, we can often use smaller networks with less training time to achieve similar or better performance than if we had trained a model from scratch.

3.3 **Computer Vision Target Tasks**

We end with some additional context into the two computer vision tasks addressed in this thesis. Section 3.3.1 discusses the premise behind object detection, which is the key challenge we tackle in Chapter 5. Section 3.3.2 then gives an overview of activity segmentation, seen in Chapter 4.

3.3.1 **Object Detection**

Object detection in computer vision refers to identifying and locating objects in an image. In its most basic setting, this localization is done with a bounding box that ideally bounds an object of interest as tightly as possible. Multiple classes of objects can be identified and evaluated. Additionally, as opposed to image classification which typically seeks to only identify a single object class for an image (and so does not require predicting a bounding box), in object detection there can be one, more than one, or zero objects in the image that each require a bounding box, where each class may also appear in multiple locations or not at all. For evaluation and training, although the specific loss function mechanics may differ based on the actual object detector, for each box we generally compare both the predicted class against the ground-truth label in a classification component, as well as the predicted versus ground-truth bounding boxes in a coordinate or area-overlap based accuracy metric. Specific evaluation mechanics relevant to our work are discussed in Chapter 5.

3.3.2 **Activity Segmentation and Discovery**

In activity segmentation, we are interested in the challenging task of determining the sequence of actions or activities that occur within a video. Intuitively we can think of this problem as identifying
Figure 3.1: (a) Examples of object detection using the Faster R-CNN model on the PASCAL VOC 2007 dataset (original source: Ren et al. 2015). (b) Sample activity segmentations on videos from the Breakfast Actions dataset.

the start and stop time points between multiple activities being depicted in a video, or separating a larger video into smaller video clips, i.e. segmentations, that each depict a specific activity. Videos in a dataset may all be of a fixed or variable length. The later scenario often warrants a more challenging but realistic setting, and in this case the videos are often referred to as “untrimmed”. Additionally, the actual activities themselves may have various durations, and different activities may be represented in different videos. During segmentation we typically infer an activity class per frame, where each frame can be assigned a ground-truth label denoting which activity that frame belongs to. Over all such frames we can then calculate a mean accuracy over frames. In unsupervised settings where we do not have this ground-truth information, we typically rely on using our learned embeddings per video frame in some downstream task such as clustering to assign individual frames to specific clusters. Using this approach we can still generate segmentations, and refer to this process as activity discovery. The usefulness of our embedding representation can then be evaluated qualitatively, where although we do not have actual labels to assign to each segmentation we can evaluate them on the basis of if they seem to distinguish between different activities.
Automated recognition and indexing of activity in videos carries great promise for a variety of video-based real-world applications. However, while computer vision has demonstrated several promising advances in a variety of tasks such as activity detection and temporal segmentation, most of this success has been restricted to supervised settings in some sense, whether frame-level or with respect
Figure 4.1: Hyperbolic embeddings for activity segmentation. In this work we learn video embeddings in the Poincaré hyperbolic space, which allows us to directly capture naturally hierarchical relationships in video embeddings. Our embeddings organize related activities by proximity to each other in Poincaré space, and hierarchy by distance to the origin (e.g. closing a box as part of pouring cereal). We propose explicit hierarchical comparisons to recover the implicit hierarchy present in videos and their activities, e.g. making cereal involves pouring cereal and milk into a bowl, and each of these actions is made up smaller actions captured in shorter video segments.

Collecting ordered lists corresponding to individual videos can be arduous even without frame-by-frame annotations. Popular activity segmentation datasets pre-define the set of all available actions \(^2\),\(^{40}\), and moving outside of these test domains and into real-world dynamic settings typically entail a non-trivial amount of additional label curation. However, such approaches fail to scale with the growing wealth of activity recognition applications. These applications then motivate unsupervised approaches to learn useful semantic representations without pre-defined labels.

We build on this motivation by extending a growing area of work in unsupervised activity understanding, where the goal is to identify the individual actions present in a large collection of untrimmed videos without any accompanying frame-level labels or ordered activity lists. Key to this is learning a useful encoding of input video segments, in other words modeling underlying latent...
structure in the data. Clustering in the embedded codomain can then discover semantically similar video clips corresponding to meaningful activities.

While much prior work focuses on time-related dependencies between frames in videos to learn such embeddings\textsuperscript{45,66,79}, in this work we explore modeling a video’s inherent hierarchical structure of activity. Real-world actions are naturally hierarchical, in the sense that complex longer activities are identifiable by the presence of shorter sub-activities or actions, which may themselves be made up of a sequence of even shorter sub-actions. Such observations are not new, having influenced the design of common baseline datasets such as the Breakfast dataset\textsuperscript{45}, but here we seek to directly capture these relationships through hyperbolic embeddings of video data (Figure 4.1). While previous work leveraging temporal dependencies have contributed important advances in unsupervised learning\textsuperscript{45,66}, we show that additionally leveraging the hierarchical relationships between video clips leads to significant performance gains.

We accomplish this by introducing a hyperbolic embedding framework for learning activities directly from a collection of unlabeled, untrimmed videos. Compared to their more common Euclidean counterparts, hyperbolic embeddings (and in particular those modeled with a Poincaré disk geometry) have been shown to more effectively represent hierarchical relationships. Previous work in natural language processing (NLP)\textsuperscript{15,21,48,60,79} has demonstrated their superiority in embedding hierarchical word relationships with fewer dimensions. Unlike text however, videos are not represented as pre-defined discrete elements (i.e. words) that exhibit explicit hierarchies (e.g. is-a relationships).

Our main contribution is therefore a hyperbolic embedding learning approach that allows effectively representing the implicit hierarchical structure in videos, in order to obtain more useful semantic representations. Specifically, we represent video elements as frame-sequences of different lengths, and introduce a video sampling method to capture video element tuples with varied hierarchical and temporal relationships. We then incorporate these with a triplet-based ranking loss
to learn a semantically meaningful embedding space where hierarchical activity structure is naturally captured. We demonstrate that our learned hyperbolic embeddings significantly outperforms prior work when evaluated on unsupervised activity segmentation in the Breakfast\textsuperscript{40,41} and 5oSalads benchmark datasets\textsuperscript{76,77}, where we follow existing approaches for activity discovery from the embedding space (clustering and label assignment). Finally, we further analyze the hierarchical nature and structural representation of our embeddings.

4.1 Related Work

While there is a large body of work on understanding complex activities and automated action recognition\textsuperscript{10,40,42,71,88,90}, we focus on methods that rely on unsupervised and weakly supervised approaches. Our work falls under the common task of activity segmentation, where the objective is to label every frame in the video as belonging to a particular activity class.

4.1.1 Unsupervised Activity Segmentation in Videos

Activity recognition in videos is a popular task, and frame-by-frame video annotations have contributed to great success in activity segmentation\textsuperscript{20,42}. However, these approaches typically suffer from an annotation bottleneck, where labeling real world datasets may be tedious and impractical at scale. To alleviate this, a popular form of weakly-supervised activity segmentation involves weak ordering supervision, where the goal is to align video frames to activities with only the ordering of activity classes available\textsuperscript{8,11,17,28}. Most relevant to our work is the growing area of fully unsupervised learning of action classes, where only the video frames are available\textsuperscript{29,61,57,75}. Previous methods have commonly looked to the temporal structure of videos and their individual frames to impart some form of training signal. Kukleva et al.\textsuperscript{45} use the intuition that actions at large are composed of sub-actions frequently performed in certain orders, such that similar sub-activities are more likely to
occur at similar relative times within their larger actions. They propose a continuous temporal embedding for frame-based features by optimizing a regression loss to predict the relative time in the video of a frame. In contrast, Ramanathan et al. leverage temporal dependencies between frame-level features in the form of a hinge loss that ranks context features in close proximity to sampled target features higher than others. Their method leverages multi-level resolution sampling of these context vectors, sampling at various temporal granularities, and additionally sample negatives from the same videos as those of the context vectors to avoid ranking by visual similarity alone. With respect to both of these works, we acknowledge the importance of temporal dependencies for broader video detection, and consider the hierarchical relationships captured through hyperbolic space to be an additional dimension of information to construct more informative embeddings.

4.1.2 Hyperbolic Embeddings

As previously introduced, hyperbolic embeddings have enjoyed great success over their Euclidean counterparts in embedding data with inherently hierarchical structures. In their original paper, Nickel et al. embed WordNet graphs with an explicit hierarchical structure for learning hypernymies, and show that leveraging these embeddings leads to superior performance in graph reconstruction and linkage prediction tasks. With these graph inputs they optimize a ranking loss that maximizes the distance between embeddings of unrelated objects. Dhingra et al. take this a step further. They propose the use of hyperbolic spaces to embed text directly without a graph-based structure. Their method demonstrates the applicability of hyperbolic embeddings to temporal data, where they impose a Poincaré distance variant of a Skip-Thoughts model. In doing so, they propose a re-parametrization technique to use any popular optimization technique, such as Adam, over the previously requisite Riemannian Stochastic Gradient Descent (RSGD). Along these lines, Ganea et al. expand the versatility of learning hyperbolic embeddings for downstream tasks, by introducing hyperbolic variants for common neural network architectures such as feed-forward net-
works. They evaluate their methods on explicitly hierarchical NLP datasets. In contrast, our work is the first to deal with contiguous video data. While the hierarchical nature of activity discovery is implicit, our contributions demonstrate that hyperbolic embeddings are similarly well-suited for this task.

4.2 Background: How to Learn in Hyperbolic Space

We now give an overview of hyperbolic space, which carries several properties that may make it a more appropriate space to learn hierarchical video embeddings in. We also motivate this with tie-ins to our more specific problem of video segmentation. As previously noted hyperbolic space can be thought of as a continuous version of tree data structures. Unlike in Euclidean space, a disk centered at the origin in hyperbolic space experiences exponential growth in area as its radius increases. This makes hyperbolic embeddings a particularly efficient representation to capture hierarchical structures such as trees, where the number of nodes grows exponentially with depth. Accordingly, such embeddings are capable of preserving arbitrary hierarchical relationships at low dimensions without distortion. We hope to leverage this geometry such that shorter video segments representing sub-activities present in a video can be directly embedded in the same dimensions as the larger segments and more complex activities they make up, and hypothesize that the preserved hierarchical relationships between these segments will also lead to an improved overall structural organization.

Another distinction between hyperbolic and Euclidean spaces is that there are several ways to model hyperbolic space. As in previous work, we choose to learn embeddings with the Poincaré ball model, which is particularly suitable for learning an embedding function due to its differentiable distance function. In this representation we embed points in an open $d$-dimensional unit ball $B^d = \{ x \in \mathbb{R}^d : ||x|| < 1 \}$, where $|| \cdot ||$ denotes the Euclidean norm. The distance between
any two points \( x, y \in B^d \) is given by

\[
d(x, y) = \cosh^{-1} \left( 1 + 2 \frac{||x - y||^2}{(1 - ||x||^2)(1 - ||y||^2)} \right)
\]  

(4.1)

Because the Poincaré ball is conformal to Euclidean space, cosine similarity in \( B^d \) is equivalent to that in Euclidean space \(^{60}\). We aim to learn embeddings \( \Theta = \{ \theta_i \}_{i=1}^N \) for a set of video segments \( S = \{ S_i \}_{i=1}^N \) with a loss function \( L \) that minimizes the distance between video embeddings capturing similar activities and maximizing the distance those capturing distinct activities.

Finally, with Euclidean vector \( u \in \mathcal{R}^n \) we can perform operations with its hyperbolic counterpart \( x \in B^d \) through the exponential—logarithmic bijection:

\[
x = \exp_0^c(u) = \tanh(c||u||) \frac{u}{\sqrt{c}||u||}
\]  

(4.2)

where we let \( c \) denotes the Poincaré ball’s radius and the logarithmic map \( \log_0^c(x) \) is the direct inverse of (2). In practice, we use these maps to transition between the Euclidean and Poincaré ball representations of a vector. For example we can train hybrid models where an encoder with Euclidean weights takes in features in \( \mathcal{R}^n \) and uses the exponential map to output the embeddings in \( B^d \).

4.3 A Method for Hierarchical Activity Discovery

We now describe our method for unsupervised activity discovery with hyperbolic embeddings. As in \(^{45}\), we consider a collection of \( M \) videos \( \mathbf{V} \), where each video \( V_m \in \mathbf{V} \) can be represented as an ordered list of \( N_m \) frames, i.e. \( V_i = \{ v_{1m}^m, v_{2m}^m, \ldots, v_{N_m}^m \} \). Given a collection of unlabeled and untrimmed videos then, where \( N_m \) may vary with the individual video \( V_m \), we aim to discover activities by determining the specific frames corresponding to distinct activities. For \( K \) sub-activities, we
match each frame to a label \( l \in \{ 1, 2, \ldots, K \} \).

Figure 4.2 illustrates our overall approach. Unlike past domains with hyperbolic embeddings, videos do not possess explicit hierarchical structures, let alone a natural discrete vocabulary. Further processing must be done to the videos themselves then to capture hierarchical dependencies. We approach this task by leveraging both temporal and hierarchical dependencies from segments sampled from across our collection of videos. These are then fed into our embedding model. We first generate valid training samples composed of target video segments to anchor our ranking comparisons on, hierarchically-related positives, and hard negatives all from the same video (Section 4.3.1). We then directly embed these segments with a ranking loss function incorporating both temporal and hierarchical relationships (Section 4.3.2). Finally we cluster the resulting embeddings, assign labels to the embedded segments, and assign labels to individual frames belonging to those segments. We detail these procedures in Section 4.3.3.

4.3.1 Sub-video Segment Sampling

As we hope to capture hierarchies in our training samples—something that cannot be done with the individual frame embeddings used in previous work\(^{45,66,70}\)—we generate our training data by sampling whole video clips from within our dataset videos. We introduce a sampling mechanism to efficiently do so across videos as follows. Video clips are sampled as either target \((S_t)\), positive \((S_p)\), or negative \((S_n)\) segments, where segment \( S \subset V \) is also a contiguous video frame sequence. Similarly to the is-a relationships in word hypernyms\(^{60}\), hierarchically related segments (i.e. target-positive pairings) are those that display some form of overlap with respect to their initial time boundaries (Figure 4.2a). This fits our initial intuition where if we wish to recognize a higher-level activity by its lower-level sub-activities from within a video, the lower-level activities must occur in frames from within the bounds of a video segment depicting the higher-level activity, such that \( |S_t \cap S_p| > 0 \). Conversely, to round out our triplet samples we also sample \( S_n \) such that \( S_n \cap S_p \) and \( S_n \cap S_t \) are
Figure 4.2: Overview of our proposed hyperbolic embedding framework. (a) With a set of unlabeled videos we first devise a sampling mechanism that allows us to sample various segments from source videos, making up our triplet-based training dataset (b). Segments themselves are embedded through our encoder \( \Phi \) from Euclidean space \( \mathbb{R} \) to the Poincaré ball \( \mathbb{B} \), which is trained with a hierarchical ranking loss (c, d). Poincaré vector representations \( \Phi^* \) are then used to perform downstream clustering (e) and label assignment to individual frames (f).

Both \( \emptyset \), which are then used to train our embedding model (Figure 4.2b). Because our positive and target segments share video frames, we face the potential issue where segments from the same video are embedded closer to each other based on visual similarity alone. Accordingly as in \(^{66}\) we counteract this by sampling “hard negatives” \( S_n \) from the same video as \( S_t \) and \( S_p \) to encourage our ranking loss to distinguish examples by semantic similarity.

We also sample our segments with a range of different lengths to capture activities at multiple complexities. At the start of each training iteration and for all videos, we randomly select a target length from a set of possible video segment lengths \( L \), and determine where the target segment occurs by uniform-randomly selecting a starting frame index from within the video. We then uniform-randomly sample positive segments by first choosing a segment length from \( L \), and then picking either a starting or ending frame from within the start and end frames of the target segment. Finally
for each positive segment we sample multiple negative segments, where we again determine each segment’s length randomly from $L$. As noted in $^{66}$ and $^{60}$ respectively, both video coverage through the total number of samples and target-positive-negative ratios are an important experimental consideration. We further discuss these details in Section 4.4.2.

### 4.3.2 Self-supervision with Temporal and Hierarchical Ranking Losses

Given a collection of segments sampled from our initial video collection, we now detail our embedding method and loss function. For each training sample $S \in \mathcal{R}^{m \times n}$, we wish to learn an embedding $\Phi(S) \in B^d$ such that the contextual segments are closer in hyperbolic and cosine distance to the target segment than the hard negative. Because we sample segments of multiple lengths, we first convert segments to fixed-length feature vectors $\hat{S} \in \mathcal{R}^m$ by taking the average of frame-based features included in $S$, i.e. $\hat{S} = \frac{1}{|S|} \sum_{v \in S} f(v)$, where $f(v) \in \mathcal{R}^m$ is the initial extracted feature representation of the video frame $v$ that we learn our embeddings on top of. The segments are then fed to a fully-connected encoder $\Phi : \mathcal{R}^m \mapsto B^d$ with two hidden layers of dimensions $2d$ and $d$ respectively, which we train in Euclidean space before outputting our embeddings with the exponential map (Equation 4.2).

To enable unsupervised activity detection of activities, we aim to train our encoder such that embedded segments closer to each other in the Poincaré codomain are more likely to depict similar activities than those further apart (Figure 4.2c). While we hypothesize that the Poincaré distance metric in Equation 4.1 should provide a useful representation by embedding segments depicting sub-activities of the same activity closer to each other due to a common linkage to the larger activity’s embedding, we also explore the addition of a temporal ranking objective as in $^{45,66}$ (Equation 4.4). Experimentally leveraging both in a weighted manner can lead to superior performance. Our
embedding objective for each training sample is given by

\[ L_H = \max \left[ 0, 1 - \left( d_H(\Phi(S_p), \Phi(S_n)) - d_H(\Phi(S_t), \Phi(S_p)) \right) \right] \]  \hspace{1cm} (4.3)

\[ L_T = \max \left[ 0, 1 - \left( \cos(\Phi(S_p), \Phi(S_t)) - \cos(\Phi(S_t), \Phi(S_n)) \right) \right] \]  \hspace{1cm} (4.4)

\[ \mathcal{L} = \lambda_H L_H + \lambda_T L_T \]  \hspace{1cm} (4.5)

where \( d_H \) is the Poincaré distance function in Equation 4.1, \( L_H \) and \( L_T \) are our hierarchical and temporal loss functions respectively, and \( \lambda_H \) and \( \lambda_T \) are weights tuned manually to control their relative importance.

### 4.3.3 Decoding Embeddings and Label Assignment

We next cluster the feature videos into \( K \) clusters following the K-means clustering algorithm with Poincaré distance as a similarity metric (Figure 4.2e). After discovering clusters we use the frame-wise Viterbi decoding procedure in \(^{45}\) to sequentially align video segments to classes (Figure 4.2f). While during training we generate multiple segments at various hierarchical levels, for frame-wise label assignment we embed the individual frame-wise features vectors using our model. To do so we first organize our embeddings into \( K \) clusters, where \( K \) is assumed to be known in an unsupervised setting, e.g. the number of classes one hopes to discover. We are interested in calculating the probability that any one embedding belongs to a class \( k \). Therefore we use our clustering to initialize a Gaussian Mixture Model (GMM), where for each cluster we model \( v_n^m \mid k \sim \mathcal{N}(\mu_k, \Sigma_k) \) and estimate the Gaussian parameters. As each cluster now represents a sub-activity class, for each cluster we average the times of each segment assigned to that cluster, and then compare the associated
segment averages to come up with a cluster-level temporal ordering of sub-activities. These are then used to inform decoding and label assignment. During label assignment, for a video $V_m$ with $N_m$ frames, we are interested in finding the sequence of sub-activities that maximizes the probability of the observed sequence of frame-level feature vectors. We then use the Viterbi algorithm to solve for

$$
\hat{l}_1^{N_m} = \arg\max_{l_1, \ldots, l_{N_m}} \prod_{n=1}^{N_m} \Pr(v_n^m \mid l_n) \cdot \Pr(l_n \mid l_{n-1})
$$

(4.6)

where we let $\Pr(v_n^m \mid l_n = k)$ be the probability that the $n$-th frame in the $m$-th video belongs to the $k$-th cluster. From our previous cluster-wise ordering, we restrict the transition probabilities such that $\Pr(l_n = k_i \mid l_{n-1} = k_j) = 0$ if $i \neq j$ or $i \neq j + 1$ in the ordering. For unsupervised learning, achieving clusters that separate embedded segments based on their depicted sub-activities is sufficient. However for activity segmentation, we finally require a one-to-one mapping between the discovered clusters and the ground-truth labels to calculate an accuracy metric. As in\textsuperscript{1,4,5,70}, we then use the Hungarian algorithm\textsuperscript{44} to find the activity-wise mapping that maximizes the mean over frames (MoF) for evaluating segmentation performance.

4.4 Methods

4.4.1 Datasets

We evaluate our approach on the Breakfast\textsuperscript{40} and 50Salads\textsuperscript{76} datasets. We use Breakfast which is standard due to its relative comprehension, depicting ten common yet complex kitchen activities each containing various sub-activities from a set of 48 total sub-activities. The activities are performed by 52 different actors in multiple kitchen locations captured through different viewpoints, and among all 1989 videos both video duration and frame length associated with sub-activities vary significantly. We also evaluate on the 50Salads dataset due to its relative video duration and natural...
fit for hierarchical activity recognition: compared to Breakfast each video is longer with an average length of 10,000 frames. Each also depicts a variable number of sub-activities, and for evaluation there are two levels of label granularity. As in Kukleva et al., we evaluate segmentation on both “mid” with 17 sub-action classes and “eval” with 9 sub-action classes.

### 4.4.2 Implementation Details

We first generate video segments by randomly sampling video clips of frame length from $L = \{10, 50, 100, 150, 200\}$, and use Fisher Vectors as in for the frame-wise representation. We chose the possible frame-lengths to span the distribution of activity durations known in the dataset (see Figure 4.3a for the frame-length distribution of the Breakfast dataset). In other general unsupervised settings, we can determine lengths using a-priori knowledge or estimates from data. During training, we found that including 16 negative segments for every positive segment and 4 positive segments per target increased performance. Intuitively doing so allows us to make multiple ranking loss comparisons anchored on the same target and positive segments to better shape where the negatives embed, and also lead to more stable loss convergence. For each training iteration, we sampled 16 distinct target segments per video, for 1024 ranking loss calculations per video per iteration. Ratios were selected through cross-validation.

As in, we use a two-layer encoder with embedding dimension size $D = 20$, on top of frame-wise Fisher Vectors, for fair comparison with previous state-of-the-art. However we output our vectors and perform clustering in Poincaré space. We also report results for embedding dimensions $D = 2$ to $D = 20$ to provide further insight, and show further visualization and analysis of the $2D$ embedding space which is easier to interpret than the higher dimensions.

Finally, loss function weights $\lambda_H$ and $\lambda_T$ were manually tuned based on downstream evaluation of activity segmentation on the Breakfast dataset. We initially tried $\lambda_H \in \{0.0, 0.2, 0.4, 0.5, 0.6, 0.8, 1.0\}$ with $\lambda_T = 1 - \lambda_H$, and found that there was an insignificant difference in performance between
Values outside this range lead to a decrease in performance, suggesting that explicitly incorporating both temporal and hierarchical dependencies was beneficial.

### 4.4.3 Segmentation Evaluation Setup

Our decoding procedure involves taking in individual frames as inputs for frame-by-frame sub-activity label assignment. Accordingly, while we train our embeddings with variable-length segments as in Sections 4.1 and 4.2, during evaluation we use our trained embedding model to encode every frame in each video of our test set, producing frame-level embeddings which we fit our clustering methods and perform Viterbi decoding on. For evaluation, we follow prior work: we evaluate on videos belonging to each higher level activity class separately, and the final MoF was calculated as total correct frames divided by total frames counting all classes. We also follow the 4-fold cross-validation protocol: after dividing all videos into 4 datasets, we train on three of them while evaluating on the last. After doing this for all splits, we compute the overall MoF over all held-out sets as a fraction of all correct frames over all total frames, and report this number.

### 4.5 Experiments: Embedding Performance

### 4.5.1 Hyperbolic Embedding Results

We now compare hyperbolic embeddings against prior embeddings for unsupervised activity segmentation, including recent state-of-the-art approaches based primarily on temporal embedding (Table 4.1). We found that with 20-dimensional embeddings (the same dimension as prior work), our encoding plus Viterbi decoding outperforms all prior methods. This suggests the value of an embedding representation that can effectively model hierarchical structure. For the Breakfast dataset, we also show classwise comparisons with the previous state-of-the-art in Figure 4.3b, where we outperform in 8 out of 10 classes.
Table 4.1: Comparison of our full model (ours) and model learned with $\lambda_T = 0$ (no temporal) to previous results. Our models perform competitively with the previous state-of-the-art on Breakfast and 50Salads datasets.

<table>
<thead>
<tr>
<th>Breakfast</th>
<th>MoF (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet $^{66}$</td>
<td>21.2</td>
</tr>
<tr>
<td>I3D $^{10}$</td>
<td>25.1</td>
</tr>
<tr>
<td>Dense Trajectories $^{88}$</td>
<td>31.6</td>
</tr>
<tr>
<td>Temp. Ranking $^{66}$</td>
<td>30.1</td>
</tr>
<tr>
<td>Ranking + Mallows $^{70}$</td>
<td>34.6</td>
</tr>
<tr>
<td>Temp. Regression $^{45}$</td>
<td>41.8</td>
</tr>
<tr>
<td>Ours (no temporal)</td>
<td>41.2</td>
</tr>
<tr>
<td>Ours</td>
<td>47.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>50Salads</th>
<th>Level</th>
<th>MoF (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fisher Vectors $^{42}$</td>
<td>eval</td>
<td>25.7</td>
</tr>
<tr>
<td>Temp. Reg. $^{45}$</td>
<td>eval</td>
<td>35.5</td>
</tr>
<tr>
<td>Ours (no temporal)</td>
<td>eval</td>
<td>43.9</td>
</tr>
<tr>
<td>Ours</td>
<td>eval</td>
<td>41.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>50Salads</th>
<th>Level</th>
<th>MoF (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fisher Vectors</td>
<td>mid</td>
<td>23.6</td>
</tr>
<tr>
<td>Temp. Reg.</td>
<td>mid</td>
<td>30.2</td>
</tr>
<tr>
<td>Ours (no temporal)</td>
<td>mid</td>
<td>28.1</td>
</tr>
<tr>
<td>Ours</td>
<td>mid</td>
<td>31.3</td>
</tr>
</tbody>
</table>

Figure 4.3: (a) Distribution of ground-truth sub-activity lengths. We aim to sample segments that closely align with activity lengths, and across videos most activities are captured by segments under 200 frames long. (b) Activity-wise comparison of sub-activity segmentation MoF with previous state-of-the-art $^{45}$

4.5.2 Temporal Loss Ablation

As described in Equation 4.4, our native model incorporates a context-based ranking loss function similar to that in $^{66}$. We also compared results against embeddings trained without the temporal component of our loss function to quantify the performance coming from only the hierarchical loss. We note that since we use frame overlap as a proxy for hierarchy, even with only the hierarchical loss the learning process can implicitly still factor in relative time as a self-supervised signal. Table 4.1 shows that using the hierarchical loss term alone (no temporal) is already able to achieve very strong performance, while using our full loss is able to further improve, suggesting that a combination of
both explicitly hierarchical and temporal rankings is optimal.

4.5.3 Euclidean vs Poincaré Embeddings

To assess the value of learning our embeddings in Poincaré space, we also compared our base model with a Euclidean counterpart (Table 2). Here the model architecture is preserved and we still try to learn both hierarchical and temporal dependencies in our loss, but use Euclidean distance for the distance metric. Across a variety of embedding dimensions, we found that Poincaré embeddings outperformed their Euclidean counterparts, although MoF for both manifolds improved with increased dimensionality. A primary appeal of hyperbolic embeddings is their ability to capture tree-like relations more efficiently. Consistently better performance then suggests the promise of hierarchical understanding for activity segmentation.

4.5.4 Qualitative Segmentation Examples

Figure 4.4 contains examples of our unsupervised segmentations. Furthermore, in Figure 4.5 we visualize example embeddings of our variable length segments sampled across videos during training. Our embeddings show natural clustering of different sub-activities, and embedded segments depicting similar sub-activities from different videos are still near each other. Each segment makes up a single point, where the activity class is illustrated by color and marker shape. The size of the markers indicates the size of the embedded segment (larger depicts a longer segment).

4.6 Experiments: Further Embedding Analysis

4.6.1 Structural Analysis of Poincaré Embeddings

We finally perform a set of experiments on the Breakfast dataset to further inspect the relationships learned through our Poincaré embeddings. As seen in Figure 4.6, our embeddings are able to cap-
Table 4.2: Manifold and dimension embedding ablations. Poincaré embeddings outperform their Euclidean counterparts on Breakfast segmentation MoF.

<table>
<thead>
<tr>
<th>Dimensionality</th>
<th>Breakfast</th>
<th>50Salads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manifold</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>34.0</td>
<td>30.9</td>
</tr>
<tr>
<td>3</td>
<td>37.4</td>
<td>31.4</td>
</tr>
<tr>
<td>5</td>
<td>38.3</td>
<td>33.5</td>
</tr>
<tr>
<td>10</td>
<td>38.7</td>
<td>33.7</td>
</tr>
<tr>
<td>20</td>
<td>40.1</td>
<td>34.6</td>
</tr>
</tbody>
</table>

Poincaré embeddings:

- Breakfast:
  - 2: 36.7
  - 3: 38.9
  - 5: 42.5
  - 10: 43.5
  - 20: 47.1

- 50Salads:
  - 2: 31.3
  - 3: 34.8
  - 5: 35.1
  - 10: 37.0
  - 20: 36.7

Figure 4.4: Examples of activity segmentation on Breakfast videos, using clustering and Viterbi decoding on our learned Poincaré embeddings.

Figure 4.4: Examples of activity segmentation on Breakfast videos, using clustering and Viterbi decoding on our learned Poincaré embeddings.

ture both hierarchical and temporal information (Figure 4.6a). We include visualizations for both complex activities such as “coffee” and “milk”, as well sub-activities of the “fried egg” activity class (the three on the right). In the top row, we show that our embeddings are able to distinguish video segments by their length; longer segments tended to gravitate towards the origin (red), while shorter ones were pushed to the edge (blue). This makes sense as longer segments typically correspond to more complex activities, so our embeddings are able to pick up on the hierarchical structure, despite there not being any explicit hierarchy given in the input videos. In the middle row, our embeddings also distinguish video segments by their relative temporal occurrence. We see this pattern es-
Figure 4.5: Video segment embeddings in Poincaré space. We train our model to learn hyperbolic embeddings for ten higher level activities in the Breakfast dataset. In both (a) and (b), we show embeddings learned across multiple videos, and display select segments from the same video in (a). We show that longer segments with the same groundtruth label as smaller embeddings carry additional related activities, such as the different phases of stir-frying an egg or buttering a pan. Our model also embeds similar sub-activities from different videos in proximity despite visual differences (b).

especially present when inspecting subclasses with a clearly defined temporal order as with “friedegg”: “take_plate” often occurs at the beginning (blue) or end (red) of a video, depending on how an individual gets ready, while “crack_egg” and “put_egg2plate” unambiguously happen towards the beginning and end of the “friedegg” activity respectively.

4.6.2 Multi-level Activity Comparisons

We additionally explored how our embeddings structured groundtruth video segments from multiple label granularities, and embedded the groundtruth segments for “coarse” and “fine” labels in the Breakfast dataset. Then given a segment from the coarse level, we found the nearest fine-level segments with respect to cosine distance, listing examples in Table 3 along with their hyperbolic norms. We see that the labels are intuitively related, e.g. fine-level segments “carry_cereal” and “open_cereal” are close to the coarse-level “pour_cereals” clip. Additionally, coarse label seg-
Table 4.3: Nearest unique neighbors with cosine distance for Poincaré embeddings of groundtruth activity segments. We list the nearest “fine”-level segments under sample “coarse” groundtruths in the Breakfast dataset, also including the Poincaré distance to the origin. We note that across sub-activities, the closest fine actions are intuitively related to their coarse counterparts.

<table>
<thead>
<tr>
<th>Action</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>pour_water</td>
<td>0.32</td>
</tr>
<tr>
<td>stirfry_egg</td>
<td>0.38</td>
</tr>
<tr>
<td>cut_fruit</td>
<td>0.65</td>
</tr>
<tr>
<td>put_toppingOnTop</td>
<td>0.77</td>
</tr>
<tr>
<td>reach_cup</td>
<td>0.34</td>
</tr>
<tr>
<td>carry_spatula</td>
<td>0.38</td>
</tr>
<tr>
<td>cut_fruit</td>
<td>0.66</td>
</tr>
<tr>
<td>reach_toppingpiece</td>
<td>0.75</td>
</tr>
<tr>
<td>pour_water</td>
<td>0.46</td>
</tr>
<tr>
<td>fry_egg</td>
<td>0.36</td>
</tr>
<tr>
<td>reach_fruit</td>
<td>0.73</td>
</tr>
<tr>
<td>reach_knife</td>
<td>0.76</td>
</tr>
<tr>
<td>carry_kettle</td>
<td>0.37</td>
</tr>
<tr>
<td>scramble_egg</td>
<td>0.44</td>
</tr>
<tr>
<td>carry_fruit</td>
<td>0.64</td>
</tr>
<tr>
<td>reach_bread</td>
<td>0.79</td>
</tr>
</tbody>
</table>

ments embed closer to the origin than fine-leveled segments, which corresponds to the behavior of variable-length segments observed through the embedding visualizations in Figure 4.6a. We also compared embeddings between coarse labels from the same activity class. Using agglomerative clustering, we recreate “parse trees” for the segments (Figure 4.6b), listing the Poincaré distance at each node. This suggests that iteratively clustering segments in an expanding manner could discover higher-level activities not found in our groundtruth labels, e.g. “smear_butter” happens in close hierarchical proximity to a combined larger activity that involves “put_bunTogether” and “put_toppingTop”.

4.6.3 Multi-level Activity Segmentation

Finally, we show that the hierarchical structure in the hyperbolic embedding space can naturally enable alternate segmentations of activity classes at different levels of granularity, as is indeed the most natural fit for video data with inherent hierarchical structure. To do this, instead of decoding individual frame embeddings as in Sec 4.3 and Kukleva et al. 2019, we hypothesize that feeding in representations of multi-frame segments could lead to alternate segmentations at different granularity. As we now cluster multi-frame segments, we adjust our label assignment procedures as follows. During evaluation, we continue to sample segments of differing lengths, from the same set \( L \) as during training. We then filter the embeddings by segment length, and fit clusters on embeddings.
Figure 4.6: Structural analysis of learned Poincaré embeddings for the Breakfast dataset. (a) We visualize class-wide embeddings of variable-length segments by both length and relative time of occurrence, including examples for both higher level classes and sub-activities. Our embeddings learn to distinguish segments both by length (top row, short blue vs long red), organize by relative time (middle row), and separate sub-activities. (b) "Parse trees" from agglomerative clustering of coarse groundtruth segments. We list Poincaré distance at each node. (c) Multi-level activity segmentation. We compare segmentations from clustering embedded segments of different lengths. Colors represent discovered activities (clusters), and are distinct for the different levels shown. Decodings tend to agree on higher-level segmentations, but shorter inputs help discover organized shorter segmentations indicative of finer activities.

belonging to a single frame length for all frame lengths. For each level, we use the level-specific clusters to come up with an alternate video-level decoding as in Sec 4.3. Finally for each newly labeled embedded segment, we assign every frame associated with that segment to the sub-activity label. In cases of overlap (i.e. a frame has two or more associated labels from being in two or more distinct embedded segments), among the possible labels we pick whichever label was most common. The procedure essentially performs the same decoding algorithm for each represented segment length.

In Figure 4.6c, we show example segmentation at differing granularities. To interpret the figure
we focus on the changes in assigned label, as coloring is arbitrarily assigned during clustering (i.e. any matches between the two levels are by coincidence). The segmentations with shorter input segments are more noisy, but we find qualitative evidence for discerning sub-activities while still observing relative agreement in the division of activities across levels. By including when the Breakfast-provided coarse label occurs, we gain additional insight into lower-level discovery. For example, “stir_milk” is recognized as a single activity with 50-frame segment inputs, but a cyclic or repetitive action is discovered with using 10-frame inputs. Similar agreement with additional segmentation occurs with the “cereals” videos between segmentation with 10 and 100-frame input segments. Our results suggest hyperbolic embeddings can help discover multiple levels of sub-activities without explicit labels.

4.7 Additional Hyperbolic Embedding Visualizations

We further present visualizations of our two-dimensional Poincaré and Euclidean embeddings trained on segments sampled from various videos depicting activities in the Breakfast dataset (Figure 4.7). As in previous work with Poincaré embeddings\textsuperscript{15,48,60,79}, our 2D embeddings conform to the Poincaré disk, with discernible sectors corresponding to various activity classes.

4.8 Discussion

In this work we introduced a hyperbolic embedding approach for unsupervised activity discovery. Our approach is motivated by the inherent hierarchical structure of activities in video, and the hypothesis that simultaneously modeling both temporal and hierarchical relationships can lead to more useful semantic embeddings. We demonstrate that our embedding approach leads to state-of-the-art performance, and we present further analysis and examples of multi-level activity segmentation using the former. Our findings suggest that stronger hierarchical modeling through hyperbolic embeddings is a promising direction for video activity understanding and discovery.
Figure 4.7: Comparison of Poincaré and Euclidean embeddings across various Breakfast activity classes. Sub-activity is illustrated by color. We are able to learn useful organization with two dimensions both regarding length of the activity (bigger vs. smaller symbols) as well as the semantic sub-activity (color) in Poincaré space. Euclidean embeddings do illustrate some sub-activity separation, but lack hierarchical organization. Legends include each associated sub-activity per activity class.
Computer Vision for Hand Detection and Tracking in Open Surgery Videos

In this work, we take a significant step towards automated open surgery assessment with computer vision. Having shown that deep learning-based approaches can accomplish unsupervised activity discovery in everyday domains, we initiate our exploration into
the surgery setting with a supervised approach to detect operating hands in surgery videos. While prior work involving supervised convolutional neural networks (CNNs) has shown promise for detecting tools, several conditions make the present task more difficult. Unlike tools, hands are visually deformable and may vary greatly in appearance. In addition, open surgery’s enhanced generality—where surgeons may frequently manipulate several tools from multiple angles—along with variations in video quality, lighting, zoom levels, and camera angles present further challenges. Finally, while supervised deep learning models have demonstrated success in difficult object detection tasks taken from everyday settings\textsuperscript{51,52}, they do so with significant quantities of diverse training data. Given the lack of obvious equivalents in open surgery, our problem additionally motivates the collection of similarly diverse training videos.

We therefore combine a CNN framework for object detection with a diverse collection of open surgery videos obtained from YouTube, a publicly available data source, which we annotate for the spatial bounding boxes of operating hands. Our object detection algorithm leverages \textit{RetinaNet} \textsuperscript{51}, a state-of-the-art CNN based on single-stage feature extraction and focal loss classification. Our approach achieves strong performance for detecting hands on a subset of the collected YouTube videos held-out for evaluation, substantially outperforming models with identical architectures trained on pre-existing hand-detection datasets. Finally, we show that combining our predicted detections with fast object-tracking algorithms enables temporally-consistent hand-tracking in surgery videos, allowing for further analysis regarding movement patterns and economy of motion to assess surgical performance.

\section*{5.1 Background: Surgery Motivation}

Improvements in surgical outcomes have been achieved by careful analysis of peri-operative hospitalization data to identify best practice and standardize care. Identification of evidence-based metrics
Figure 5.1: In this work we propose a computer vision-based deep learning method to enable automated open surgery understanding. We collect a set of publicly available, open surgery videos from YouTube (a), and annotate video frames from these with spatial bounding boxes of operating hands to train a RetinaNet model for hand-specific object detection (b, c). Our model takes frames of unlabeled videos as input, and outputs inference for detection (d, top), which we follow with downstream hand-tracking (bottom) in surgeries.

has led to numerous pre- and post-surgery care pathways that significantly reduce re-admissions and morbidity. Despite such peri-operative standardization, significant differences remain in patient outcomes when stratified by surgeon\textsuperscript{7}. Accordingly, surgeon skill and conduct during an operation can significantly determine peri-operative outcome.

Despite these incentives, efforts to improve surgical quality are hampered by limited quantity and quality of data within the operative episode. Surgical procedures are often merely recorded in retrospect by the practicing surgeon. These operative notes serve as the principle record of the surgery, but often poorly identify critical steps and overlook important aspects of individual procedures\textsuperscript{82}. Reliance on the subjective recall of surgeons also prevents effective cross-surgeon comparison and feedback on adherence to evidence-based practice\textsuperscript{55}.

Objective intra-operative data has the potential to generate new quality metrics and augment the capabilities of the surgeon\textsuperscript{Topol}. Various studies have demonstrated promising results using deep learning and computer vision for frame-level surgical tool detection in laparoscopic procedures.
52
paratively easier to adopt and more readily available, where one camera setup in an operating room can be extended to track an arbitrary number of hands. Additionally, previous issues regarding camera-based tracking systems and line-of-sight obstructions \(^1\) can be tackled with recent advances in computer vision, and we demonstrate that our approach is robust to visual occlusion in many circumstances.

Outside of the surgical domain, there has been interest in hand-specific localization and tracking methods targeted to hands in the computer vision community. This has led to the release of datasets such as the EgoHands dataset \(^3\), which contains 4800 labeled video frames of hands from a first-person camera perspective, and the Oxford Hands dataset \(^5\), which contained annotations for 13050 hand instances in third-person perspective images. While previous work in domain transfer might suggest that deep learning models trained on these diverse pre-existing datasets could then trivially be applied to track hands in the surgical setting, we empirically show that this is not the case. A separate line of work in computer vision has aimed to estimate entire body poses including hand keypoints \(^2\), \(^4\), \(^6\). Although these do not explicitly target hand detection, in theory such methods could produce hand detection by calculating the tightest bounding box around all estimated hand keypoints. However, we found that these indirect approaches did not work well in practice on surgical videos. In our work, we therefore collect and annotate a new dataset of hands in videos of open surgery, and leverage a state-of-the-art object detection framework \(^5\) to train an effective model for detecting and tracking hands in surgical video.

5.3 Methods

In this section, we first describe the data that we collected for model development. We then present our model for hand detection in video frames of open surgery, and then describe how we link these detections across frames to track hand movement.
5.3.1 Youtube Dataset Curation

In order to obtain a diverse set of data for our model development, we scraped videos of open surgery from YouTube. Specifically, we utilized a list of search terms compiled from all surgical procedures covered by a large insurance company (Southern Cross Health Society). Individual terms from the list were searched on YouTube (e.g. “sublingual gland excision”) and queried results were added if they depicted open surgery, included surgeon hands, and were 144p or greater resolution.

From all search terms in the breast, gastrointestinal, and head-and-neck surgery sections of the insurance company’s list, 23 yielded at least one unique useful video (Fig. 5.2a). To train and evaluate our model on a good representation of surgeries, we then filtered our search to a dataset of 188 videos, consisting of 70 breast, 88 gastrointestinal and 30 head-and-neck videos.

We next selected a subset of all potential video frames to label with bounding box hand annotations for hand detection. Videos were first converted to 15 fps. For videos over 20 minutes long, only the middle 20 minutes were processed. We then selected 10 frames at uniform intervals from each video, which allowed our image set to maintain diversity across different videos with respect to what kind of surgical procedure was being performed, the number of hand instances observed, and various video attributes such as quality, frame size, camera angle, perspective, and zoom.
To obtain annotations of the spatial boundaries of hands, we developed a web-based application allowing research assistants to draw bounding boxes for all hands in a given image (Figure 5.2b). The tool additionally asked annotators to indicate whether each hand was a right hand or a left hand; however for the purposes of this work, we did not use these annotations. Seven trained research assistants performed the labeling. Our final annotated dataset consists of 1880 labeled images, each containing zero or more bounding boxes associated with a hand. In all, 700 images came from breast surgery videos, 880 from gastrointestinal and 300 from head-and-neck videos. Among the aggregated images, 940 were allocated for training, 380 for validation, and 560 for testing. We split breast, gastrointestinal, and head-and-neck images equally across all sets.

### 5.3.2 RetinaNet Detection Model

Our hand detection model is based on the RetinaNet neural network architecture, a CNN-based model for object detection that has achieved state-of-the-art performance on object detection tasks. We direct readers to Lin et al. for model details, but briefly describe RetinaNet’s architecture as
a single neural network composed of three parts: a backbone responsible for converting image pixel representations to convolutional features, and two task-specific subnetworks for classifying and regressing bounding boxes. The backbone incorporates a base Feature Pyramid Network (FPN)\textsuperscript{50}. This rests on top of a ResNet architecture which has been shown to extract powerful visual features\textsuperscript{25}. The FPN enables feature processing and object detection at multiple resolutions. Each level of the pyramid feeds into two fully-convolutional subnets, which aim to classify objects at every spatial position and output their spatial boundaries.

To output predictions, RetinaNet divides input images according to a set of pre-defined reference boxes called “anchors”, which correspond to sliding window positions of the backbone-generated feature map. Given an annotated image with a ground-truth bounding box, anchors are then assigned to the bounding box if the intersection-over-union (IoU) is greater than 0.5, and to background if IoU \( \leq 0.5 \). Image features are input to the subnetworks, which share the same architecture. We train the bounding box nets with standard L2 regression loss functions.

In contrast to previous methods, RetinaNet uses a novel focal loss for classification:

\[
\text{FL}(p_t) = -(1 - p_t)^\gamma \log(p_t), \quad p_t = \begin{cases} 
  p & \text{if } y = 1 \\
  1 - p & \text{otherwise}
\end{cases}
\]  

(5.1)

where \( y \in \{-1, 1\} \) denotes a ground-truth class (e.g. if a hand exists or not) for an image representation, \( p \in [0, 1] \) denotes the estimated probability for the class, and \((1 - p_t)^\gamma\) serves as a modulating factor with tunable focusing parameter \( \gamma \geq 0 \). Object detectors typically face a huge class imbalance prediction problem, where when trying to assign anchors to either target foreground classes or background, they may scan \( 10^4 - 10^5 \) candidate locations per image only for a few to actually contain objects\textsuperscript{51}. Accordingly the \((1 - p_t)^\gamma\) tries to downweight easy examples, focusing training on hard negatives to maintain an optimal positive/negative ratio.
5.3.3 Transfer Learning From Pre-existing Datasets

While we anticipated that training on existing hand datasets alone would not transfer well to surgery due to large differences in appearance, we were interested in whether pre-training on them could help our model learn useful general hand detection feature maps, to be fine-tuned on our surgery video data. We tried various sequential pre-training permutations. The datasets considered were:

- **EgoHands**\(^4\), a collection of 4800 images from 48 videos containing pixel-level ground-truths for over 15 thousands hands. Videos depict various interactions between people, and each contribute 100 labeled images containing spatial boundaries of hands.

- **Oxford (Hands)**\(^5\), a comprehensive compilation of 13050 hand instances from various public image datasets. Of these, 4170 are considered high quality hand instances with hands larger than 1500 square pixels.

5.3.4 Tracking Hands Across Frames

Motivated by the importance of surgical assessment and correlation between metrics such as economy-of-motion with surgical skill and medical outcomes\(^7\), we next apply the output of our hand detection model to frame-by-frame hand tracking in surgery videos. We use the *Simple Online and Realtime Tracking* (SORT) algorithm, which enables multiple object tracking from un-identified bounding box inputs\(^6\), and selected SORT for its simplicity and ability to perform quick inference for real-time tracking.

SORT enables object-specific tracking through (1) object state propagation, (2) detection association with existing objects, and (3) lifespan management of tracked objects\(^6\). During state propagation, SORT calculates metrics such as position, velocity, and box size given non-identified input bounding boxes and their spatial displacement across multiple frames, using a Kalman filter for
motion prediction. Accordingly, for every updated prediction of each detection instance, SORT generates a predicted bounding box for the next frame, associating subsequent detection ground truths with the previous frame’s boxes based on IoU and performing assignment given these metrics with the Hungarian algorithm. Finally to avoid unbounded growth of tracking identities, native SORT deletes identities every time objects move out of frame. However, due to the frequency at which hands belonging to the same surgeon may move in and out of sight in our dataset, we adapted SORT such that instead of deleting and creating a new identity upon an object’s exit and re-entry into view, we merely update the pre-existing identity with the re-entry state calculated through our object state propagation and keeping track of items in a first-out, first-in manner. After model detections, we also augment SORT’s tracking performance by interpolating predicted bounding boxes between frames during post-detection processing. Details are provided in Section 5.4.1.

5.4 Experiments and Results

We now evaluate our approach on detection and tracking of open surgery operating hands. In Section 5.4.1 we include further details on how we trained our model for detection and processed outputs for tracking. Section 5.4.2 contains quantitative evaluations for our approach using the surgery hands dataset, and Section 5.4.3 presents qualitative examples of hand tracking towards assessing surgical performance.

5.4.1 Implementation Details

For RetinaNet, we followed Lin et al.\textsuperscript{51}, where we used a ResNet-50-FPN backbone network, and an IoU threshold of 0.5 between predicted bounding boxes and ground truth labels to denote a positive instance. All models were trained using the Adam optimizer with learning rate of 10\textsuperscript{-5} and batch size of 4. Hyperparameters were compared on the validation set, and best parameters
were selected to train on a combined training and validation set for the final model. For all datasets, we first trained our models for at least 50 epochs or until convergence. For those pre-trained on multiple datasets, we trained sequentially, training until convergence completely with one dataset before moving on to another. At each stage, the best performing checkpoint was then selected for additional pre-training with a subsequent dataset or fine-tuning with our surgery dataset. For fine-tuning with the surgery dataset, we trained for 10 epochs. Total training time for our largest dataset permutation took approximately five hours on an NVIDIA GeForce RTX 2080 Ti GPU. For post-detection processing, we found that a temporal window size of one frame (overall interpolation context of three frames), and a simple max-voting procedure to determine addition or subtraction of bounding boxes in the middle frame, led to the most stable SORT tracking output. After running test set frames through our hand detection model, we calculated interpolations with a stride-one sliding window for all frames from the same original video.

5.4.2 Spatial Hand Detection in Open Surgery

We compare hand detection performance using RetinaNet in Table 5.1. Training with our collected surgery hands dataset significantly outperforms training using existing hand datasets, and to the best of our knowledge our model is the first to demonstrate effective visual hand detection on real-world open surgery videos. All models were implemented with an identical RetinaNet architecture and tested against the same open surgery frames. For models trained using multiple datasets, we list the datasets in training order. We use mean average precision (mAP) to evaluate our model. A detection is considered correct if the intersection-over-union of a predicted bounding box with ground truth is at least 50%, i.e. an IOU threshold of 0.5.

Despite the vast quantity of annotated data in both the EgoHands and Oxford datasets, models only trained on these datasets perform substantially worse in comparison to those trained with our surgery data, suggesting a significant domain transfer problem related to the characteristics and rep-
Table 5.1: Surgery detection performance across annotated hand datasets. Although hands are present in all datasets considered, training a RetinaNet model on existing datasets generalizes poorly to the surgery domain. We obtain a significant performance boost using our contributed data, demonstrating its value for detecting hands in surgery videos.

<table>
<thead>
<tr>
<th>Training data (comma-separated by training order)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>EgoHands</td>
<td>11.8</td>
</tr>
<tr>
<td>Oxford</td>
<td>3.4</td>
</tr>
<tr>
<td>EgoHands, Oxford</td>
<td>5.9</td>
</tr>
<tr>
<td>Oxford, EgoHands</td>
<td>9.4</td>
</tr>
<tr>
<td><strong>Ours (surgery hands)</strong></td>
<td><strong>70.4</strong></td>
</tr>
</tbody>
</table>

representation of hands in a surgical environment. To improve performance, we explored pre-training with additional datasets, doing so sequentially with both the EgoHands and Oxford datasets. Because pre-training was done in succession, we also experimented with the order of training data. The model with our dataset remained ahead, but we found interestingly that on multiple occasions the biases present in the Oxford dataset seemed to be detrimental to performance during fine-tuning.

Table 5.2: Pretrained RetinaNet detection performance with fine-tuning on our dataset

<table>
<thead>
<tr>
<th>Pre-training dataset (comma-separated by training order)</th>
<th>mAP (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>70.4</td>
</tr>
<tr>
<td>EgoHands, Oxford</td>
<td>75.4</td>
</tr>
<tr>
<td>EgoHands</td>
<td>74.8</td>
</tr>
<tr>
<td>Oxford, EgoHands</td>
<td>71.5</td>
</tr>
<tr>
<td><strong>EgoHands, Oxford</strong></td>
<td><strong>75.4</strong></td>
</tr>
</tbody>
</table>

We also investigated if pre-training on hand-detection datasets could lend useful detection priors for fine-tuning with our surgery hands dataset. Accordingly, we next evaluated performance using RetinaNet with our dataset and different pre-training permutations. We found that pre-training generally helped, and a permutation with all three datasets lead to our best-observed performance of \(\text{mAP} = 75.4\) (Table 5.2).

Figure 5.4 contains example frames from our best-performing detection results, where our model successfully detects a variety of hand instances ranging in number, zoom, and image quality. Our
Figure 5.4: Example hand detections on open surgery frames. Blue boxes represent ground truth, and cyan boxes denote predictions. We found the model to be effective across a variety of video qualities and camera perspectives, also successfully detecting multiple hands (b, c, d, f, g, h), and occluded instances or partial hands (c, e, g). Mistakes skewed towards false positives, with misclassification of objects appearing in plausibly similar contexts to hands.

The model is effective at detecting hands in a variety of situations, including close-up shots (5.4a, 5.4e), heavy occlusion by boundaries (5.4c, 5.4g), and lighting variations (5.4d, 5.4h). In aggregate our model suffered more from false positives, most prominently tending to misclassify objects that were both visually similar and appeared in positions or plausibly similar contexts to correct hands in the image (such as the cloth in Fig. 5.4j, 5.4k, 5.4l). This suggests that our model learned more than hand-specific visual features to classify operating hands, and also provides one possible explanation to why visually-consistent hands in pre-existing datasets do not transfer well to the surgical domain.
5.4.3 Assessing Surgical Performance with Automated Hand-tracking

We now assess the second part of our methods related to hand tracking. Given frame-by-frame model-predicted hand detections from our trained RetinaNet detector, we use SORT to generate hand-specific tracking predictions. Although we are unable to quantitatively assess tracking without ground-truth trajectory labels over sequences of video frames, we include qualitative examples of tracking outputs and their respective expert-driven interpretation in Figure 5.5, additionally creating multi-hand trajectory maps to aid in visualization (last column). These instances show that our tracking output is useful for assessing surgical characteristics such as motion pattern and economy of motion in various procedures.

Our tracker was generally effective at identifying hands consistently through time, even in frames depicting multiple left and right hands (5.5c, 5.5d). Additionally, we found that we could identify high economy of motion instances from the trajectory maps alone. For example, actions requiring a steady hand such as excising tissue with an electrocautery rendered very little overall tracking movement (cyan in 5.5b, yellow in 5.5c, 5.5d). This was consistent across all four example videos. For techniques that involved larger lateral movements such as tying a knot with suture, trajectories were smooth and efficient (top hand, cyan in 5.5a). Finally, the mapped trajectories also highlight instances of highly controlled dexterity, such as in Figure 5.5d where the bottom left hand relies on minor finger adjustments as opposed to larger motions to apply counter-traction (blue), while the right hand uses electrocautery to divide tissue (yellow). This compares to the larger movements applied by assistants to ensure counter-traction (blue and green trajectories). Without explicit hand identification labels, our detections and tracking algorithm were able to generate coherent hand-specific tracking. Overall, the trajectory maps highlighted insights that were consistent with an independent surgeon review of videos.
<table>
<thead>
<tr>
<th>(a)</th>
<th>Surgeon completing a one-handed tie (cyan) while assistant retracts tissue (blue).</th>
</tr>
</thead>
<tbody>
<tr>
<td>(b)</td>
<td>Surgeon excising adipose tissue using electrocautery (cyan) while assistant repositions tissue.</td>
</tr>
<tr>
<td>(c)</td>
<td>Surgeon excising adipose tissue using electrocautery (yellow) while assistant repositions tissue (cyan).</td>
</tr>
<tr>
<td>(d)</td>
<td>Surgeon dividing muscle tissue using electrocautery (yellow) while assistant adjusts retraction (green and cyan) and surgeon establishes counter-traction (blue).</td>
</tr>
</tbody>
</table>

**Figure 5.5:** Hand tracking during surgery videos. Figure is best viewed in color. Given post-processed predicted detection boxes as tracking algorithm input, we obtain temporally consistent and hand-specific frame-level assignments. Tracking boxes for videos are visualized on top of sequential frames sampled across time (first three columns). The epicenter of each box is used to generate trajectory maps depicting hand motion over time (last column).
Towards automated assessment of open surgeries, we present a CNN-based computer vision model, trained on data of diverse, publicly accessible videos of open surgeries, which achieves strong performance on spatial hand detection in real-world surgeries. To the best of our knowledge, this is the first work to produce an effective surgery hand detector from visual image-level data alone. Finally, we combine our detector with an off-the-shelf object tracking algorithm to enable hand-specific identification and tracking throughout videos, allowing further downstream assessment of crucial surgical properties such as movement patterns and economy of motion on singular hands. Touching upon our original themes, we note that trying to do hand detection in surgery videos provides a dataset domain shift challenge, which motivates moving beyond existing pre-trained detection models and available datasets. At the same time, collecting annotations can be an arduous process, and care must be taken at both the curation and annotation steps to ensure a diverse, representative, and accurate training resource. We hope to extend these capabilities in future work by further increasing the quantity and representation of surgery videos in our dataset, along with the addition of finer joint-specific labels and hand identification annotations.
Automated Discovery of Activities in Surgery Videos

Having previously introduced an algorithm for unsupervised activity discovery and demonstrated hand detection and tracking in surgery videos, we now seek to build on both of these contributions towards unsupervised and fully automated discovery of surgery activities.
In Chapter 4 we set up the problem of unsupervised activity discovery. Our goal was to discern when specific activities happened throughout a series of untrimmed and unlabeled videos. By leveraging the intuition that activities are not only temporally related but also inherently hierarchical, we were able to achieve state-of-the-art performance on activity datasets depicting humans in everyday context. In the process, we introduced both an algorithm with novel sampling and ranking loss methods, as well as an application of hyperbolic space to video embeddings.

While promising, this previous work was limited in its demonstration of real-world open-ended applications. Chapter 5 provided one such application for computer vision, focusing on how to detect and track operating hands in open surgery videos. Motivated by the need for more quantifiable and scalable assessment, we curated a dataset accessibly sourced from YouTube and built an object detector that could automatically identify surgical hands in a variety of videos.

Towards actual surgical assessment however, it is imperative to not only consider the ability to track and detect specific objects of interest, but also exhibit general understanding of activities happening throughout operations. While detecting and quantifying the fine-grained mechanics of a surgeon’s actions is beyond the scope of this work, we now explore if broader activities such as suturing and cutting can automatically be discerned in a video through a deep learning approach. One way to do this would involve similar steps to our work in Chapter 5. After scraping the web for a large number of videos depicting surgeons carrying out various tasks, we could put together another dataset by annotating the timestamps for various activities happening throughout each of the videos. Finally, we could train a state-of-the-art video architecture model in a supervised manner to recognize our aforementioned labeled activities.

However such an approach may be sub-optimal for several reasons. Curating video datasets can take a non-trivial amount of time, where specific attention must be paid to the quality and content of videos. Additionally, the task of annotating the start and stop timestamps of various activities often requires considerable manual labor and is accordingly subject to human error. The context
Figure 6.1: Example frames depicting three surgical actions in the YouTube surgery action recognition dataset.

of surgery may also preclude the possibility that anyone can do this task. Human annotators must first learn to recognize specific operating procedures before they can label the scenes to supervise an algorithm. Creating an annotated dataset here can therefore carry a uniquely high opportunity cost—our target annotators would be surgeons and medical school students, with computer science graduate students and anyone else willing to annotate facing additional learning hurdles to get around in their spare time.

All of these considerations motivate our final contribution, where we apply the algorithms developed in Chapter 4 to surgery videos. Videos were collected in a similar manner to those in Chapter 5, but we demonstrate that by carrying out the same unsupervised learning algorithm in Chapter 4, we can discover various activities without requiring prior annotations across breast, gastrointestinal, and head-and-neck procedures. We thus demonstrate strides in automatic activity discovery and show that from breakfast to surgery, this previous unsupervised activity discovery

6.1 Methods

6.1.1 Youtube Action Recognition Dataset

Since the previous YouTube dataset in Chapter 5 was created with training a supervised hand detection model in mind, we evaluated our unsupervised activity discovery algorithm on videos selected
from an expanded action recognition dataset. While created under the same labeling effort as the previous surgery hands dataset, the expanded YouTube dataset contains additional videos specifically containing various suturing, cutting, and tying actions. Like the previous hand detection dataset, individual videos depict either breast, gastrointestinal, or head-and-neck procedures. In total, the dataset contains over 50 hours of video spread across 340 videos, with about 10 hours of cutting, 2 hours of tying, and 4 hours of suturing procedures (Figure 6.2). Although time-specific ground-truth annotations were available, in our unsupervised activity discovery setting we ignored these during training and evaluation.

### 6.1.2 Frame-wise Feature Extraction with 3D CNNs

Following the video processing procedure described in Section 3.2, we first extracted frame-wise features from our sample videos with a pretrained CNN architecture. For our feature extractor we chose a ResNet-101 architecture with 3D convolutional kernels due to its performance on the Kinetics action recognition dataset\(^\text{13}\). The model combines two innovations for informative feature map generation: 3D convolutions and ResNet blocks. Discussing the former first, 3D CNNs can directly extract both spatial and temporal features from raw video segments, unlike their two dimensional counterparts introduced in Section 2.1.2 which are limited to frame-wise feature generation.\(^\text{68}\)
When trained from scratch, 3D CNNs have been shown to produce results on action recognition datasets like the Kinetics dataset comparable to that of 2D CNNs pretrained on ImageNet, contributing to their appeal as a video architecture feature extractor.

The extractor we use combines 3D convolutions with ResNet, one of the most successful architectures for image classification, and a model we previously introduced in Chapter 2.3 as the backbone for the state-of-the-art RetinaNet object detector. Briefly, the main innovation of ResNet is a “skip connection”. These connections allow the model to skip layers as it calculates a gradient through backpropagation, leading to greater numerical stability and the ability to train very large networks. With extra layers, the ResNet model has an increased capacity to learn a larger variety of nonlinear functions, and we use the popular ResNet-101 architecture with 101 layers.

6.1.3 Hyperbolic Embedding Network

We next fine-tune these 3D ResNet feature maps through our unsupervised learning method to produce final frame-wise embeddings suitable for activity segmentation. Aside from not incorporating the hierarchical dependencies we previously discussed in Chapter 4, the Kinetics-pretrained ResNet feature maps alone may still encounter potential problems with domain shift when dealing with the surgery videos. We accordingly adapt the same two-layer FCN architecture introduced in Chapter 4 to take in the features produced from our 3D ResNet feature extractor. For training, we adopt the same sub-segment sampling technique (Section 4.3.1) and ranking loss (Equation 4.5) used in Chapter 4. Afterwards we proceed with the same clustering and Viterbi decoding procedure as in Section 4.3.3. As a potential limitation, for training we first group sample videos into their larger surgery class (breast, gastrointestinal, head-and-neck), and learn separate embeddings for each surgery class. While we do not use ground-truth annotations to match with our cluster labels, we can still follow the decoding procedure to generate a video-long segmentation. We can then qualitatively compare the timestamps to observe if activities are indeed being discovered.
6.2 Experiments and Results

6.2.1 Implementation Details

For feature extraction, we first use ffmpeg to extract frames from the video MP4 files at the default constant frame rate. We then calculate average features over every 16 consecutive frames using the 3D ResNet-101 feature extractor. To train our hyperbolic embedding network with the hierarchical and temporal ranking loss (Equation 4.5), for every video we sample 4 positive segments per target segment, 16 negatives per positive, and 8 target segments for a total of 512 segments per video. Segment lengths are uniform randomly selected from the possible lengths \{10, 50, 100, 200\}. Finally, we train for 40 epochs with batch size 256, Adam optimizer, and learning rate $10^{-4}$ to produce two-dimensional embeddings for downstream segmentation. Aside from these training parameters, we only need to specify a number of activities that we would like to discover. This corresponds to the number of clusters we fit using the GMM approach described in Section 4.3.3. To produce the actual segmentations, we fit the specified number of activity clusters after learning our embeddings, and proceed with the Viterbi decoding procedure.

6.2.2 Qualitative Activity Discovery

We show that our unsupervised activity discovery algorithm is able to discern separate activities across various surgery videos without prior dataset-specific supervision and annotations. Accordingly, for a set of videos we can effectively discover activities without the need for explicit knowledge on what we are looking for, and again only need to specify the number of activities we would like to discover. In this case, we decode segmentations for 4 and 10 activities, corresponding to different granularities. Although the videos were collected to showcase cutting, tying, and suturing, we also wanted to model the segmentations with a background class, and were further interested in if more
activities could be discovered.

In Figure 6.3 we show that across videos, segmentations at both granularities seem to correspond with changes in the activity depicted in their corresponding videos. Long clips of the same activity, such as suturing in Figures 6.3b and 6.3c, correspond with equally long segmentation bars. Due to the unsupervised nature of our approach, the activities discerned from within the videos also are not strictly limited to the three activities known to exist in our videos. Padding an operating site with gauze shows up as a separate activity in both Figures 6.3d and 6.3e. This suggests an added benefit to unsupervised learning beyond being able to bypass labor-intensive dataset curation and annotation; instead of having to pre-define the activities we would like to recognize, we can discover arbitrary ones from the data alone. Additionally, cutoffs between activities at the four-activity segmentations tend to align with the cutoffs at the ten-activity level. While further analysis is needed to understand the actual types of activities discovered at both these levels, there does seem to be some notion of a hierarchical relationships. We can partially observe this through the clips included at uniform time intervals. For example in Figure 6.3e the gauze application is only discovered at the ten-activity level, while the entire preparation-before-cutting procedure is grouped into a single activity at the four-activity level. We do note that added granularity could contribute to additional noise, so for future work we would like to conduct additional analysis into what should count as an atomic activity. Finally, it is interesting to observe that although the videos depict various surgery actions, they are not trimmed to only include these, and our method also seems to detect these non-surgery actions as well. This includes the non-operating clips in Figure 6.3a, and the title sequence in 6.3e. While still qualitative, these experiments show that unsupervised activity discovery in surgery videos is indeed possible.
Figure 6.3: Unsupervised activity discoveries in surgery videos. We show activity segmentations on sample videos depicting various actions from head-and-neck (a, b), gastrointestinal (c, d), and breast (e) surgeries. Segmentation bar color denotes an activity, but is meaningless beyond recognition of a new activity. Timestamps of the corresponding video frames sampled at uniform intervals are included above. From the learned hyperbolic embeddings, we decoded segmentations to recognize 4 and 10 activities (from $K = 4$ and 10 clusters to discover), and show that across videos unsupervised segmentations seem to correspond with changes in video activity. Segmentations also tend to agree across both granularities, highlighting potential hierarchical relationships between the discovered activities.
6.3 Discussion

Building on our previous contributions for unsupervised activity discovery and the application of deep learning computer vision methods to the surgery domain, we demonstrate the ability to automatically discover surgery actions without prior video-specific knowledge or annotations. In the process we overcome additional challenges such as dealing with multiple surgery types and embedding and segmenting across untrimmed videos. We use the 3D ResNet-101 architecture to extract semantically meaningful features from the raw videos. These features are then fed to our hyperbolic embedding network to learn embeddings specifically useful for activity segmentation, and combined with the rest of the encoding and decoding procedure outlined in Chapter 4 we are able to achieve unsupervised segmentation in the new surgery domain. Our work showcases additional benefits of unsupervised learning. We can recognize activities without having to pre-define or limit the detection to a set of pre-defined actions of interest, and also reduce the need for time-intensive and laborious manual annotations. Finally, we also provide additional support for our previous hypothesis on the importance of both hierarchical and temporal understanding for parsing videos from activities. Segmentations for two different numbers of activities end up aligning with each other, suggesting that various activities do seem to exist at multiple complexities. Although there is ample room for future work, our contributions so far present a promising step towards fully automated activity discovery in real-world settings.
Conclusion

In this work, we sought to expand the capabilities of automated video understanding through both new developments in unsupervised learning as well as new applications to the surgery domain. Leveraging both of these contributions, we finally showcased an application of our unsupervised learning algorithm into the real world, tackling the well-motivated problem of discovering actions in open surgery videos. Along the way we discussed various scenarios where one size may not fit all, where the past success and dominance of pre-existing methods and datasets do not necessarily uni-
versally do well across all problem settings. Our contributions also do not necessarily ascribe to an end-to-end approach popular in the deep learning literature. Finally, in this thesis we tackle both supervised and unsupervised learning tasks, and demonstrate that they can both be hard, albeit in different ways.

7.1 Summary of Contributions

In Chapter 4, we considered the challenging task of unsupervised activity discovery. We sought to discern the start and stop timestamps of various activities in a variety of unlabeled and untrimmed videos. Leveraging the intuition that activities are inherently hierarchical, we developed a novel unsupervised learning algorithm to explicitly extend this understanding to a deep learning architecture. We introduced the use of hyperbolic space to video embeddings, taking advantage of the fact that hyperbolic embeddings are able to encode tree-like hierarchical structures much more efficiently than their Euclidean counterparts. To actually learn such embeddings in the video domain, we proposed a video sampling procedure and temporal-hierarchical loss. Our embeddings were able to beat the previous state-of-the-art in unsupervised activity segmentation, and we followed up on this result with additional analysis into the structure of our embeddings, their ability to capture hierarchical relationships, and their potential to discover activities at multiple complexities.

Following this, in Chapter 5 we tackled the other main direction of this thesis—how to expand the application domains of deep learning video understanding. We consider the well-motivated problem of automated hand detection and tracking in surgery videos, and curate a new dataset that when used to train the deep CNN-based RetinaNet architecture substantially outperforms existing hand detection datasets. The work highlights the important challenge of domain shift that can limit the application of deep learning approaches to various domains, and also showcases the steps required to curate a representative yet diverse labeled dataset to extend state-of-the-art object detection
to operating hand detection in surgery videos. Given these detections, we then demonstrate hand-tracking through videos, which serves as a promising step towards quantifying surgeon activity and building automated surgical assessment tools.

Finally, in Chapter 6 we accomplish unsupervised activity discovery in surgery videos. Having proposed a new state-of-the-art unsupervised algorithm and demonstrated the ability to apply computer vision to the surgical setting, we build on both these developments to learn surgical activity embeddings that can be decoded to discover surgery actions from untrimmed and unlabeled operating videos. We show that the same training procedure, hyperbolic embeddings, and clustering-to-decoding procedure extends beyond the benchmark datasets tackled in Chapter 4, and that we are able to at least qualitatively demonstrate promising results for segmentation. Beyond bypassing the manually intensive dataset curation and annotation process for a similar supervised approach that we undertook in Chapter 5, the unsupervised method proposed here can also discover activities without any prior knowledge on what to exactly recognize or look for. While one limitation of not having explicit labels involves not being able to exactly classify what activities are happening when, the flexibility of being able to recognize when arbitrary activities occur throughout a video is still a major contribution. As a final empirical result, we show that the discovered activities in surgery videos also seem to maintain hierarchical relationships, providing further evidence for hierarchically-motivated video understanding going forth.

7.2 Future Work

Just as our main contributions were motivated by expanding the automated learning capabilities of deep learning video models and extending the real-world domains they could be applied to, future directions of open research also exist along both these dimensions.

While achieving state-of-the-art unsupervised activity segmentation is certainly commendable,
there is still a wide margin for potential improvement. Up to now and including this work, the most
cutting edge methods involve some learning some video frame or clip-wise embeddings with a self-
supervised training task and appropriate loss function, and then clustering such embeddings for de-
coding with a sequential probabilistic model. While their use is well-motivated and supported with
a strong mathematical foundation, we only need to look towards NLP for inspiration on potential
alternatives. Video understanding introduces a temporal component to computer vision that has
naturally been a part of natural language processing. In particular, with enough abstraction activity
segmentation can be reduced to topic modeling over sentences and documents\(^{19,85}\). Accordingly,
just as in recent years neural architectures such as long short-term memory (LSTM) networks\(^{78}\) and
Transformer models\(^{83}\) have eclipsed more probabilistic approaches in NLP, could it be the case that
such deep learning approaches can lead to performance boosts in unsupervised activity understand-
ing for videos?

Finally, much exciting work lies in improving the ability to automatically perceive surgical actions
and events through computer vision. Further work on evaluating the unsupervised surgery activity
segmentations can be done by directly comparing the segmentations to ground-truth video labels.
While more labor intensive, annotating the videos with more activities, as well as activities at differ-
ent levels of surgical complexity, could provide a valuable ground-truth to elucidate what levels of
activities are capable of being captured at different granularities. Expanding the types of surgeries
represented in the dataset as well as the representation of different camera angles, lighting setups,
and other visual characteristics could also improve the robustness and generalizability of surgery
activity embeddings. Lastly, since unsupervised learning methods learn directly from the details pro-
vided inherently in the given data, as they get better we should be able to see direct improvements in
a variety of application domains.
A

Additional Materials
A.1 Additional Qualitative Activity Segmentation

We present additional qualitative results across different Breakfast activity classes, visualizing example temporal activity segmentations from classes not shown in Chapter 4 in Figure A.1. We include the frame-level assignments between groundtruth labels and our model’s output, distinguished by color.

(a) Example for the “cereals” activity. Sub-activities in order are SIL, take_bowl, pour_cereals, pour_milk, SIL.

(b) Example for the “friedegg” activity. Sub-activities in order are SIL, crack_egg, fry_egg, put_egg2plate, SIL.

(c) Example for the “juice” activity. Sub-activities in order are SIL, take plate, take glass, cut orange, take_squeezer, squeeze_orange, pour_juice, SIL.

(d) Example for the “milk” activity. Sub-activities in order are SIL, spoon_powder, pour_milk, stir_milk, spoon_powder, stir_milk, SIL.

(e) Example for the “pancake” activity. Sub-activities in order are SIL, crack_egg, spoon_flour, pour_milk, stir_dough, pour_oil, pour_dough2pan, fry_pancake, take_plate, put_pancake2plate, SIL.

(f) Example for the “salat” activity. Sub-activities in order are SIL, take plate, take bowl, cut fruit, put fruit to bowl, cut fruit, peel fruit, cut fruit, put fruit to bowl, SIL.

Figure A.1: Example unsupervised activity segmentations in the Breakfast dataset.
Table A.1: Nearest unique neighbors with respect to cosine distance for Poincaré embeddings of groundtruth activity segments, with hyperbolic norms included.

<table>
<thead>
<tr>
<th>Nearest Neighbor</th>
<th>Cosine Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>cut_bun (0.50)</td>
<td>peel_fruit (0.52)</td>
</tr>
<tr>
<td>scoop_butter (0.51)</td>
<td>cut_fruit (0.55)</td>
</tr>
<tr>
<td>carry_knife (0.52)</td>
<td>carry_fruit (0.53)</td>
</tr>
<tr>
<td>reach_bread (0.48)</td>
<td>reach_fruit (0.55)</td>
</tr>
<tr>
<td>fry_egg (0.44)</td>
<td>spoon_flour (0.62)</td>
</tr>
<tr>
<td>turnonOff_stove (0.47)</td>
<td>scoop_flour (0.62)</td>
</tr>
<tr>
<td>carry_spatula (0.43)</td>
<td>liftopen_liftFlour (0.63)</td>
</tr>
<tr>
<td>fry_egg (0.47)</td>
<td>carry_flour (0.63)</td>
</tr>
<tr>
<td>pour_oil (0.40)</td>
<td>add_teabag (0.60)</td>
</tr>
<tr>
<td>screwopen_capOil (0.40)</td>
<td>open_teabag (0.56)</td>
</tr>
<tr>
<td>pour_oil (0.40)</td>
<td>reach_teabag (0.70)</td>
</tr>
<tr>
<td>screwclose_capOil (0.40)</td>
<td>reach_cup (0.63)</td>
</tr>
</tbody>
</table>

A.2 Further Multi-level Comparisons

As in Section 4.6.1, we include more nearest neighbors video segments between the “coarse” and “fine” labeling schemas in the Breakfast dataset (Table A.1). In general, the coarse segments are closer to the origin than nearest fine neighbors.

A.3 Surgery Hand Dataset Annotation Tool

Additional screenshots of the interface provided in the annotation tool used to collect data in Chapter 5 are included in Figures A.2 and A.3.
Figure A.2: Surgery annotation tool, bounding box interface.
Surgery Hand Keypoints

Currently annotating hand 2 (right) and keypoint 0

Review

If you're happy with all the keypoints labeled for the current hand, press N to begin another hand, or Enter to submit labels. If you'd like to go back and modify previous annotations, hit Backspace.

Figure A.3: Surgery annotation tool, hand keypoint interface.
References


