UNIVERSITY OF CALIFORNIA SAN DIEGO

Towards solving computer vision problems: datasets, labels, algorithms, and applications

A dissertation submitted in partial satisfaction of the
requirements for the degree of Doctor of Philosophy

in

Computer Science

by

Iljung Samuel Kwak

Committee in charge:

Professor David Kriegman, Chair
Professor Serge Belongie
Professor Julian McAuley
Professor Mohan Trivedi
Professor Zhuowen Tu

2019

The Dissertation of Iljung Samuel Kwak is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

Chair

University of California San Diego

2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

Throughout my graduate career, I have been fortunate enough to have constant support from those around me. I'd like to thank my Ph.D. advisor, David Kriegman, for being a patient and willing mentor. I am especially grateful for all our meetings regardless of time zone differences. I also thank Serge Belongie for his guidance and showing me his enthusiasm towards research. In addition, I thank Serge for allowing me to be a part of his research group, SE(3), during a complex transition period. I thank Kristin Branson for also providing me with mentorship and allowing me to finish my graduate research as a part of her lab.

Next, I'd like to thank my many co-authors and colleagues. In particular, Ana Murillo was my first mentor on my first graduate research project. We spent many strange, but productive, hours working together while she was in Spain and I in San Diego. To Kimberly Wilber, I will always cherish our work together and remember your awe-inspiring spirit towards research and life. Finally, I thank all the members of all the research labs I have been a part of. Although at times I felt lost and distracted, every lab welcomed me as a part of their own. For Kai, Steve, and Oscar for being wonderful senior members of the lab when I first started. To Catherine, Tsung-Yi, and Zak, thanks for being awesome to hang out and discuss research with. To Alice, Roian, and Nakul, thanks for being patient with me and helping with my move to Virginia. And to my friends for being there and making the journey more fun.

Finally, thanks to my family. My parents and brother have been, and continue to be, supportive in all my endeavors. Although I forget to call and keep in touch, they always forgive and contact me. To my brother, Paul, I thank you for always helping me out whenever you were able to. I joke about how you changed professions into a computer related field. But in reality, I couldnt be happier that you found a profession you enjoy and it is a field that we share.

The work proposed in the first chapter was supported by ONR MURI Grant #N00014-08-1-0638 and by projects DPI2012-31781, DPI2012-32100 and DGA T04-FSE. The work described in Chapter 2 was partially supported by an NSF Graduate Research Fellowship award (NSF DGE-1144153), a Google Focused Research award, and AOL-Program for Connected

Experiences. We also wish to thank Laurens van der Maaten and Andreas Veit for insightful discussions.

The work in this dissertation is based on the following publications.

Chapter 2 is based on "From Bikers to Surfers: Visual Recognition of Urban Tribes," I. S. Kwak, A. C. Murillo, P. N. Belhumeur, D. Kriegman, and S. Belongie, British Machine Vision Conference (BMVC) 2013 [KMB+13]. The dissertation author was the primary investigator.

Chapter 3 is based on the following papers: "Cost-effective hits for relative similarity comparisons." M. J. Wilber, I. S. Kwak, and Serge J. Belongie, Second AAAI conference on human computation and crowdsourcing (HCOMP) 2014 [WKB14a]. and "Learning concept embeddings with combined human-machine expertise," M. J. Wilber, I. S. Kwak, D. Kriegman, and S. Belongie, International Conference on Computer Vision (ICCV) 2015 [WKKB15]. The dissertation author was one of two contributing authors of this paper in both algorithm and manuscript development.

Chapter 4 is based on "Detecting the Starting Frame of Actions in Video," I. S. Kwak, D. Kriegman, K. Branson and is currently being prepared for submission for publication of the material. The dissertation author was the primary investigator and author of this paper.

VITA

2005	Bachelor of Science, Carnegie Mellon Universisty
2015	M.S. in Computer Science, University of California, San Diego
2019	Ph.D. in Computer Science, University of California, San Diego

PUBLICATIONS

I. S. Kwak, D. Kriegman, K. Branson, "Detecting the Starting Frame of Actions in Video." *Under review*.

Wilber, M., Kwak, I. S., Kriegman, D., and Belongie, S. "Learning concept embeddings with combined human-machine expertise." In *Proceedings of the IEEE International Conference on Computer Vision*. 2015.

Wilber, Michael J., Iljung S. Kwak, and Serge J. Belongie. "Cost-effective hits for relative similarity comparisons." In *Second AAAI conference on human computation and crowdsourcing*. 2014.

Cao, C., Kwak, I. S., Belongie, S., Kriegman, D., and Ai, H. (2014, July). "Adaptive ranking of facial attractiveness." In *IEEE International Conference on Multimedia and Expo (ICME)*, 2014.

E. Christiansen, I. S. Kwak, S. Belongie, and D. Kriegman. "Face box shape and verification." In *International Symposium on Visual Computing (ISVC)*, 2013.

I. S. Kwak, A. C. Murillo, P. N. Belhumeur, D. Kriegman, and S. Belongie. "From Bikers to Surfers: Visual Recognition of Urban Tribes." In *British Machine Vision Conference (BMVC)*. 2013.

A. C. Murillo, I. S. Kwak, L. Bourdev, D. Kriegman, and S. Belongie. "Urban tribes: Analyzing group photos from a social perspective." In *Computer Vision and Pattern Recognition Workshops (CVPR Workshop)*. 2012.

ABSTRACT OF THE DISSERTATION

Towards solving computer vision problems: datasets, labels, algorithms, and applications

by

Iljung Samuel Kwak

Doctor of Philosophy in Computer Science

University of California San Diego, 2019

Professor David Kriegman, Chair

The solution to a supervised computer vision problem consists of an application, algorithm, input data, and a set of human generated labels. Solving these kinds of tasks involves collecting large quantities of data, collecting appropriate labels, and developing machine vision algorithms tailored to the application. Progress on these problems has often benefited from large scale datasets with high fidelity labels. Successful algorithms display a synergy between application goals and the size and quality of the dataset. This thesis presents work highlighting the importance of each component of a supervised vision task.

First, the problem of automatically classifying groups of people into social categories is introduced. This problem is called Urban Tribe Classification. To tackle this problem, each

individual and the entire group of individuals are modeled. Since this was a newly introduced computer vision problem, a dataset for this task was created. On this dataset, the combined representation of group and individuals outperforms using only the person representations. This model showed promising results for automatic subculture classification.

Second, the problem of creating perceptual embeddings based on human similarity judgements is tackled. This work focuses on triplet similarity comparisons of the form "Is object i more similar to j or k?", which have been useful for computer vision and machine learning applications. Unfortunately, triplet similarity comparisons, like many human labeling efforts, can be prohibitively expensive. This work proposes two techniques for dealing with this obstacle. First, an alternative display for collecting triplets is designed. This display shows a probe image and a grid of query images, allowing the user to collect multiple triplets simultaneously. The display is shown to reduce the cost and time of triplet collection. In addition, higher quality embeddings are created with the improved triplet collection UI. A 10,000-food item dataset of human taste similarity was created using this UI. Second, "SNaCK," a low-dimensional perceptual embedding algorithm that combines human expertise with automatic machine kernels, is introduced. Both parts are complementary: human insight can capture relationships that are not apparent from the object's visual similarity and the machine can help relieve the human from having to exhaustively specify many constraints.

Finally, the precise localization of key frames of an action is explored. This work focuses on detecting the exact starting frame of a behavior, an important task for neuroscience research. To address this problem, a loss designed to penalize extra and missed action start detections over small misalignments. Recurrent neural networks (RNN) are trained to optimize this loss. The model is shown to reduce the number of false positives, an important criteria defined by the neuroscientist. The performance of the model is evaluated on a new dataset, the Mouse Reach Dataset, a large, annotated video dataset of mice performing a sequence of actions. The dataset was created for neuroscience research. On this dataset, the proposed model outperforms related approaches and baseline methods using an unstructured loss.

Chapter 1

Introduction

As technology improves, the amount of data that can be created and collected greatly increases. Because a large portion of this data is in the form of images and videos, computer vision is uniquely positioned to help catalog and explore this data. Automatic analysis of visual data provides an opportunity for collaboration with many fields. In addition to improved data collection, advancements in technology have allowed more powerful algorithms to be created. Supervised vision has benefited from these changes in technology.

Through social media, personal photo albums are often shared publicly online. Within these personal albums, a common type of photo is the group photo. Where a group of friends have their picture taken together in order to remember a shared moment. The work in Chapter 2 focuses on classifying these group photos into social categories. Understanding the social categories that individuals ascribe to can help automated systems suggest similar interests for those individuals. This problem is called Urban Tribe classification, and a new dataset for the task is provided.

As more data is created and collected, it is not always clear what the best annotations for new datasets are. Some datasets are not categorical by nature and sometimes the goal of the dataset is to explore and discover categories. For these tasks, triplet comparisons have been a useful type of label. Unfortunately, large numbers of triplets need to be collected to be useful. The third chapter suggests two techniques for improving triplet collection. First, HCI

improvements for collecting triplets and their impact is explored. Second, human collected triplets are augmented with learned representations.

Precisely detecting the start of actions in video has many useful applications. Within neuroscience research, being able to pinpoint the exact frame a behavior begins can help researchers correlate visible actions with neural recordings. The work in chapter 4 describes an algorithm for detecting the start of behaviors and introduce a dataset for action start detection. This dataset was labeled by experts in neuroscience and is better suited for precise detection of events than existing action detection datasets. A structured loss for classifying single video frames as the start of actions was designed. The proposed loss is competitive on the existing dataset as well as the collected dataset. Accurately detecting action starts will be extremely useful many vision applications and in particular for neuroscience research.

The supervised vision pipeline involves data, labels, algorithms, and applications. The work in this thesis explores this pipeline. The work in chapter 2 designed a new dataset for a new application. Chapter 3 explores improving interactions between researchers and labelers. Finally, chapter 4 explores the entire pipeline by designing algorithms and constructing a dataset more specific to the task.

Chapter 2

Urban Tribe Classification

2.1 Introduction

The popularity of social media has created a massive influx of images. For some social media platforms, such as Snapchat and Instagram, uploading and sharing images is a core part of the user's experience. Facebook alone receives over 300 million photos a day[Arm]. The abundance of social media presents a compelling opportunity to analyze the social identities of individuals captured within images. This points to an excellent opportunity for computer vision to interact with other fields, including marketing strategies and psychological sociology [CBC+10, JGC+10].

At the time of publication of the original paper described in this chapter, there had been major strides in image semantic content analysis (in face, object, scene, and clothing recognition), but algorithms at that time failed to fully capture information from groups of individuals within images. For example in Fig. 2.1, visual searches of groups of people often provided uninspiring results. Rather than matching personal style or social identity, the search provided images with similar global image statistics. The mainstream media had noticed this deficiency in some recent discussions [Car12] and wondered when vision algorithms will catch up to their expectations.

Since the publication of the original paper, semantic analysis of image content has improved greatly. In particular, fashion and group photo analysis have become large fields of research. In the fashion domain, researchers have created algorithms for generating outfit

recommendations from a single article of clothing [MTSVDH15, HWJD17] and retrieval of clothing similar to a query article of clothing [LLQ+16, HKHL+15]. Similarly, in group photo analysis, algorithms for detecting familial relationships [DCSH15] or classifying the type of gathering [SGCC13] have been created. This chapter's work represents early research in both group and style analysis.

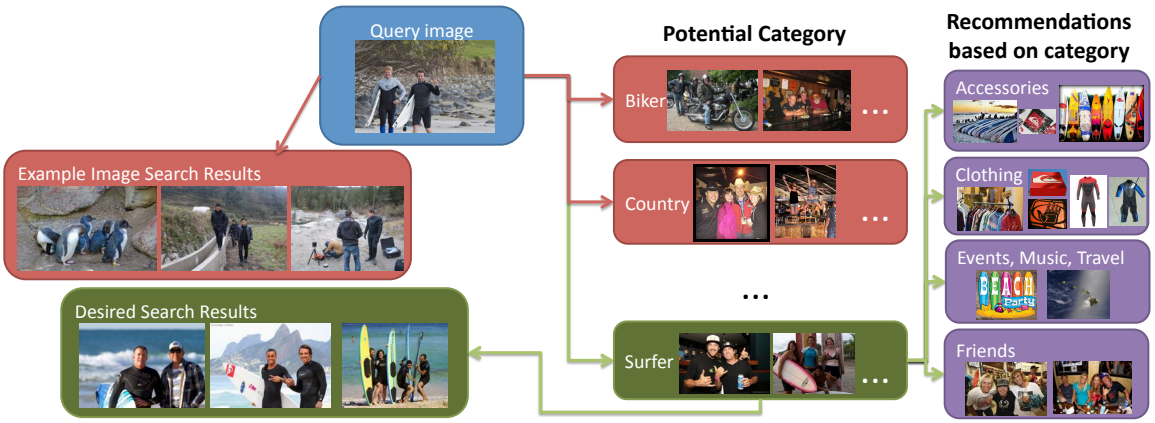


Figure 2.1. The social groups influence the appearance of their members. This work leverages this intuition to classify images of groups of people into social categories. This can improve recommendation systems and user experience with social media, and image search engines can take advantage of this classification and provide more meaningful search results.

In 1985 Michel Maffesoli described *urban tribes* [Maf96] as a group of people who have similar visual appearances, personal style, and ideals. Among tribe members, similar personal styles often manifest as common accessories such as leather jackets or surfboards. The scene context also provides useful information: surfers are more likely to be photographed outdoors by the sea, whereas bikers may congregate at biker bars or be photographed by their bikes on the road. Though not as discriminative, the overall demeanor between tribes can vary as well, such as the laid back smiling surfers versus the frowning dark subculture members. The visual cues shared by members of these tribes provide the basis for this work; members from the same urban tribe are expected to look more similar than members of different tribes, and they can be easily identified by people just from visual information.

Automatic recognition of these urban tribe categories could provide interesting benefits

and applications. More relevant image searches can be conducted; more relevant advertisements can enhance the web experience of both businesses and consumers; social networks can provide better recommendations. Urban tribe classification can also improve surveillance of social demographics. Unfortunately, this categorization problem is difficult because of the ambiguous nature of social categories and the high intraclass variance. Social categories can evolve and fracture into separate groups; individuals may exhibit features of multiple urban tribes or certain individuals may not present a visually salient style at all.

This work highlights the problem of image group categorization from a social perspective, and contributes towards its solution in several ways. Rather than approaching this problem by classifying isolated individuals (as most fashion/style analysis works do), this work calculates meaningful group features and models. Following this idea, a novel recognition pipeline is presented (see Fig. 2.2) and different modeling approaches are evaluated, following common frameworks used in other recognition tasks. Finally, a dataset, the Urban Tribes dataset, is provided. The Urban Tribes dataset has around 100 labeled images per class, from 11 different classes. This dataset is publicly available in order to facilitate further research on social categorization of group pictures[1].

The work described in this chapter is based on "From Bikers to Surfers: Visual Recognition of Urban Tribes," I. S. Kwak, A. C. Murillo, P. N. Belhumeur, D. Kriegman, and S. Belongie, British Machine Vision Conference (BMVC) 2013 [KMB+13].

2.2 Related Work

The work in this chapter attempts to recognize the content of an image from a social perspective, which is a growing area of study [VPB09]. Ding and Yilmaz [DY11] show interesting results for the subjective interpretation of action analysis, proposing how to discriminate positive and negative social relations of individuals in a video sequence. [SWHY11, SLF13] present promising approaches for predicting the occupation of a subject given that individual's

[1] `http://vision.ucsd.edu/content/urban-tribes`

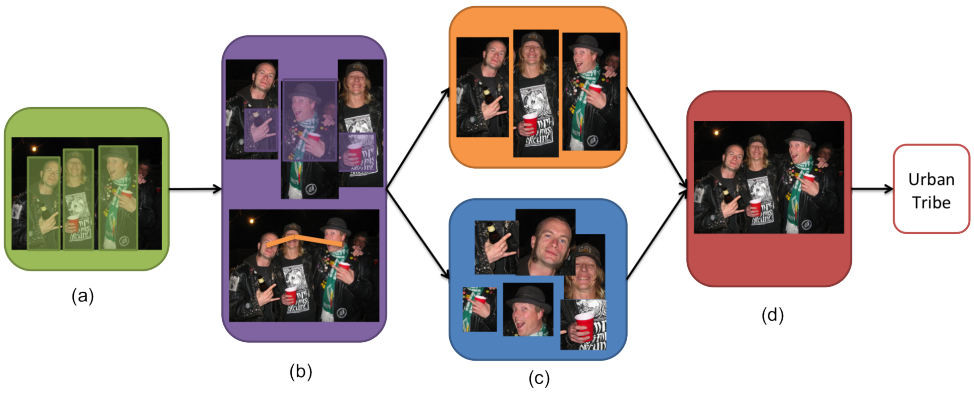


Figure 2.2. Group image categorization. The approach consists of: (a) people detection, (b) local and global descriptor computation, (c) group modeling and (d) classification.

clothing and a rough scene context description. Lee and Grauman [LG11b] present a system for discovering unfamiliar faces in photo albums by leveraging the "social context" of co-occuring people. Closer to the goals and applications of the work in this chapter, Yu *et al.*[YJHL11] analyze a user's photo album and the associated metadata in order to suggest possible social groups of interest, e.g., flickr or Facebook groups about flowers or animals. This is closely related to the goal of analyzing social media, however the approach described in this chapter deals only with visual information and focuses on the analysis of images with groups of people. Finally, [WC15] has improved the classification results introduced in this chapter by using powerful learned feature representations. A common element among most these works is the need for both global image statistics as well as more semantic individual level attributes.

An important aspect of the work in this chapter involves analyzing a group of individuals within an image. Group photos have been applied to a wide range of applications since the publication of the original paper. The group structure or locations of faces has been used to detect social relationships [SSF17, DCSH15, SGCC13, GC09], such as family members or sports teammates. Group analysis has also been shown to improve individual identification [MKBK11]. Dhall *et al.* [DJRG12] highlights the importance of group analysis as a whole, and use it to better understand the mood of the group of people in an image. In most of these works, the target goal is to automatically analyze group photos, a common part of personal photo albums, from a social

context.

Automatic fashion analysis has become a well studied topic [KYBB14, HWJD17, LLQ+16, AHSG17, MTSVDH15, VBK17]. The work in this chapter is most similar to Hipster Wars [KYBB14], where the authors learn the styles of a variety of social classes. Hipster Wars focused on images of individuals and explore both discovery and prediction of fashion styles for social classes. In contrast the work in this chapter focuses on classifying group images into social categories. In addition to style classification, other researchers have focused on analyzing the quality of fashion styles [MTSVDH15, AHSG17, HWJD17]. For example, the authors of [MTSVDH15, HWJD17] can generate outfit recommendations based on an article of clothing.

Often attributes of individuals are used or predicted by socially motivated vision algorithms. Semantic descriptions of clothing or faces are predicted by many algorithms [AHSG17, PG11, LLWT15, LLQ+16], and their use has shown to improve many tasks, such as social group and fashion analysis [SSF17, HWJD17]. In particular, Sun *et al.* [SSF17] argues for a stronger connection with social psychology. Semantic attributes help with interpretability and can be designed with the social domain theory. Han *et al.* [HWJD17] also argue for the importance of attributes for clothing retrieval. Fashion recommendation systems may need to be able to use either image examples or text based input

The work in this chapter takes advantage of techniques for individual person detection and description. At the time, one of the leading methods is that of Bourdev and Malik [BM09], which is based on the detection of person parts named *poselets*. Its effectiveness has been demonstrated for human parsing [WTL11] and recognizing semantic attributes such as hair style or clothing type [BMM11]. Clothing recognition itself has become a growing field [LSL+12, CGG12]. One of these works, [YKOB12], mentions the interesting link between visual appearance and social identity.

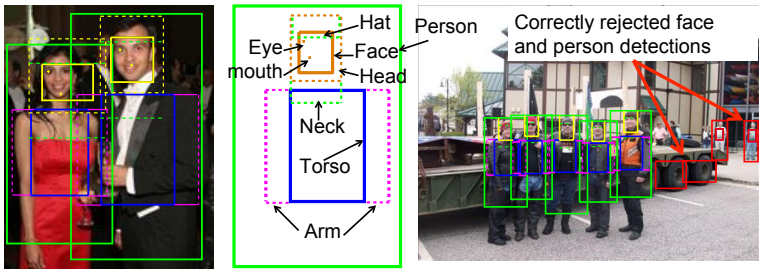


Figure 2.3. Person part detection. Each person hypothesis can have up to six parts and fiducial points for eyes and mouth. The part descriptors are computed within the bounding boxes of these regions. The examples on the left and middle show a close up of the detections. The example on the right presents two face and two person detections that were correctly rejected thanks to the hypothesis construction process proposed (faces must be aligned with a person hypothesis; person bounding box sizes can't have large deviations).

2.3 Group description

The group modeling involves detecting individuals, extracting individual and group features and building the group representation, as key steps for the classification detailed later.

2.3.1 Person detection and description

Individuals are detected within the image and each individual is described as a combination of parts. Similar to [MKB+12], individuals are detected by a combination of poselet based person detection [BM09] and an open source face detector [ZR12]. A detected person is composed of six part bounding boxes: face, head, upper head (hat), neck, torso and arms. Both face and person detections are merged into a single person hypotheses whenever the overlap of the face and body are above a threshold. This simple step filters out many detected people that are in the background, rather than the main scene, as shown in the example in Fig. 2.3. Thus, an image containing p persons is represented by a set of p hypotheses $\{h_1, h_2, \ldots, h_p\}$, and each person hypothesis is composed of a set of parts (not all parts need to be detected to build a hypothesis, as the torso or the arms may not appear in the image). Therefore each individual h_i is represented by a set with the corresponding parts descriptor vector d_{part_name}: $h_i = \{d_{head}, d_{face}, d_{hat}, d_{neck}, d_{torso}, d_{arms}\}$.

The descriptor set is detailed in Table 2.1, which is computed for each part. This set is built on the descriptor set used in [MKB+12].

Table 2.1. Individual descriptors. They are computed within the bounding box of each part.

Ratio of skin pixels vs. total amount of pixels in the patch, obtained with a simple skin-color-based segmentation (normalized to the average face color detected). This descriptor reflects the type of clothing used and how much body is covered with it.
RGB, Luminance and Hue histograms computed for all pixels and computed only for non-skin pixels. This will help modeling the type of clothing used.
Top 3 **dominant values** in Red, Green, Blue, Hue and Luminance color bands. Dominant colors in clothes and accessories are very specific at some social categories.
HoG features [DT05], which will help capture the different poses.

2.3.2 Global group descriptors

In order to account for context and group properties, the image is represented by each of the individuals h_i, as explained above, together with a global group descriptor set, d_{global}: $G = \{h_1, h_2, \ldots h_p, \quad d_{global}\}$. The global descriptors are split in two sets, low level descriptors (detailed in Table 2.2) and high level descriptors (detailed in Table 2.3).

Table 2.2. Low level group descriptors. These are computed over all image pixels and comprise the scene general information, such as lighting and color.

Ratio of pixels within the detected **person bounding boxes** vs. total amount of pixels.
RGB, Luminance and Hue histograms computed on all pixels, on pixels out of the detected person bounding boxes, i.e., background pixels.
Gist [OT01] and **HOG** [DT05] descriptors.

Table 2.3. High level group descriptors. These represent higher level semantic information and are based on the distribution and pose of the detected persons in the group.

Proximity between individuals in the image. A histogram of distances between faces are computed and the average ratio of overlap between person bounding boxes.
Alignment or **pose of the group**. The average angle between a face and its neighboring ones according to a minimum spanning tree computed on the detected faces as proposed in [GC09] is computed.
Scene **layout** of individuals. A histogram of face locations within the image, using a coarse image grid, is computed.

2.4 Group classification

To classify a group of individuals into a social category, the features computed for each person hypothesis and the group features are modeled jointly. This section describes two different approaches studied. The group model is built upon the model proposed in past work [MKB+12]. The group is modeled as a set of individuals, combining their responses in a hierarchy of SVM classifiers. Note that as described, a person hypothesis may have different number of parts detected, which requires a careful classification framework able to deal with heterogeneous descriptor sizes.

Bag of Parts-based classification.

Using the Bag of Parts model to represent a group of people, a bag of m people parts are created and combined with a global descriptor vector d_{global}. The combined group model will be referred to as $G = \{p_1, \ldots, p_m, d_{global}\}$. This model combines all visible parts and the group description. This approach will be named as BoP_k, where k is the size of the vocabulary used. The vocabulary is built for each part type, by running k-means clustering for each part. This vocabulary built for each part type and referred as $V_{part} = \{w_1, ..., w_k\}$. $w_k \rightarrow hist_{wk} = [count_{L_1}, ..., count_{Lj}]$ is the histogram of frequencies for each word in each possible class L.

A signature of the image for each part is created for each vocabulary V_{part}: $hist_{part} = [count_{w1}...count_{wk}]$, where $w_1, \ldots, w_k$ are the words from the corresponding V_{part}. To be able to deal with missing parts, each part type is evaluated separately, and later the distances are combined for each part type into $d_{BoP}(G, L_j)$. The distance from each part type p detected in the image to the corresponding class is weighted by its frequency in the training:

$$d_{BoP}(p, L_j) = 1 - \frac{\sum_{i=1}^{k}(count_{wi} \times hist_{wi}(j))}{k}. \tag{2.1}$$

There are usually several parts of each type in a group image (several faces, arms), and the BoP models somehow how many occurrences of each possible part (i.e. part word) happen in the group. It's not possible to model the group descriptors in a similar way, because there is only one

per image. Therefore, the nearest neighbor between each reference, L_j, and test global descriptor is used:

$$d_{global}(G, L_j) = min_{j=1}^{t}(\, |g_i, g_j| \,), \tag{2.2}$$

where t is the number of training images in class L_j. This distance is normalized between $[0, 1]$ to allow for easy combination with d_{BoP}, which is also normalized. Then, with c possible labels, the label of the group using BoP_k is calculated as:

$$L = \quad \underset{j \in [1...c]}{\arg\min} \, (d_{BoP}(G, L_j) + d_{global}(G, L_j)). \tag{2.3}$$

SVM-based classification.

Alternatively, the group G is represented as a set of persons and the problem is modeled as finding the most likely class C given a particular group image G, i.e., estimating $P(C|G)$ for each possible class. In this case, each person hypothesis has a probability for each class and the final class estimation is a combination of all of them. In this setting, LIBSVM [CL11] is used to train a multi-class SVM on the person hypotheses and the global descriptors. LIBSVM's built in function to calculate probabilities for each class is used. More formally, The goal is to calculate $P(C|G) = P(C|h_1, h_2, \ldots, h_p, d_{global})$, and then the final label L assigned to the query group as follows:

$$L = \quad \underset{j \in [1...c]}{\arg\max} \, P(C = j|h_1, h_2, \ldots, h_p, d_{global}). \tag{2.4}$$

$P(C|G)$ is estimated in the following ways:

$$P(C|G) = P(C|d_{global}) \prod_{i=1}^{p} P(C|h_i) \qquad\qquad P(C|G) = \prod_{i=1}^{p} P(C|h_i \, , d_{global}) \tag{2.5}$$

Figure 2.4. Examples of social groups in the Urban Tribes dataset. The images show a wide range of inter-class and intra-class variability. More details can be seen at the dataset website.

2.5 Experiments and Results

This section evaluates the performance of the proposed algorithms.

2.5.1 Urban Tribes dataset

Creating the Urban Tribe Dataset posed an interesting challenge. Similar to computer vision problems, such as clothing or beauty evaluation, urban tribe categories can be ambiguous and subjective. This is a contrast to other classification problems involving people, where accurate descriptions of each class and its standard appearance can be found, such as age, gender or occupation evaluation. In order to obtain an unbiased dataset, the classes labels are provided by Wikipedia. Eight categories from their list of subcultures[2] to facilitate image collection were selected. In addition to these social groups, three other classes corresponding to typical social venues (formal events, dance clubs and casual pubs) were added. These classes are intended to include some of the most common social event pictures that may not belong to a clear subculture, but still present common appearances in clothing style.

[2] `http://en.wikipedia.org/wiki/List_of_subcultures`

Table 2.4. Summary of Urban Tribes dataset.

Label	# images	# people	Label	# images	# people	Label	# images	# people
biker	114	443	hip-hop	90	253	club	100	365
country	107	347	hipster	102	288	formal	103	414
goth	99	226	raver	116	305	casual/pub	125	459
heavy-metal	102	266	surfer	100	333			

For each of the selected classes, images of groups of people were searched with different search engines. Group labels were used as search keywords combined with location and event keywords such as *bar*, *venue*, *club* or *concert*. Example search terms include 'bikers' and 'biker bar'. The dataset contains a broad range of scenarios for each class, both indoor and outdoor venues, large group pictures acquired from the distance and close-up images, etc. As shown in Fig. 2.4, the groups show a variety of realistic conditions and most classes present high intra-class variation in appearance. Table 2.4 shows the class labels as well as the number of images (# images) per class and total amount of detected persons for each class (# people). Although the number of images per class was balanced, the number of detected persons per image was different depending on the group.

2.5.2 Social group recognition experiments

This section evaluates the performance of the proposed algorithms and the most promising paths towards this novel problem framework. Training and testing images were randomly selected from each of the categories for 50 different iterations. Note that for the 11 classes in the Urban Tribes Dataset, chance classification is $\frac{1}{11} = 0.09$. For each experiment, a fixed number of the images from each class are used for learning the models, and the rest of images are used for testing. A test is considered correct if the most likely group label is correct according to the ground truth labels.

As explained in section 2.4, the Bag of Parts modeling builds a visual vocabulary for each part using the training set, with k visual words per vocabulary. After evaluating different values of k, $k = 30$ (BoP_{30k}) for the rest of experiments, because the performance increased significantly

when increasing k until $k = 30$. The other approach modeled the group as a set of people and used the training set descriptors to train several SVM classifiers. Different options were evaluted: 1) SVM_1, training a single SVM with all the descriptors of each person concatenated, including null values for non-detected parts and replicating the same global descriptor for all hypothesis from a particular image; 2) SVM_2, training one SVM for all part descriptors similarly concatenated and a second SVM for the global image descriptors; 3) SVM_8, training a separated SVM classifier for each part descriptor set and an additional SVM for the global image descriptors. The responses from all the SVMs in each case are simply combined, providing a final probability of each image being of a particular class. The option SVM_1 provided significantly lower performance than the rest during the preliminary tests, therefore results for configuration are not provided for the rest of experiments.

Results from [WC15] are also provided, which were computed after the publication of the original paper. [WC15] create a model called Net_{SDense}, which uses features from an AlexNet [KSH12] network pre-trained on ImageNet [RDS+14]. Net_{SDense} consists of two AlexNets. One for person crop classification and the other for scene image classification. A person and scene SVM classifier uses the feature representations from AlexNet to compute probabilities for an input to belong to a class. The person class probabilities are averaged and weighted by the scene class probability.

Table 2.5 shows a summary of recognition experiments with different amount of training data and different amount of parts used in the modeling, given the most suitable configurations found for each modeling option considered (BoP_{30k}, SVM_2 and SVM_8). Column $allParts + global$ shows the accuracy when combining all person parts and global descriptors; $allParts$ shows the accuracy when combining only person parts; $global(scene)$ column shows the results if only the global descriptors are used).

The last columns show additional baseline results: *individual* shows the average accuracy when each person is classified independently from the rest of the group, i.e., there is no consensus from the group nor group global descriptors used at all. $face + head + global$ shows the results

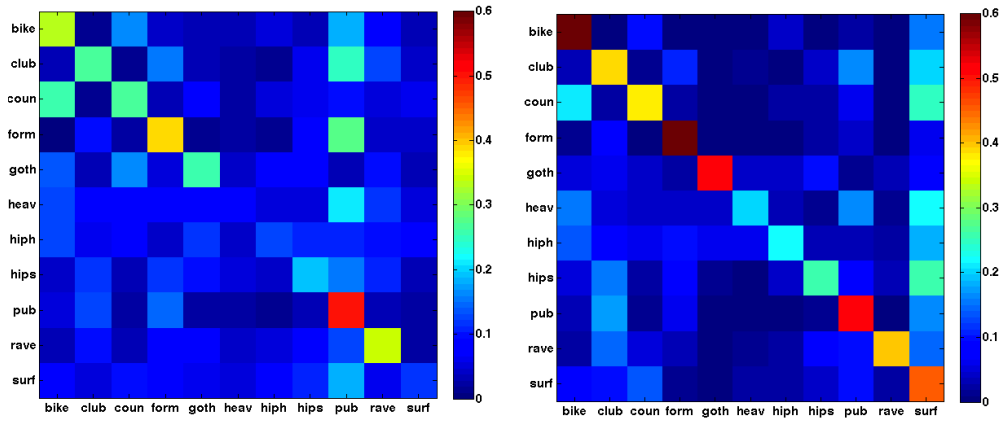
Table 2.5. Average accuracy for the recognition of all classes using different approaches.

Approach	$allParts + global$ (std)	$allParts$	$global(scene)$	$individual$	$face + head + global$
80 random train images per class, 50 iterations, 278 tests per iteration					
SVM_2*	0.43 (0.04)	0.40	0.37	0.34	-
SVM_8	0.46 (0.02)	0.40	0.37	0.38	-
BoP_{30k}	0.37 (0.02)	0.36	0.18	-	0.30
Net_{SDense}[WC15]	0.71(0.004)	-	0.67(0005)	-	-
40 random train images per class, 50 iterations, 718 tests per iteration					
SVM_2*	0.38 (0.03)	0.38	0.31	0.33	-
SVM_8	0.41 (0.01)	0.36	0.32	0.35	-
BoP_{30k}	0.33 (0.02)	0.33	0.17	-	0.25
* SVM_2 does not include arm parts because all part descriptors concatenated to train a single SVM was performing significantly worse. This behavior was not observed for the rest of approaches.					

Average accuracy for each type of descriptor considered separately (80 train images per class)								
descriptors used:	d_{face}	d_{head}	d_{torso}	d_{hat}	d_{neck}	d_{armL}	d_{armR}	d_{global}
BoP_{30k}	0.24	0.24	0.22	0.26	0.21	0.21	0.2	0.18

using the Bag of Parts configuration used in the referred previous work [MKB+12]. The last rows in the table show the contribution of each type of descriptor separately. This analysis is shown for the BoP approach, since it provides the classification result per part. For all modeling options, even just one set of descriptors was able to classify the images above chance, but the final classification scores are clearly improved when all the parts from all the individuals in the image are combined. Particularly interesting is the increase due to the use of global group information. Additionally, we have noted that the global descriptors classifier would correctly predict the correct label between 10% to 15% of the tests (depending on the modeling option). This supports the idea that group and context provide complementary information to person local parts.

Among the models described in 2.4, the SVM based classification provided the best results at the time. It is probably better at learning which components of each descriptor set are more discriminative. We experimented with a reduced set of attributes for BoP approaches did not improve the performance. From the confusion matrices shown in Fig. 2.5, interesting hints for future improvements can be seen, such as the clear confusion in both matrices between *bikers* and *country* (columns 1 and 3) in both directions. This can point to the need for additional



(a) (b)

Figure 2.5. Confusion matrix for classification results obtained with (a) BoP_{30k} and (b) SVM_8, using 80% of the data for training. The rows show results in alphabetical order of the labels (detailed in Table 2.4, from top to bottom. To enhance contrast the color scale is set to $[0, 0.6]$.

Figure 2.6. Examples of classification results. These images have been classified as *hipsters* in all their tests. The two left images are correct, but the label for the right two images should be *goth*.

descriptors or attributes. Analyzing the results, all images have been included, in average, in the test set of 12 experiments. Figure 2.6 shows some sample classification results.

The Net_{SDense} model, benefiting from powerful learned representations, out performs all results by a significant margin. The authors of [WC15] also report classification results on just each person, 0.47 and all the people, 0.67 in a scene. These results aren't included in Table 2.5 because the human crops are not from poselets. Interestingly, these results match the findings that the best results are from the combined representation of the group as well as the scene.

2.6 Conclusions and Future Work

An individual's social identity can be defined by the individual's association with various social groups. Often these social groups influence the individual's visual appearance. An exhaustive baseline analysis is provided for the task as well as a dataset to aid future research. The task introduced in this work opens opportunities for computer vision to improve targeted advertising and social monitoring and provide more tailored experiences with social media.

The work in this chapter has shown that group representations are powerful. Although more recent learned representations have outperformed past results, the recent results do not fully capture the group structure of the individuals in an image. Future work should learn group structure features. For example, Graph Neural Networks [ZCZ+18] could be used to learn a representation for the group. With a graph neural network, detected individuals could be represented by nodes in a graph and the network could model the structure as a whole. Alternatively, the Transformer architecture [VSP+17] has shown promising results in natural language parsing and could also be used to model the group structure. Here the representation of individuals could be passed in as a sequence. In either case the entire group is an input to a network architecture in order to learn group features.

2.7 Acknowledgments

Chapter 2 is based on "From Bikers to Surfers: Visual Recognition of Urban Tribes," I. S. Kwak, A. C. Murillo, P. N. Belhumeur, D. Kriegman, and S. Belongie, British Machine Vision Conference (BMVC) 2013 [KMB+13]. The dissertation author was the primary investigator.

Chapter 3

Collecting & Using Human Judgements of Similarity

3.1 Introduction

Rankings and similarity constraints provided by humans have been used in many fields, such as psychometrics, social sciences, and computer vision. A common application for these constraints is to construct embeddings for visualization and exploration. The overall goal of the work in described in this chapter is to generate these perceptual embeddings. Many researchers use similar embeddings to enhance the performance of classifiers [SKP15, BHW+14, WHB+14a], build retrieval systems [VdMW12b, McF12], and create visualizations that help experts better understand high-dimensional spaces [DBH14, DSK+14]. This problem is tackled by improving the collection of similarity constraints and by combining human similarity constraints with learned representations.

The work in this chapter focuses on collecting and using the triplet constraint. Triplet constraints are of the form, "Is object i more similar to object j or object k?". These constraints have been used in many vision applications, such as face recognition [SKP15] and image retrieval [FSSM07]. Additionally triplets have been shown to be a more reliable form of human similarity constraint [Ken48] than human based rankings or human pairwise similarity constraints. By combining learned representations and human generated triplets, perceptual embeddings can be created for the visual exploration of data.

Unfortunately, asking experts to exhaustively and authoritatively annotate a dataset is not always possible [BEK+12]. Additionally, triplet based comparisons can potentially have O(n^3) complexity [TLB+11]. Hiring actual domain experts is often out of the question, and even crowdsourcing websites such as Mechanical Turk can be prohibitively expensive. Intelligently sampling comparisons can help alleviate this issue [TLB+11, JN11], but the number of constraints to collect can still be too large for human annotation. To reduce the cost of collecting triplets, a new user interface and query was designed and evaluated.

This chapter describes methods for improving the creation of perceptual embeddings from triplet constraints. In 3.3, UI guidelines for collecting similarity comparisons are described, with insights on how to manage the trade-off between user burden, embedding quality, and cost. The method's effectiveness is shown on a series of synthetic and human-powered experiments. In 3.4, an algorithm for creating perceptual embeddings called SNE-and-Crowd-Kernel Embedding, SNaCK, is presented. The algorithm combines expert triplet hints with machine assistance to efficiently generate concept embeddings. The effectiveness of SNaCK embeddings are shown on tasks such as visualization, concept labeling, and perceptual organization.

The SNaCK algorithm performs dimensionality reduction in order to create visualizations. Consider N objects, $X = \{x_i\}_{i=1}^N$ and $x_i \in \mathscr{R}^D$. The goal of dimensionality reduction is to produce a target d-dimensional embedding $Y = \{y_i\}_{i=1}^N, y_i \in \mathscr{R}^d$, where $d < D$. A classic technique for creating low dimensional visualization is metric Multidimensional Scaling (MDS) [Wic03, KW78]. MDS creates the embedding, Y, by attempting to preserve distances between points in the original space by optimizing:

$$\min_Y \sum_{i \neq j = 0,1,\ldots,N} (d_e(x_i,x_j) - d_e(y_i,y_j))^2 \tag{3.1}$$

where d_e is the Euclidean distance. Isomap [BS02] extends MDS by replacing $d_e(x_i,x_j)$ with $d_g(x_i,x_j)$, where d_g is the geodesic distance between points. This change helps Isomap to preserve local structure in the lower dimensional representation. t-Distributed Stochastic

Neighbor Embedding (t-STE) [VdMH08], which will be described in detail in Section 3.4.1, also follows the intuition of MDS, but instead converts distances to conditional probabilities. t-STE attempts to match the probability distribution of the original and target spaces. The probability represents the likelihood that a point chooses another as its neighbor. Like Isomap, this helps t-SNE preserve the local structure in the embedding. An alternative to metric MDS is non-metric MDS (NMDS) [Kru64]. Non-metric MDS optimizes an alternative cost:

$$\min_{Y} \frac{\sum_{i \neq j=0,1,\ldots,N} (d_r(x_i,x_j) - d_e(y_i,y_j))^2}{\sum_{i \neq j=0,1,\ldots,N} d_e(y_i,y_j)} \tag{3.2}$$

where d_r is any monotonic function, but it typically computes the rank order of pairwise distances. This formulation is interesting because it no longer requires knowing the exact distances between points in the original space, which is useful when requesting similarities from humans. Rather than requiring the full rank ordering of points, generalized non-metric MDS (GNMDS) [AWC+07] focuses on pairwise comparisons of similarities $d_e(x_i,x_j)^2 < d_e(x_k,x_l)^2$. When collecting data from humans, this formulation is even more appealing than NMDS and MDS. Stochastic Triplet Embedding (t-STE) [VdMW12b], described in detail in Section 3.4.1, is another algorithm for embedding points using relative comparisons. t-STE converts the pairwise comparisons into probabilities and creates an embedding that matches the probability distribution of the original space. The SNaCK algorithm proposed in this chapter combines t-SNE and t-STE.

The work described in this chapter based on the following papers: "Cost-effective hits for relative similarity comparisons." M. J. Wilber, I. S. Kwak, and Serge J. Belongie, Second AAAI conference on human computation and crowdsourcing (HCOMP) 2014 [WKB14a]. and "Learning concept embeddings with combined human-machine expertise," M. J. Wilber, I. S. Kwak, D. Kriegman, and S. Belongie, International Conference on Computer Vision (ICCV) 2015 [WKKB15].

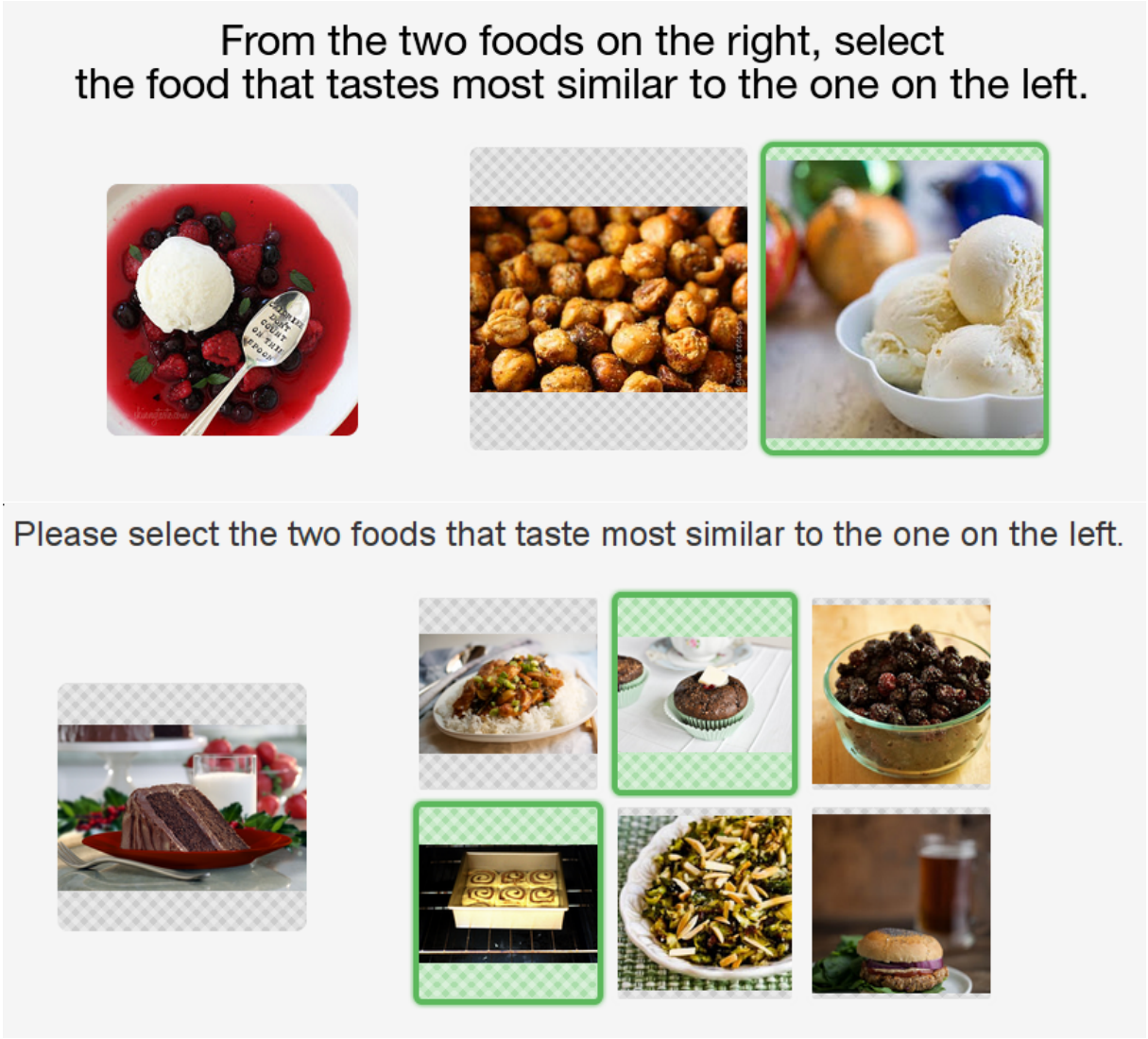


Figure 3.1. Questions of the form "Is object i more similar to j or object k?" have been shown to be a useful way of collecting similarity comparisons from crowd workers. Traditionally these comparisons, or triplets, would be collected with a UI shown at the top. Here, triplets are collected using a grid of images and ask the user to select the two most similar tasting foods.

3.2 Related Work

Perceptual embeddings have been used for a wide range of applications. In [AWC+07], the authors created a two-dimensional embedding where one axis represented the brightness of an object, and the other axis represented the glossiness of an object. [vdMW12a, McF12] construct an embedding based on musical artist similarity. The goal is to use triplets to collect and construct perceptual similarity embeddings. And as mentioned before, SNaCK combines aspects of both t-Distributed Stochastic Neighbor Embedding (t-SNE, from [VdMH08]) and Stochastic Triplet Embedding (t-STE, from [VdMW12b]).

The work in this chapter focuses on using triplet similarity constraints. In addition to being useful for constructing perceptual embeddings [vdMW12a, FG09, GWKP11, McF12], triplets have been used for classification tasks as well. In this case, triplets can be automatically generated using categorical labels, where "object i is more similar to object j than object k", if i and j are in the same class and i and k are not in the same class. [SKP15] showed state of the art face recognition results and [HBL17, CGZ+16] has shown that the re-identification problem can also take advantage of triplet similarity constraints.

Alternative similarity constraints such as pairwise or rank ordering have also been used to create perceptual embeddings. However when collecting these constraints from humans, it has been shown that triplet constraints have been most consistent [Mil56, Ken48, DBH14]. In an experiment comparing the speed and effectiveness of pairwise, triplet, and spatial arrangement embeddings, [DBH14] found that triplet comparisons yield the least variance of human perceptual similarity judgments than other methods. Unfortunately, triplet embeddings can require $O(n^3)$ constraints to be uniquely specified [KvL14], even though many triplets are strongly correlated and do not contribute much to the overall structure [SKP15].

A variety of methods have focused on reducing the number of triplets to collect [TLB+11, JN11]. These algorithms focus on collecting triplets one at a time, but sampling the best triplets first. The idea behind these systems is that the bulk of the information in the embedding can be

captured with a very small number of triplets, since most triplets convey redundant information. For instance, Crowd Kernel Learning [TLB+11] considers each triplet individually, modeling the information gain learned from that triplet as a probability distribution over embedding space. This chapter includes a new UI design to improve the speed of triplet collection and leverages learned representations to reduce the number of triplets required. Since the publication of the UI design, a few works [LHLF15, PMB18] have used the proposed design for collecting triplets. Others have altered the initial design to collect alternative types of similarity constraints [VMN+16].

3.3 Cost Effective Hits

Instead of asking "Is object i more similar to object j or object k?", humans are presented with a probe image and ask "Mark k images that are most similar to the probe," as in Fig. 3.1. This way, with a grid of size n, a human can generate $k \cdot (n-k)$ triplets per task unit. This kind of query allows researchers to collect more triplets with a single screen. It allows crowd workers to avoid having to wait for multiple screens to load, especially in cases where one or more of the images in the queried triplets do not change. This also allows crowd workers to benefit from the parallelism in the low-level human visual system [Wol94]. Because collecting triplets require human effort, the right way to measure the embedding quality is with respect to human cost rather than the number of triplets. This human cost is related to the time it takes crowd workers to complete a task and the pay rate of a completed task. Some authors [WHB+14b, TLB+11] already incorporate these ideas into their work but do not quantify the improvement. The goal is to formalize their intuitive notions into hard guidelines.

It is important to note that the distribution of grid triplets is not uniformly random, even when the grid entries are selected randomly and provided perfect answers. No authors that use grids acknowledge this potential bias even though it deteriorates quality of the collection of triplets, as will be shown in the experiments. Figure 3.2 shows a histogram of how many times each object occurs in triplet answers. 59520 triplets were collected for both histograms. Each

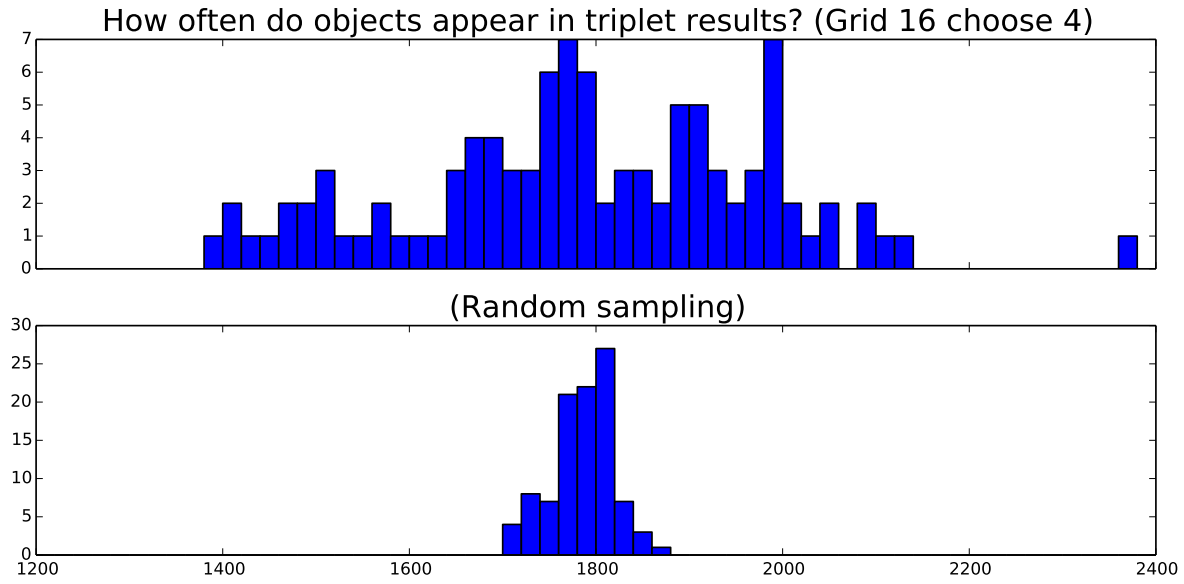


Figure 3.2. Random triplets have a different distribution than grid triplets. The top histogram shows the occurrences of each object within human answers for "Grid 16 choose 4" triplets, the bottom is a histogram of sampling random triplets individually. The variation when using grid triplets (top) is much wider than the variation when sampling triplets uniformly.

object occurs in the answers about $\hat{\mu} = 1785$ times, but the variation when using grid triplets, top histogram, is much wider ($\hat{\sigma} \approx 187.0$) than the variation when sampling triplets uniformly (bottom histogram, $\hat{\sigma} = 35.5$). This effect is not recognized in the literature by authors who use grids to collect triplets. When using grid sampling, some objects can occur far more often than others, suggesting that the quality of certain objects' placement within the recovered embedding may be better than others. The effect is less pronounced in random triplets, where objects appear with roughly equal frequency. This observation is important to keep in mind because the unequal distribution influences the result.

3.3.1 Synthetic Experiments

The goal of the synthetic experiments is to answer two questions: Are the triplets acquired from a grid of lower quality than triplets acquired one by one? Second, even if grid triplets are lower quality, does their quantity outweigh that effect? To find out, synthetic "Mechanical Turk-like" experiments were run with synthetic workers. For each question, a probe and a grid of n

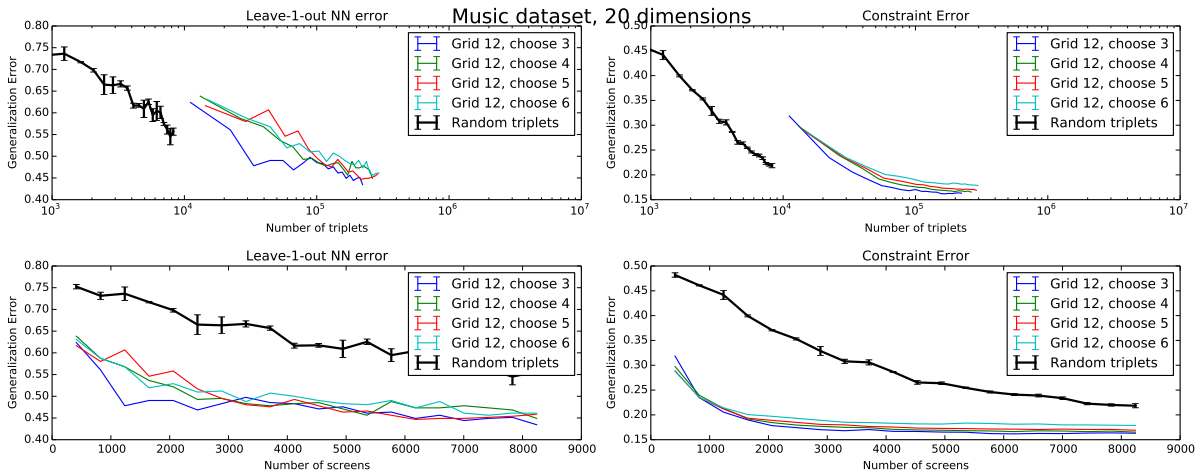


Figure 3.3. When the embedding quality is viewed as the number of triplets gathered (top two graphs), it appears that sampling random triplets one at a time yields a better embedding. However, when viewed as a function of human effort, grid triplets create embeddings that converge much faster than individually sampled triplets. See text for details.

objects are shown. The synthetic workers use Euclidean distance within a groundtruth embedding to choose k grid choices that are most similar to the probe. As a baseline, triplet comparisons are randomly sampled from the groundtruth embedding using the same Euclidean distance metric. After collecting the test triplets, a query embedding is built with t-STE [VdMW12b] and compared to the groundtruth. This way, the quality of the embedding with respect to the total amount of human effort is measured, which is the number of worker tasks. This is not a perfect proxy for human behavior, but it does let us validate the approach, and should be considered in conjunction with the actual human experiments that are described later.

Dataset. UI paradigm is evaluated on the music similarity dataset from [VdMW12b]. The dataset contains 9,107 human-supplied triplets for 412 artists.

Metrics. The quality of the embedding is evaluated using two metrics from [vdMW12a]: Triplet Generalization Error, which counts the fraction of the groundtruth embedding's triplet constraints that are violated by the recovered embedding; and Leave-One-Out Nearest Neighbor error, which measures the percentage of points that share a category label with their closest neighbor within the recovered embedding. As pointed out by [VdMW12b], these metrics measure

different things: Triplet Generalization Error measures the triplet generator UI's ability to generalize to unseen constraints, while NN Leave-One-Out error reveals how well the embedding models the (hidden) human perceptual similarity distance function. These metrics test the impact that different UIs have on embedding quality.

Results. The experiments show that even though triplets acquired via the grid converge faster than random triplets, each individual grid triplet is of lower quality than an individual random triplet. Figure 3.3 shows how the music dataset embedding quality converges with respect to the number of triplets. If triplets are sampled one at a time (top two graphs), random triplets converge much faster on both quality metrics than triplets acquired via grid questions. However, this metric does not reveal the full story because grid triplets can acquire several triplets at once. When viewed with respect to the number of screens (human task units), as in the bottom two graphs in Figure 3.3, the grid triplets can converge far faster than random with respect to the total amount of human work. This implies that "quality of the embedding wrt. number of triplets" can be the wrong metric to optimize. A researcher who only considers the inferior performance of grid triplets on the per-triplet metric will prefer sampling triplets individually, but they could achieve much better accuracy using grid sampling even in spite of the reduced quality of each individual triplet. In other words, efficient collection UIs are better than random sampling, even though each triplet gathered using such UIs does not contain as much information.

3.3.2 Human Experiments

To verify that the UI modifications build better embeddings, Mechanical Turk experiments were run on a set of 100 food images sourced from Yummly[1] recipes with no groundtruth. The images were filtered so that each image contained roughly one entrée. For example, images of sandwiches with soups were avoided. Example images are shown in Fig. 3.4. For each experiment, the same amount of money was allocated for each hit, allowing the embedding quality with respect to cost to be quantified. This dataset and the human annotations are available for

[1] https://www.yummly.com

Figure 3.4. Example images from the dataset. The images in the dataset span a wide range of foods and imaging conditions. The dataset as well as the collected triplets will be made available upon publication.

download at the companion website, https://vision.cornell.edu/se3/projects/cost-effective-hits/.

Design. For each task, a random probe and a grid of n random foods are shown. The user is asked to select the k objects that "taste most similar" to the probe. n is varied across $(4, 8, 12, 16)$ and k is varied across $(1, 2, 4)$. Three independent repetitions of each experiment were run. Each HIT paid \$0.10, which includes 8 usable grid screens and 2 catch trials. To evaluate the quality of the embedding returned by each grid size, the same "Triplet Generalization Error" was used as in the synthetic experiments: all triplets from all grid sizes are gathered and a reference embedding via t-STE is constructed. Then, to evaluate a set of triplets, a target embedding is constructed, and the number of reference embedding's constraints violated by the target embedding are counted. Varying the number of HITs shows how fast the embedding's quality converges.

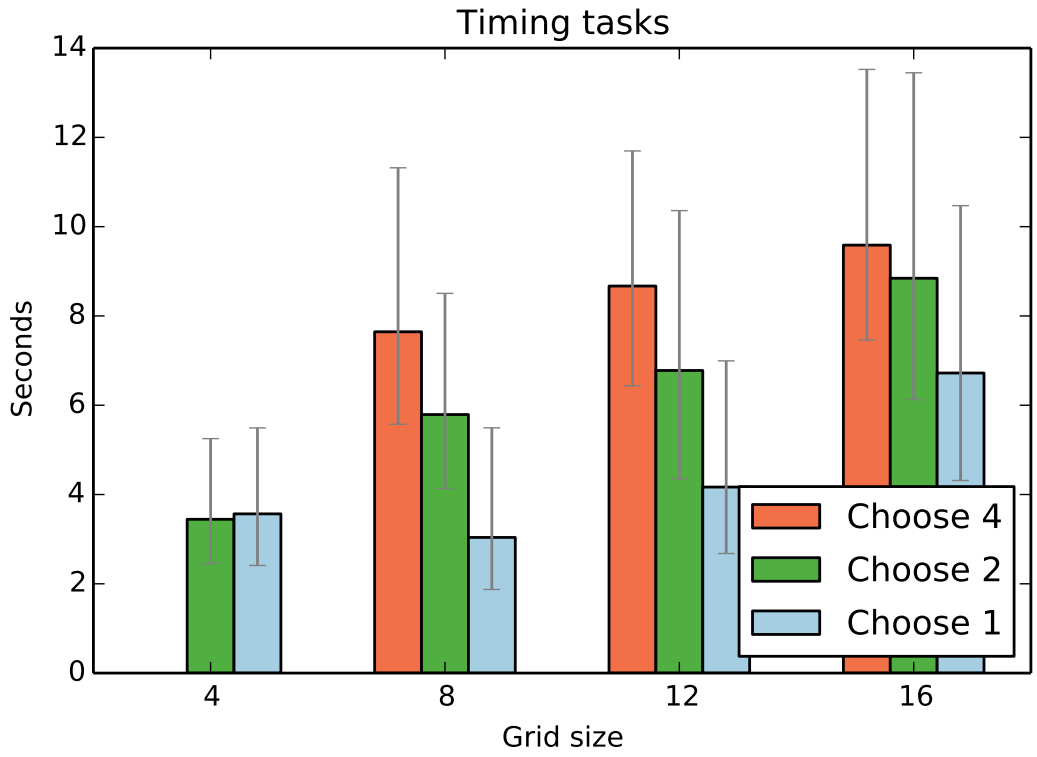


Figure 3.5. The median time that it takes a human to answer one grid is shown. The time per each task increases with a higher grid size (more time spent looking at the results) and with a higher required number of near answers (which means more clicks per task). Error bars are 25 and 75-percentile.

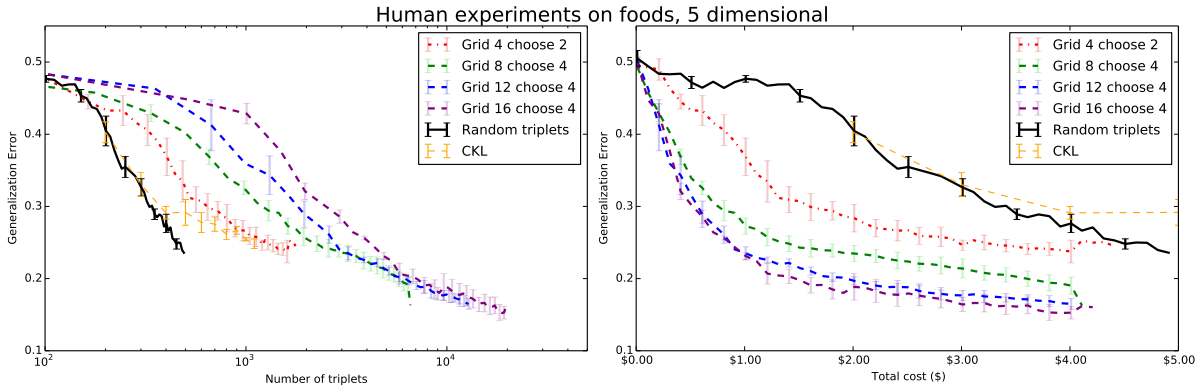


Figure 3.6. Results of the human experiments on the food dataset. Left graph: Triplet generalization error when viewed with respect to the total number of triplets. Right: The same metric when viewed with respect to the total cost (to the researcher) of constructing each embedding.

Baseline. Since the goal is to show that grid triplets produce better-quality embeddings at the same cost as random triplets, random (i, j, k) constraints are collected from crowd workers for comparison. Unfortunately, collecting all comparisons one at a time is infeasible (see the "Cost" results below), so instead, a groundtruth embedding from all grid triplets is construct and random constraints are uniformly sampled from the embedding. This is unlikely to lead to much bias because 39% of the possible unique triplets were collected, meaning that t-STE only has to generalize to constraints that are likely to be redundant. All evaluations are performed relative to this reference embedding.

3.3.3 Results

Two example embeddings are shown in Fig. 3.7.

Cost. Across all experiments, 14,088 grids are collected, yielding 189,519 unique triplets. Collecting this data cost $158.30, but sampling this many random triplets one at a time would have cost $2,627.63, which was far outside this project's budget[2]. If the 16-choose-4 grid strategy is used (which yields 48 triplets per grid), then all unique triplets would be able to be sampled

[2]There are $100 \cdot 99 \cdot 98/2 = 485,100$ possible unique triplets and each triplet answer would cost one cent. Additionally there is a 10% cut for Amazon and 20% of the tasks are devoted to catch trials.

for about $140. A feat that would cost $6737.50 by sampling one at a time.

Table 3.1. Results of the actual Mechanical Turk experiments. Workers are asked to choose the k most similar objects from a grid of n images. $1 worth of questions is invested, giving 100 grid selections. When n and k are large, each answer yields more triplets.

Grid n choose k		Error at $1	Time/screen (s)	Wages ($/hr)
n: 4,	k: 1	0.468	3.57	**$10.09**
	k: 2	0.369	3.45	**$10.45**
n: 8,	k: 1	0.400	3.04	**$11.85**
	k: 2	0.311	5.79	**$6.22**
	k: 4	0.273	7.65	$4.71
n: 12,	k: 1	0.406	4.17	**$8.64**
	k: 2	0.294	6.78	$5.31
	k: 4	0.235	8.67	$4.15
n: 16,	k: 1	0.413	6.72	$5.36
	k: 2	0.278	8.84	$4.07
	k: 4	0.231	9.59	$3.76
Random		0.477	–	–
CKL		0.403	–	–

Quality. In general, as more money is spent and more triplets are collected, t-STE does a better job generalizing to unseen redundant constraints. All embeddings converge to lower error when given more triplets, but this convergence is not monotonic because humans are fallible and there is randomness in the embedding construction. See Fig. 3.6 for a graphical comparison of grids with size 4,8,12, and 16. When viewed with respect to the number of triplets, random triplets again come out ahead. However, when viewed with respect to cost, the largest grid converges more quickly than others, and even the smallest grid handily outperforms random triplet sampling. The embedding generated using a 16-choose-4 grid costs $0.75, while an embedding with random triplets of similar quality costs $5.00.

This time, a large separation between the performance of various grid sizes is observed. Grid 16-choose-4, which yields $4 \cdot 12 = 48$ triplets per answer, uniformly outperforms the rest, with Grid 12-choose-4 (at $4 \cdot 8 = 32$ triplets per answer) close behind. Both of these outperform 8-choose-4 (16 triplets/answer) and 4-choose-2 (4 triplets/answer).

The performance grid is compared with the adaptive triplet sampling strategy described

in [TLB+11]. CKL picks triplets one-at-a-time but attempts to select the best triplet possible to ask by maximizing the information gain from each answer. In the experiments, it did not outperform random sampling. Further analysis will be future work.

Though catch trials comprised 20% of the collected grid answers, the results were generally of such high quality that no filtering or qualification was required.

Time. Fig. 3.5 shows how fast each human takes to answer one grid question. The smallest task was completed in 3.5 seconds, but even the largest grid (16 choose 4) can be completed in less than 10 seconds. Times varies widely between workers: the fastest worker answered 800 questions in an average of 2.1 seconds per grid task for 8-choose-1 grids.

Worker Satisfaction. At the standard 1¢-per-grid/$0.10-per-HIT rate, the workers are able to make more than a few dollars per hour, shown in Tab. 3.1. The smallest tasks net more than $10/hour by median, but even the largest task allows half of the workers to make $3.76 for every hour they spend. If the fastest, most skilled worker sustained their average pace in 8-choose-1 grids, they could earn over $17 per hour. Since there is a trade-off between grid size and worker income, it is important to consider just how far the workers can be pushed without stepping over the acceptable boundaries. Across all of the experiments, there were no complaints, and the tasks were featured on multiple HIT aggregators including Reddit's `HitsWorthTurkingFor` subreddit and the "TurkerNation" forums as examples of bountiful HITs.

According to the `HitsWorthTurkingFor` FAQ [3], "the general rule of thumb ...is a minimum of $6/hour." Though HITs below this amount may be completed, the best workers may pass for more lucrative HITs. Being featured in forums such as `HitsWorthTurkingFor` helped give the HITS used in this chapter more visibility to a very large audience of potential skilled turkers. Though high payouts mean higher cost, in this case, the benefit outweighed the drawback.

[3] http://reddit.com/r/HITsWorthTurkingFor/wiki/index

3.3.4 Guidelines and conclusion

The work in this section has shown that taking advantage of a simple batch User interface, researchers can save significant amounts of money when gathering crowdsourced perceptual similarity data. The recommendations can be summarized as follows:

- Rather than collecting comparisons one-at-a-time, researchers should use a grid to sample comparisons in batch, or should use some other UI paradigm appropriate to their task. However, researchers should not assume that such batch comparisons are of identical quality to uniformly random sampling. This is a trade-off that should be considered.

- If cost is an issue, researchers should quantify their results with respect to dollars spent. Using the simple UI paradigm can create embeddings of higher quality than those created using algorithms that pick the best triplet one-at-a-time.

- Researchers should continuously monitor the human effort of their tasks, so that they can calculate an appropriate target wage.

- When using grids to collect triplets, researchers should consider the trade-off between size and effort. Consider that an n-choose-k grid can yield

$$k(n-k) \tag{3.3}$$

  triplets per answer. Since this has a global maximum at $n = 2k$, one appropriate strategy is to select the largest n that yields a wage of \$6/hour and set k equal to $n/2$.

There are several opportunities for future work. First, the relationship between n, k, and task completion time should be better quantified to build a more accurate model of human performance. Second, triplet sampling algorithms such as "CKL" shoud be investigated more thoroughly as there may be opportunities to adaptively select grids to converge faster than random, giving advantages of both strategies.

3.4 "SNE-and-Crowd-Kernel" (SNaCK) embeddings

To create perceptual embeddings, the hybrid embedding algorithm, SNE-and-Crowd-Kernel (SNaCK), jointly optimizes the objective functions of two different low-dimensional embedding algorithms.[4] The first algorithm, t-SNE [VdMH08], uses a distance matrix to construct a low-dimensional embedding. Its goal is to ensure that objects which are close in the original high-dimensional space are also close in the low-dimensional output without constraining points that are far in the original space. The second method, t-STE [VdMW12b], allows labelers to supply triplet constraints that draw from their domain knowledge and task-specific hints. The work in this chapter will show that surprisingly simple joint optimization can capture the benefits of both objectives. See Fig. 3.9 for an overview.

3.4.1 Formulation

As mentioned before, consider N objects. The goal to produce a d-dimensional embedding $Y \in \mathscr{R}^{N\times d}$. Let $K \in \mathscr{R}^{N\times N}$ be a distance matrix, and let $T = \{t_1, \ldots, t_M\}$ be a set of triplet constraints. Each constraint $t_\ell = (i, j, k)$ implies that in the final embedding, object i should be closer to object j than it is to k, meaning $\|y_i - y_j\|^2 \leq \|y_i - y_k\|^2$. According to [VdMH08], the loss function for t-SNE can be interpreted as finding the low-dimensional distribution of points that maximizes the information gain from the original high-dimensional space.

$$C_{tSNE} = \sum_{i\neq j} p_{ij} \log \frac{p_{ij}}{q_{ij}}, \tag{3.4}$$

[4] Code is available on the companion website, http://vision.cornell.edu/se3/projects/concept-embeddings

where

$$p_{j|i} = \frac{\exp(-K_{ij}^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-K_{ik}^2/2\sigma_i^2)} \tag{3.5}$$

$$p_{ij} = \frac{1}{2N}(p_{j|i} + p_{i|j}) \tag{3.6}$$

$$q_{ij} = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k\neq l}(1+\|y_k - y_l\|^2)^{-1}} \tag{3.7}$$

and σ_i is chosen to satisfy certain perplexity constraints.

The loss function for t-STE, given in [VdMW12b], can be interpreted as the joint probability of independently satisfying all triplet constraints. It is defined as

$$C_{tSTE} = \sum_{(i,j,k)\in T} \log p_{(i,j,k)}^{tSTE}, \tag{3.8}$$

where

$$p_{(i,j,k)}^{tSTE} = \frac{\left(1+\frac{\|y_i-y_j\|^2}{\alpha}\right)^{-\frac{1+\alpha}{2}}}{\left(1+\frac{\|y_i-y_j\|^2}{\alpha}\right)^{-\frac{1+\alpha}{2}} + \left(1+\frac{\|y_i-y_k\|^2}{\alpha}\right)^{-\frac{1+\alpha}{2}}} \tag{3.9}$$

C_{tSTE} and C_{tSNE} is used in the cost function, defined as

$$C_{SNaCK} = \lambda \cdot C_{tSTE} \; + \; (1-\lambda)\cdot C_{tSNE} \tag{3.10}$$

To optimize this cost, gradient descent on $\frac{\partial C_{SNaCK}}{\partial Y}$ is used. The implementation derives from the t-SNE implementation in `scikit-learn`, so their optimization strategy is used. In particular, t-SNE is trained for 300 iterations and early exaggeration [VdMH08] heuristic is used for the first 100 iterations.

The λ parameter specifies the relative contribution of the machine-computed kernel and the human-provided triplet constraints on the final embedding. For each experiment, λ is chosen up front such that the norm of $\frac{\delta C_{tSTE}}{\delta Y}$ is approximately equal to $\frac{\delta C_{tSNE}}{\delta Y}$ in cross validation.

3.4.2 SNaCK example: MNIST

To briefly illustrate why this formulation is interesting, a toy example is shown in Fig. 3.10 using MNIST data. In this example, suppose the expert wishes to capture the concept of primality by partitioning the dataset into prime numbers $\{2,3,5,7\}$, composite numbers $\{4,6,8,9\}$, and other $\{0,1\}$. Also, for the purpose of this simple example, assume that rather than labeling the digits directly, the expert compares images based on concept similarity, i.e., primes are more similar to primes than to other images. By running t-SNE on flattened pixel intensities, Fig. 3.10 (A) illustrates that the embedding does a reasonable job of clustering numbers by their label but clearly cannot understand primality because this concept is not apparent from visual appearance. To compensate, triplet constraints of the form (i,j,k) where i and j share the same concept and k does not are sampled. However, only 1,000 constraints are sampled for these 2,000 images. t-STE (B) attempts to discover the differences between the numbers in a "blind" fashion, but since it cannot take advantage of any visual cues, the underconstrained points are effectively random. If given many more constraints, eventually t-STE can only collapse everything into three points for each of the three abstract concepts. The SNaCK embedding (C) displays the desired high-level concept grouping into primes/non-primes/others, and it can capture the structure of each class. Points with too few constraints are corrected by the t-SNE loss and the t-STE loss captures the appropriate structure.

3.5 Experiments

The MNIST example demonstrates SNaCK's utility in a domain where concepts can be derived from category labels and everything is known a priori. How does SNaCK perform on domains where a fixed taxonomy or fixed category labels are not necessarily known up front? To explore this question, a series of experiments were performed: first, SNaCK's ability was shown to help label a subset of CUB-200 in a semi-supervised fashion. In this setting, SNaCK learns concepts that are equivalent to category labels and outperforms other semi-supervised learning

algorithms. Second, experiments on a dataset of 10,000 unlabeled food images demonstrate SNaCK's ability to capture the concept of food taste using crowdsourcing. The embedding's generalization error was evaluated on a held-out set of crowdsourced triplet constraints. Finally, SNaCK's ability was shown to embed a set of pictographic characters, demonstrating how an expert can interactively explore and refine the structure of an embedding where no prior knowledge is available.

3.5.1 Incrementally labeling CUB-200-2011

This scenario shows how SNaCK embeddings can help experts label a new dataset. Suppose an expert has a large dataset with category annotations and an unlabeled smaller set containing new classes similar to those they already know. The expert wishes to use their extensive preexisting knowledge to quickly label the new set with a minimum amount of human effort. The goal is to show that SNaCK allows the expert to collect high-quality labels more quickly than other methods. Here, the learned "concepts" are equivalent to category labels. These experiments are inspired by [LG11a]. See Fig. 3.11.

Dataset. In this task the "Caltech-UCSD Birds 200-2011" (CUB-200) dataset [WBW+11] is used. It's assumed that the expert has access to all images and labels of 186 classes in the dataset (to train a machine kernel) and wishes to quickly label a testing set of 14 classes of woodpeckers and vireos. This subset contains 776 images and was defined in [FOZ+11]. Only profile-view bird images, where a single eye and the beak is visible, were used. Images are rotated, scaled, and possibly flipped so the eye is on the left side of the image and the beak is on the right side; part locations are collected using crowdsourcing. The image is then cropped to the head. This is the same normalization strategy as [BHBP14].

Automatic similarity kernel. To generate the distance matrix K, a CNN is fine-tuned for a classification task on all images in the 186 known classes. This allows the expert to leverage their extensive pre-existing dataset to speed up label collection for the novel classes. The network is a variation of the "Network-in-Network" model [LCY13], which takes cropped normalized

bird heads as input and outputs a 186-dimensional classification result. The pre-trained ImageNet model is from the Caffe model zoo [JSD+14] and the network is fine-tuned for 20,000 iterations on an Amazon EC2 GPU instance. To do this, the last layer is replaced with a 186-class output and the learning rate is reduced for the other layers to a tenth of the previous value.[5] Finally, $K^{CNN}_{i,j}$ is the Euclidean distance between features in the final layer before softmax. To evaluate the importance of specialized kernels, this K^{CNN} kernel is also compared to Euclidean distances between pre-trained GoogLeNet [SLJ+14] features, and Euclidean distance between HOG features.

Expert constraints. To generate triplet constraints in a semi-supervised fashion, the labels for n images of the dataset are revealed and all triplets are sampled between these images that satisfy same/different label constraints to generate $T_n = \{(i,j,k) \mid \ell_i = \ell_j \neq \ell_k, \max(i,j,k) \leq n\}$. With these triplets, the amount of expert effort required to label novel images can be varied. Note that in this test, the concepts to learn are equivalent to class labels, so all of the sampled constraints are derived from ground truth. The food experiments, described in the next section, will demonstrate SNaCK's ability to learn more abstract concepts captured from subjective human judgments.

Comparisons and metrics. To perform labeling with SNaCK, an embedding of all 776 images is generated and KMeans is used to find clusters. To evaluate, all points within each discovered cluster is assigned to their most common ground truth label and the accuracy of this assignment is calculated. See Fig. 3.13 for example embeddings varying the number of expert label annotations. These results are compared against other semi-supervised learning and constrained clustering systems: Label Propagation [BDR06], the multiclass version of the Constrained Spectral Clustering KMeans (CSPKmeans) method described in [WQD12], and Metric Pairwise-Constrained KMeans (MPCKmeans) [BBM04]. Label propagation uses K^{CNN} and the n revealed labels. The constrained clustering systems use K^{CNN} and pairwise "Must-

[5] When trained using the standard training/testing protocol on all of CUB-200, this kind of model achieves 74.91% classification accuracy, which is comparable to the state-of-the-art [BHBP14].

Link" and "Cannot-Link" constraints as input, thus n image labels are revealed and all possible pairwise constraints are sampled between them. As baselines, CNN features are calculated and are clustered with KMeans and spectral clustering, which do not benefit from extra human effort. Finally, the results are compared against the cluster results of using K-Means on a t-STE embedding from the same triplet constraints used by SNaCK.

Results are shown in Fig. 3.12. SNaCK outperforms all other algorithms, but label propagation and MPCKMeans also perform well. CSPKmeans is eventually outpaced by naively asking the expert for image labels, perhaps because it was designed for the two-class setting rather than the 14-class case. These experiments show that t-STE benefits from an automatic machine kernel (compare SNaCK to t-STE), but the machine kernel can be improved with a small number of expert annotations (compare KMeans or Spectral Clustering to SNaCK).

Using a kernel that captures bird similarity well is particularly important for this task. All of the algorithms which use K^{CNN} generally outperform SNaCK when using a pre-trained GoogLeNet kernel. HOG features, which use no learning, are only slightly better than naive labeling. Finally, t-STE cannot use any visual kernel, so it can only consider the images the expert already revealed.

Sometimes the machine kernel disagrees with the expert hints. This may happen for interesting reasons, such as mistakes in the training data. For example, Figure 3.8 shows an instance of a Red-headed Woodpecker that was moved into a cluster containing many Pileated Woodpeckers. Even though the human constraints encourage this sample to lie near similarly labeled examples, this individual looks overwhelmingly similar to a Pileated Woodpecker, so the t-SNE loss overpowered the t-STE constraints. If the embedding is colored with ground truth labels, this mistake shows up as a single differently-colored point in the expected cluster, which is immediately apparent to an expert.

Discovering labels for semi-supervised classifiers

Does better incremental labeling translate into increased classification performance? This scenario extends the previous experiment: SNaCK is used to discover labels for a training set and measure the accuracy of a simple SVM classifier on a testing set. The goal is to decide whether just letting an expert reveal n labels and training on this smaller set is better than revealing n labels and using SNaCK to discover the rest. Will a classifier trained on many noisy, discovered labels perform differently than a classifier trained on a smaller, perfect training set?

Dataset. This task uses the same set of 14 woodpeckers and vireos from CUB-200 as before, but the procedure is different. The dataset is split into 396 training and 380 testing images using the same train/test split as CUB-200. Then labels are discovered on the training images using varying numbers of expert annotations and a linear SVM classifier is trained on all CNN features using the discovered labels. Finally, accuracy is reported on the 380 testing images. The idea is that the quality of the discovered labels influences the accuracy of the classifier: a poor labeling method will cause the classifier to be trained on incorrect labels. Because all methods use the same type of classifier, the quality of discovered labels is evaluated, not the classifier itself.

Comparisons. As a baseline, the SVM classifiers trained on SNaCK-discovered labels are compared to an SVM classifier trained on a smaller, better set of n correct labels provided by expert ground truth. This corresponds to the "Naive Human Sampling" method in Fig. 3.12. Baselines where the SVM training set labels are discovered using KMeans, spectral clustering, and label propagation, are also compared to.

Results are shown in Fig. 3.14. Classifiers trained on noisy labels discovered from SNaCK embeddings significantly outperform classifiers that are trained on smaller training sets, even though many of SNaCK's labels are incorrect. This is particularly true for fewer than 50 annotations. Accuracy of SNaCK, Label Propagation, and naive label sampling saturates at about 85%, which is likely due to the linear SVM's limited generalization ability.

Interestingly, classification accuracy of labels discovered with MPCKMeans does not monotonically improve with more expert annotations. This surprises us, but Fig. 3.12 does show that MPCKmeans saturates to a smaller value in the semi-supervised labeling experiments, indicating that it cannot perfectly satisfy (and thus does not benefit from) additional constraints.

Using SNaCK, an expert can build a classifier that achieves 78.8% classification accuracy by labeling 50 images (12.6% of the dataset). A standard SVM that achieves this level of accuracy requires a training set of 95 perfectly labeled images, showing that SNaCK can cut down the expert's work load to build training sets for classifiers.

3.5.2 Experiments on Yummly-10k

In this scenario, SNaCK is used to generate embeddings of food dishes. The goal is to create a concept embedding that captures the concept of taste. Two foods should be close in this embedding if they taste similar, according to subjective human judgments. This is different from the earlier bird experiments because there are no longer labels or taxonomies to help refine the embedding; all expert hints must come directly from unquantified human perception annotations. See Fig. 3.16.

Dataset. This experiment uses 10,000 food images from the Yummly recipe web site, dubbed Yummly-10k. This data contains a variety of meals, appetizers, and snacks from different cultures and styles. The images are filtered by removing all images shorter or thinner than 300 pixels and removing all drinks and non-edibles. As metadata, Yummly includes weak ingredients lists and the title of the dish, but it does not include food labels.

Automatic similarity kernels. SNaCK is not specific to any specific kernel representation, so two kinds of similarity measures are compared. Food Kernel 1 is a semantic similarity measure between two foods' ingredient lists, and Food Kernel 2 is a visual similarity measure based on a convolutional neural network. To create $K_{i,j}^{\text{word2vec}}$ (Food Kernel 1), let I_i and I_j be food i and j's ingredients lists from Yummly. Let $w(\cdot)$ be an ingredient's word2vec[MCCD13] representation, scaled to unit norm, and let cost matrix $C(a,b) = w(a) \cdot w(b)$ for $a \in I_i, b \in I_j$.

Finally, let $f : I_i \rightarrow I_j$ be the maximum-weight assignment between the two ingredient lists. Then, $K_{i,j}^{\text{word2vec}} = -\sum_{a \in I_i} C(a, f(a))$. This way, Food Kernel 1 determines foods that share many common ingredients are more similar than foods that have many dissimilar ingredients.

To build Food Kernel 2, a CNN is fine-tuned to predict a food label. Because Yummly-10k does not have any labels, the network is trained on the Food-101 dataset from [BGVG14]. Similarly to the earlier bird experiments, the network is a variation of the "Network-in-Network" model trained to classify 101 different foods. It was trained for 20,000 iterations on an Amazon EC2 GPU instance by replacing the last layer and reducing the learning rate. The final kernel is defined as the Euclidean distance between these CNN features. The CNN model provides an excellent kernel to start from. When trained via the standard Food-101 protocol, this model achieves rank 1 classification accuracy of 73.5%. The previous best accuracy on this dataset is 56.40% from [BGVG14].The best non-CNN is 50.76%. Of course, building a good classification model is not the focus, but this accuracy is reported to show that the automatic kernel is effective at distinguishing different foods.

Expert annotation. Because the goal of the embedding is to properly capture the concept of food taste, expert annotations are collected directly from humans on Amazon Mechanical Turk using the crowdsourcing interface described in 3.3. For each screen, a reference food image i and a grid of 12 food images is shown. The human is asked to "Please select 4 food images that taste similar to the reference food i." Then all possible triplet constraints $\{(i, j, k), j \in S, k \notin S\}$ are generated, where S is the user's selection. Each HIT has 10 screens and yields 320 triplet constraints. In total, 958,400 triplet constraints are collected.[6]

Experiment design. There are no labels associated with taste in the Yummly data, so the quality of the perceptual embeddings must be evaluated with other metrics. To do this, the "Triplet Generalization Error" metric common to previous work [HBSH15, ZMT15, VdMW12b, WKB14b] is used. All triplet constraints are split into training and testing sets and embeddings are generated with varying numbers of training triplet constraints. Triplet generalization error is

[6]Triplets are available from the companion website, http://vision.cornell.edu/se3/projects/concept-embeddings

defined as the fraction of violated testing triplet constraints, which measures the embedding's ability to generalize to constraints the expert did not specify. The two SNaCK kernels are compared to t-STE.

Results are shown in Fig. 3.17 and an example embedding is shown in Fig. 3.15. As more triplet constraint annotations become available, all methods produce embeddings of higher quality. SNaCK with Kernel 2 eventually converges to 28% error while t-STE reaches 33% error. Note that t-STE starts from random chance (50%) because it starts with no information, while SNaCK-based methods initially start with lower error because the Stochastic Neighbor loss on the automatic kernel encourages an initial embedding that contains some fine-grained information. Kernel 2 consistently outperforms Kernel 1, indicating that in this experiment, deep-learned visual features may be a better indication of food taste than the similarity of food ingredient lists. However, even the "weaker" semantic ingredient information provides a much better initial kernel than nothing at all.

3.5.3 Interactively discovering the structure of pictographic character symbols

In this section possible tools for exploring unlabeled data are described. The work in this section analyzes a set of 887 pictographic characters, colloquially known as Emoji. Using CNN features pre-trained on ImageNet, an embedding that does a good job of grouping visually similar Emoji together is created. However, if the goal is to capture the concept of emotion within the set of Emoji, then similarity of visual features alone may be inadequate. For example, in Fig. 3.18.A, a group of yellow faces are clustered at the upper right, but this group contains different emotions and does not contain similar images in other artistic styles.

To interactively refine the embedding, the expert selects a reference Emoji and drags a box around several images. The expert then indicates which of these images share the same emotion as the reference. In the example in Fig. 3.18, a smiling Emoji was selected and compared to all the Emoji in the green box (Fig. 3.18.B). After two bounding box selections and a few

minutes of work, we are able to collect 20,000 triplets and separate many of the smiling Emoji from the rest of the embedding. From here, we could further inspect these Emoji and separate the emotion of laughing from smiling.

As mentioned in the MNIST experiments, the SNaCK embeddings are capable of taking advantage of visual cues when triplet information is not available. An example of this can be seen in Fig. 3.18.D. A fearful face with glasses is moved from the left side of embedding to be near other faces with similar expressions. SNaCK was able to do this without requiring triplets to be collected between these faces. These examples give a brief illustration of how SNaCK can be useful for examining unlabeled data.

3.6 Conclusion

The SNaCK algorithm can learn concept embeddings by combining human expertise with machine similarity. We showed that SNaCK can help experts quickly label new sets of woodpeckers and vireos, build training sets for classifiers in a semi-supervised fashion, and capture the perceptual structure of food taste. We also presented a snapshot of a tool that can help experts interactively explore and refine a set of pictographic characters. In the future, the goal will be to pursue intelligent sampling for active learning of embeddings, and will extend the system to explore large video datasets.

3.7 Acknowledgments

Chapter 3 is based on the following papers: “Cost-effective hits for relative similarity comparisons.” M. J. Wilber, I. S. Kwak, and Serge J. Belongie, Second AAAI conference on human computation and crowdsourcing (HCOMP) 2014 [WKB14a]. and “Learning concept embeddings with combined human-machine expertise,” M. J. Wilber, I. S. Kwak, D. Kriegman, and S. Belongie, International Conference on Computer Vision (ICCV) 2015 [WKKB15]. The dissertation author was one of two contributing authors of this paper in both algorithm and

manuscript development.

Figure 3.7. **Top:** An example cuisine embedding, collected with the 16-choose-4 grid UI strategy. This embedding cost $5.10 to collect and used 408 screens, but yielded 19,199 triplets. It shows good clustering behavior with desserts gathered into the top left. The meats are close to each other, as are the salads. **Bottom:** An embedding with 408 random triplets. This embedding also cost $5.10 to collect, but the result is much dirtier, with worse separation and less structure. Salads are strewn about the right half of the embedding and a steak lies within the dessert area. From the experiments, an embedding of such low quality would have cost less than $0.10 to collect using grid strategy.

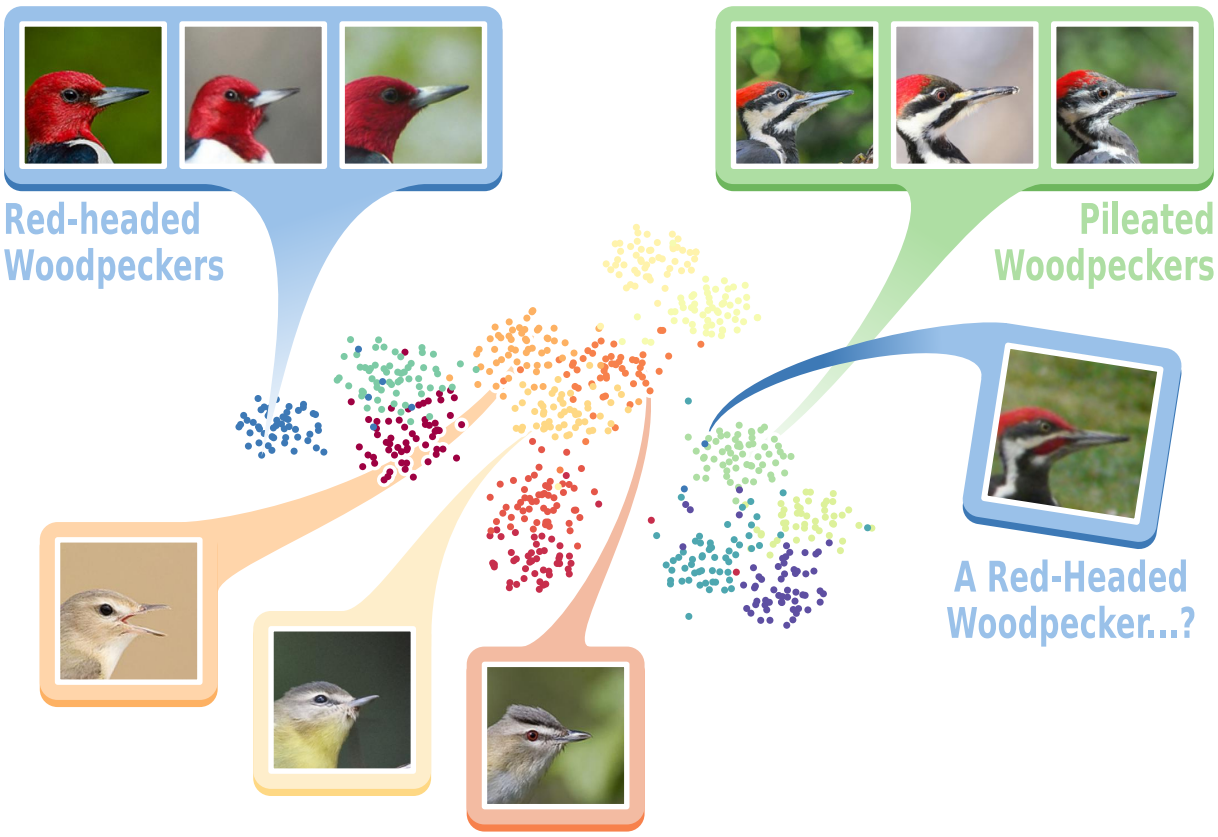


Figure 3.8. The SNaCK embeddings capture human expertise with the help of machine similarity kernels. For example, an expert can use this concept embedding of a subset of CUB-200 to quickly find labeling mistakes. Red-headed Woodpeckers are visually dissimilar to Pileated Woodpeckers, but SNaCK moved a Red-headed Woodpecker into the Pileated Woodpecker cluster because of its appearance. **This is probably a labeling mistake in CUB-200, and this SNaCK embedding helped us discover it.** The cluster of three visually similar vireo species in the embedding center may be another good place to look for label problems.

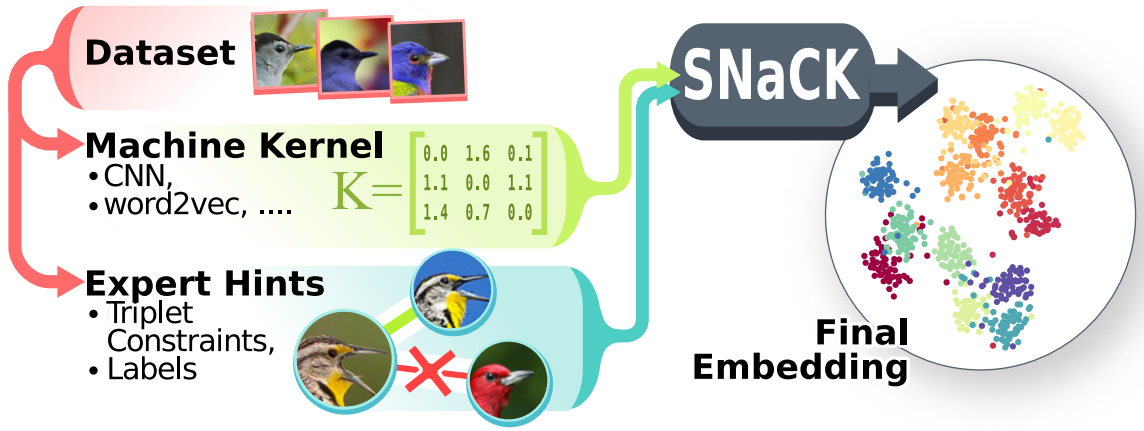


Figure 3.9. Overview of the SNE-and-Crowd-Kernel ("SNaCK") embedding method. As input, SNaCK accepts a dataset of objects, a similarity kernel K, and a set of expert constraints in the form of "Object i is closer to j than k", which may be inferred from crowdsourcing or label information. The output is a low-dimensional concept embedding that satisfies the expert hints while preserving the structure of K.

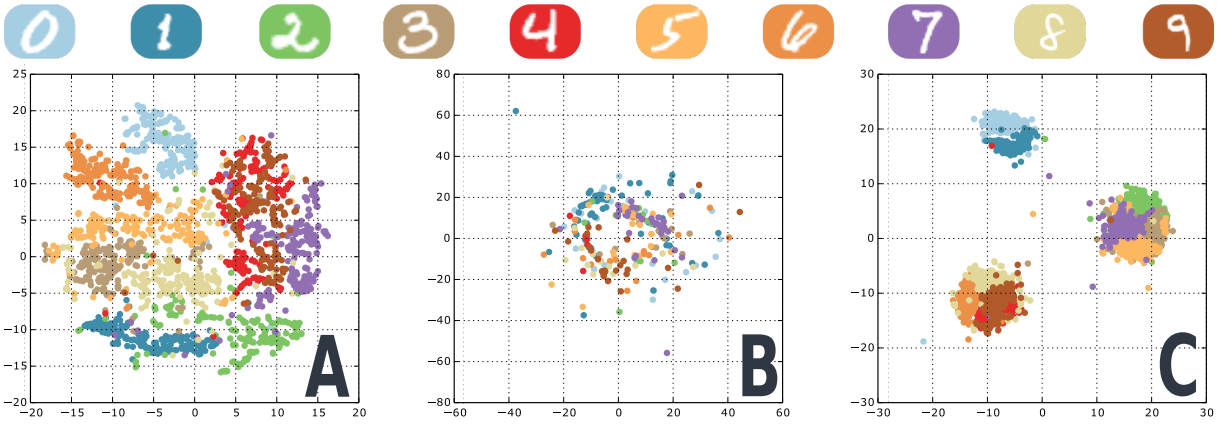


Figure 3.10. A simple MNIST example to illustrate the advantages of SNaCK's formulation. Suppose an expert wishes to group MNIST by some property that is not visually apparent, in this case: prime, composite or $\{0,1\}$. (A) shows t-SNE on 2,000 MNIST digits using flattened pixel intensities. (B) shows t-STE on 1,000 triplets of the form (i,j,k), where i and j share the same concept but k does not. (C) shows a SNaCK embedding using the same flattened pixel intensities and the same triplet constraints. The SNaCK embedding is the only one that captures the intra-class structure from (A) and the desired abstract grouping of (B). See 3.4.2 for details.

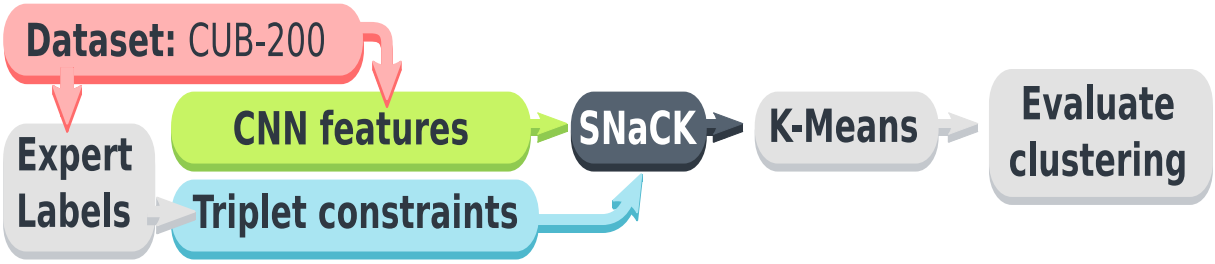


Figure 3.11. Experiment overview on CUB-200. See text for details.

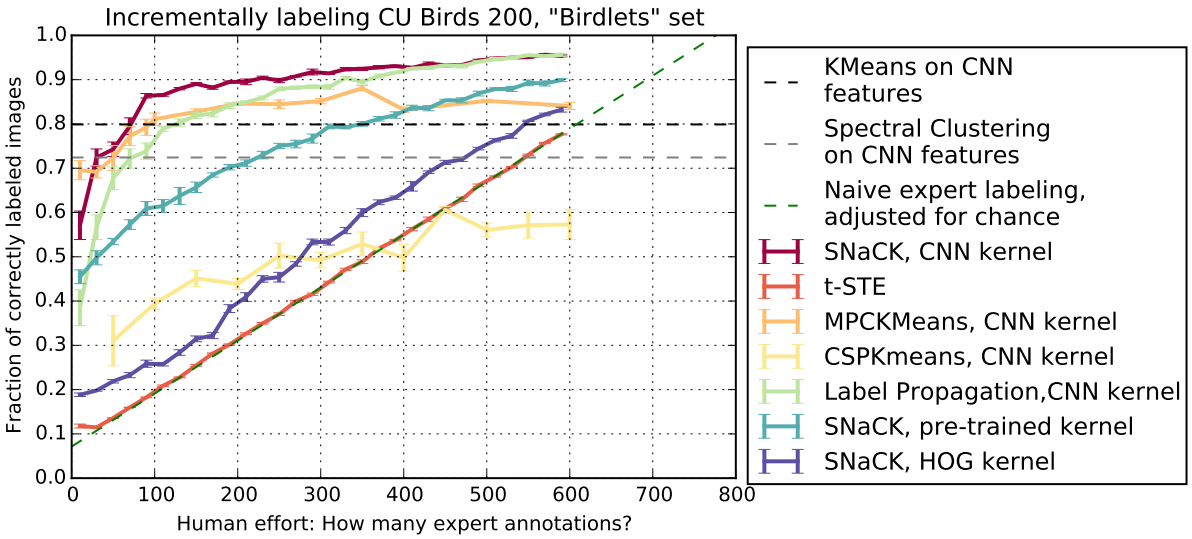


Figure 3.12. Incremental labeling accuracy of several semi-supervised methods. X axis: how many labels are revealed to each algorithm. Y axis: Dataset labeling accuracy. Error bars show standard error of the mean ($\sigma/\sqrt{n}$) across five runs. With 14 clusters, chance is ≈ 0.071. See Sec. 3.5.1 for details.

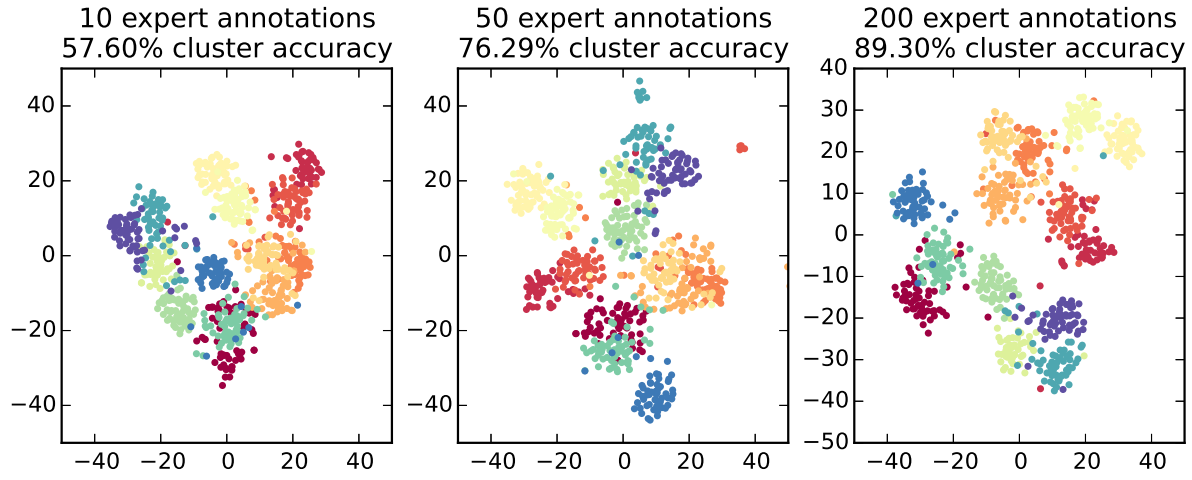


Figure 3.13. Embedding examples on CUB-200 Woodpeckers and Vireos, showing the "SNaCK" method with (left-to-right) 10, 50, and 200 expert label annotations. Colors indicate ground truth labels. As the number of expert annotations increases, clusters within the SNaCK embedding become more consistent.

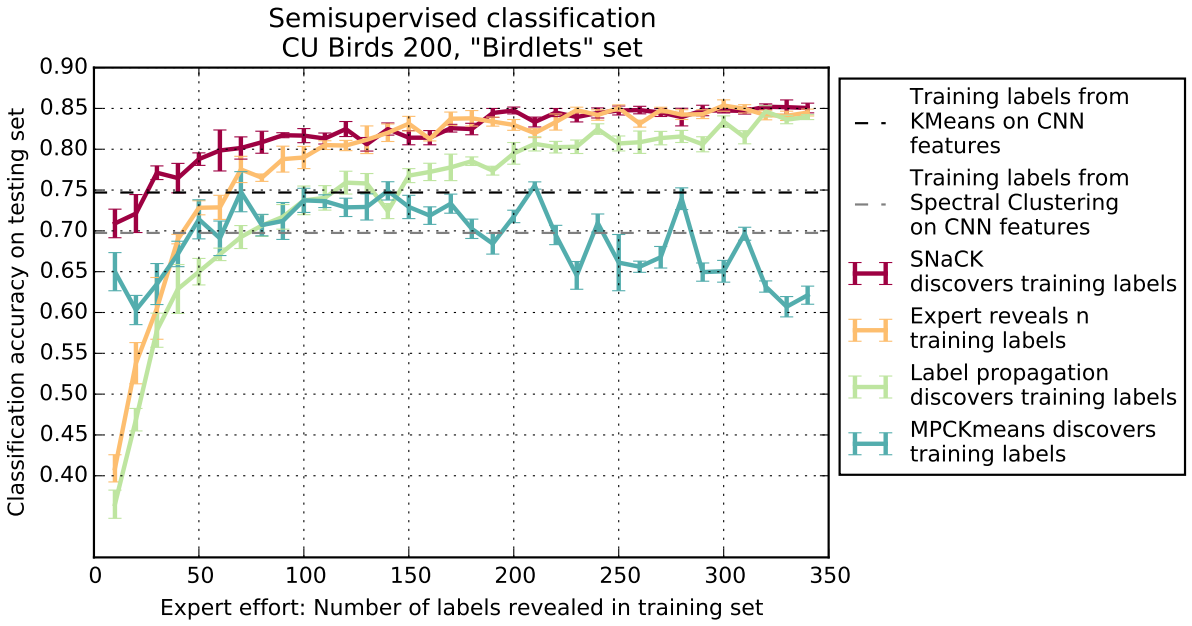


Figure 3.14. Classification accuracy of a linear SVM classifier trained on labels discovered by different methods. X axis: how many training labels are revealed to each algorithm. Y axis: Accuracy of classifier trained with these labels on the test set. Error bars show standard error of the mean ($\sigma/\sqrt{n}$) across five runs. See Sec. 3.5.1 for details.

Figure 3.15. Left: Example SNaCK embedding on Yummly-10k, combining expertise from Kernel 2 (CNN features) and 950,000 crowdsourced triplet constraints. Middle/right: Close-ups of the embedding. On a large scale, SNaCK groups major food kinds together, such as desserts, salad, and main courses. On a small scale, each food closely resembles the taste of its neighbors. See the supplementary material version for larger versions of this figure.

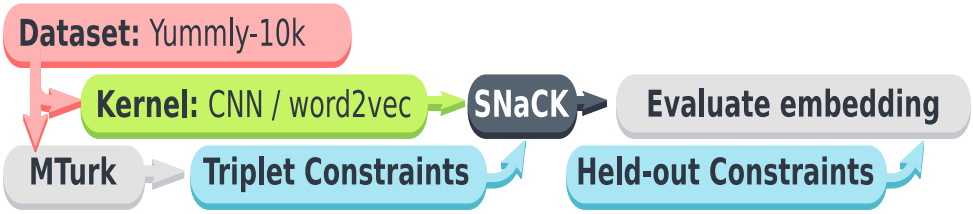


Figure 3.16. Experiment Overview for Yummly-10k. See text for details.

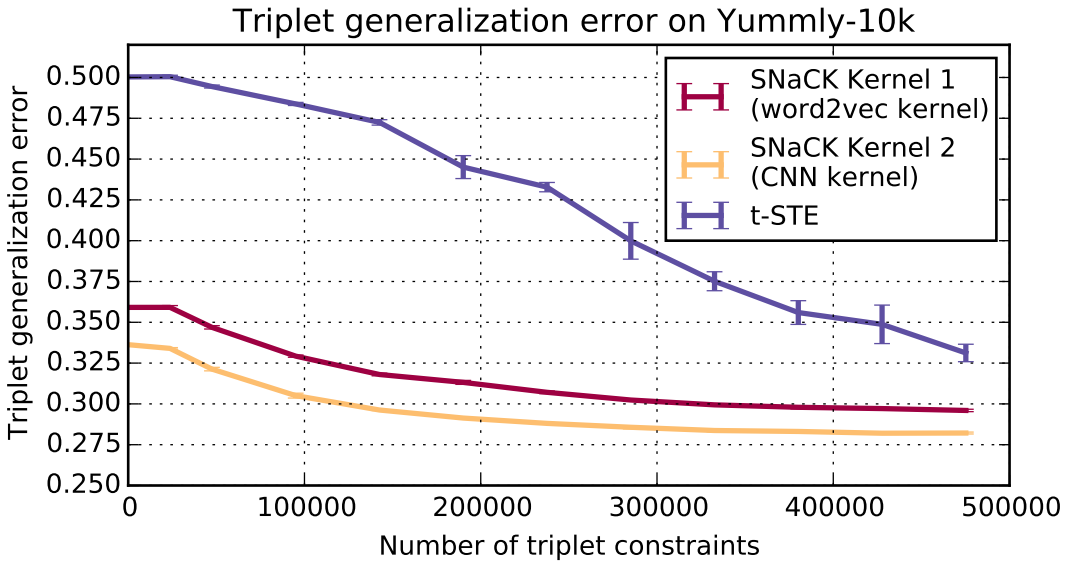


Figure 3.17. Increasing the number of crowdsourced triplet constraints allows all methods to improve the embedding quality, measured as the fraction of unsatisfied held-out triplet constraints ("triplet generalization error"). However, SNaCK-based methods converge much more quickly than t-STE and require less expert annotation to get a better result.

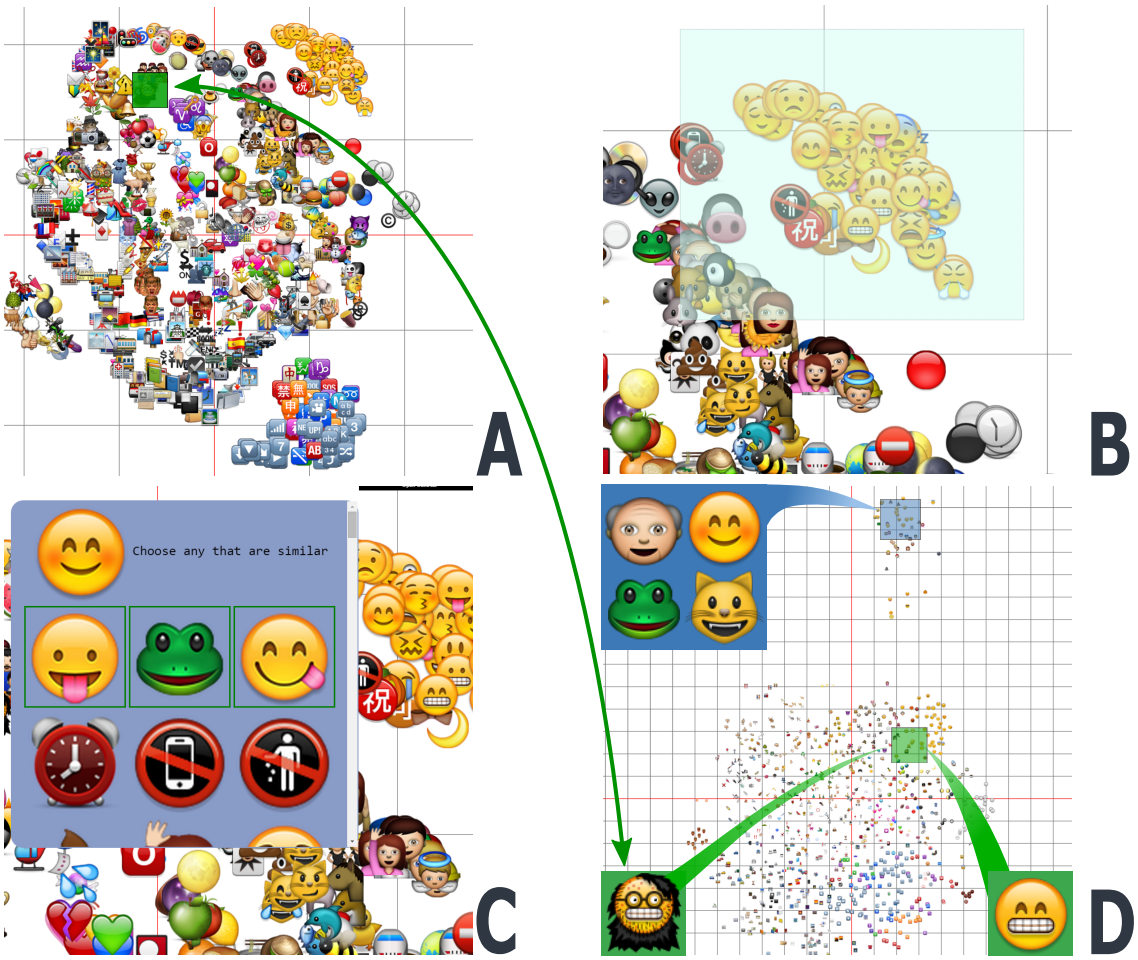


Figure 3.18. An example GUI used to interactively explore and refine concept embeddings. (A) shows a t-SNE embedding of Emoji using pre-trained ImageNet features. The user selects a set of images (B) and indicates which ones share the same emotion (C). For example, the user selected the smiling frog because it has a similar emotion to the top left image. The updated SNaCK embedding (D) moves smiling emoji away from unrelated images, regardless of the artistic style of the faces. Additionally one of the highlighted fearful faces, separate from the main cluster of faces in (A) has moved to be near faces with a similar expression without collecting triplets between them.

Chapter 4

Action Start Detection

4.1 Introduction

As technology improves, tools for automatic video analysis become more important. There is currently a large quantity of video data for a variety of applications, such as entertainment, surveillance, and robotics. In each case it becomes increasingly more difficult for humans to analyze and inspect video data manually. This is either due to the sheer quantity of data available or the need for rapid response. For example, video-based self-driving car technologies require fast reactions to changes in the environment.

Video content analysis is generally framed in one of two ways; action classification and action detection. In action classification [KTS+14, SZS12, IZJ+17], the goal is to assign a single category to a trimmed video. These videos are on average a couple seconds long, and contain a single behavior to classify. A variety of deep learning algorithms have been used to learn spatio-temporal models, such as convolutional [TBF+15] and recurrent networks [KTS+14]. These algorithms can be useful for content based video retrieval, and the learned representation can be used as input for action detection [WWN+18, EHNG16, GYC+17].

In fine-grained action detection or segmentation [IZJ+17, CHEGCN15], the goal is to determine time intervals (start and end frames) of each action category. Videos in this type of task can be much longer than action classification videos, and there may be multiple behaviors to temporally localize per video. As with action recognition, both convolutional and recurrent net-

works [SCZ+17, XDS17, SMJ+16, WWN+18] have been applied to the task. Action detection algorithms potentially have a wide range of applications, such as filtering background frames from surveillance footage, extracting highlight footage, or generating descriptions.

This chapter will focus on a recent research direction for video analysis, automatically detecting the start of actions. In contrast to action detection, where the duration of the behavior is classified, this chapter focuses on classifying only a single frame, representing the start of the behavior. [SPC+18] has proposed detecting the start of actions in an online setting, but this work will focus on detecting action starts in an offline setting.

This is an important problem for many computer-vision applications, including in neuroscience research. A fundamental goal in neuroscience is to understand the neural activity patterns that produce behavior. To do this, researchers localize the *start* of bouts of actions in videos, and then examine neural activity just prior to this [SGZ+18] (Figure 4.1a-b). This work will focus on detecting action starts, although the proposed methods could be applied to any identified key frame in an action bout.

There are three classes of errors that can be made in detecting action starts. One can miss an action start (false negative), have an extra action start (false positive), or be detected too many frames from the true start. Using an unstructured, per-frame error between the true and predicted action starts would incorrectly penalize being offset by a small number of frames more than having a false positive or negative (Figure 4.1c). Thus, this chapter proposes a structured loss that involves finding the best match between action start predictions and labels, and this allows a proper weighting of each of these three types of errors. A recurrent neural network (RNN) is used to minimize this structured loss using gradient descent. As this loss is not differentiable, an alternative differentiable proxy of this based on the Earth Mover's Distance (EMD) is proposed.

To accurately evaluate action start detection, a new video data set, The Mouse Reach Dataset, is introduced. This dataset has been annotated with the starting frames of a set of behaviors. The data set consists of videos of mice performing a task that starts with reaching for a food pellet and ends with chewing that pellet; when successful, the task consists of a sequence

of six actions (Figure 4.1a). The sequence has strong temporal structure that can be exploited by an RNN, but can also vary substantially. Actions may be repeated, such as when a mouse fails to grab the food pellet on the first attempt and then tries again. Experimental results show that an RNN trained to minimize either of the structured losses outperforms an RNN trained to minimize the per-frame loss. Furthermore, reaching tasks are often used in rodents and primates to study motor control, learning, and adaptation, and tools for automatic quantification of reach behavior would have immediate impact on neuroscience research [WP90, GGG+15, SGZ+18, KHX+19].

In summary this chapter a) introduces a novel structured loss function, b) a network architecture designed to minimize the loss, and c) contributes a new, real-world dataset for fine-grained action start detection, that has been annotated in the course of neuroscience research. The algorithm is described in Section 4.3, dataset in Section 4.5, and experimental results in Section 4.6.

Chapter 4 is based on "Detecting the Starting Frame of Actions in Video," I. S. Kwak, D. Kriegman, K. Branson and is currently being prepared for submission for publication of the material. The dissertation author was the primary investigator and author of this paper.

4.2 Related Work

Although this work focuses on detecting the start of an action, there are many similarities to fine-grained action detection, in which the goal is to categorize the action at each frame. To incorporate the temporal context, 3D convolutional networks and recurrent networks have been used to detect actions [SCZ+17, XDS17, SMJ+16, WWN+18]. Following the success of object proposals [UVDSGS13, RHGS15] for object detection, algorithms for proposing temporal segments for action classification have been developed [GYC+17, EHNG16].

Fine-grained action detection algorithms have leveraged feature representations first developed for action recognition [WWN+18, EHNG16, GYC+17], in which the goal is to categorize the entire trimmed video. Large-scale action recognition datasets [CZ17, KTS+14]

have helped produce strong representations of short video snippets, which then can be used by detection algorithms. 3D convolutional networks leverage lessons learned from successful image recognition networks [TBF+15] and simultaneously learn appearance and motion information. Recurrent models, such as LSTMs, have been used to model long range temporal relationships [KTS+14]. More recently, two stream networks [SZ14, WXW+16, CZ17] have been successful at action recognition. The work presented in this chapter uses two-stream feature representations as inputs to the detection model.

Online detection of action start (ODAS) [SPC+18] is the most similar past work to ours. In ODAS, the goal is to accurately detect the start of an action for use in real-time systems. In contrast, this chapter's work focuses on offline detection of action starts to understand the causes of behaviors, for example to understand the neural activity that produced a behavior. Both offline and online start detection have similar difficulties in label sparsity. A dataset for which the accuracy of the action start labels was the main focus in dataset creation is provided. This dataset will be useful for both online and offline action start detection research.

4.3 Problem Formulation

Let $\mathbf{X} = [x^1, x^2, ..., x^T]$ be a sequence of T video frames, where $x^t \in \mathbf{R}^d$ is the feature representation of each frame. The goal is to predict, for each frame t and behavior b, whether the frame corresponds to the start of a behavior bout ($y_b^t = 1$) or not ($y_b^t = 0$). Let $\mathbf{Y} = [y^1, y^2, ..., y^T]$ be the sequence of ground truth labels for X, where $y^t \in \{0,1\}^B$ and B is the number of behaviors.

Let $\hat{Y} = [\hat{y}^1, \hat{y}^2, ..., \hat{y}^T]$ be a predicted sequence of labels. Behavior starts, $y_b^i = 1$, are matched with predictions, $\hat{y}_b^j = 1$. Each label can be matched with at most one prediction within $|i-j| < \tau$ frames. Labeled starts without a matched predicted start are false negatives (FN) and get a fixed penalty of C_{fn}. Predicted starts without a matched true start are false positives (FP) and get a fixed penalty C_{fp}.

More formally, let $M \in \mathbf{Z}^{T \times B}$ be a matching from true to predicted starts, where $m_b^i > 0$

and $y_b^i = 1$ means that the true start of behavior b at frame i is matched to a predicted start of behavior b at frame m_b^i, and $m_b^i = 0$ indicates that the true start at frame i is not matched. Similarly, let $\bar{M}$ denote an inverse matching from predictions to labels consistent with M. Then, the error criterion can be written as a minimum over matchings M:

$$\begin{aligned} &Err(Y,\hat{Y}) = \\ &\min_{(M,\bar{M})} \sum_{bt} [\underbrace{I(y_b^t = 1)(I(m_b^t = 0)\tau}_{\text{FN}} + \underbrace{I(m_b^t > 0)|t - m_b^t|)}_{\text{TP}} + \underbrace{I(\hat{y}_b^t = 1)I(\bar{m}_b^t = 0)\tau]}_{\text{FP}} \end{aligned} \tag{4.1}$$

Note that $I(y_b^t = 1)I(m_b^t = 0) = 1$ for false negatives and $I(\hat{y}_b^t = 1)I(\bar{m}_b^t = 0) = 1$ for false positives.

The Hungarian algorithm [Kuh55] can be used to efficiently compute the optimal matching $(M,\bar{M})$ in this criterion. For each behavior, the bipartite graph consists of two sets of $N+M$ nodes, where N and M are the number of true and predicted action starts. In the first set, the first N nodes correspond to true starts and the last M nodes correspond to false positives. In the second set, the first M nodes correspond to predicted starts and the last N to false negatives. The distance matrix is then

$$D^{nm} = \begin{cases} |s^n - \hat{s}^m| & n \leq N, m \leq M \\ \tau & n > N, m \leq M \text{ (FP)} \\ \tau & n \leq N, m > M \text{ (FN)} \\ \tau & n > N, m > M \end{cases},$$

where s^n is the nth true action start and $\hat{s}^m$ is the mth predicted action start.

4.3.1 Matching Loss

A structured loss based on this error criterion is proposed, which will be referred as the Matching Loss. The classifier outputs are continuous values $\hat{y}_b^t \in [0,1]$. To compute this loss,

the classifier outputs are binarized by thresholding and non-maximal suppression, resulting in a sequence of predicted action starts $\hat{S}$. $\hat{S}$ is used to select an optimal matching $\hat{M}$ using the Hungarian algorithm, as described above. Then, the following loss is minimized, which is a differentiable function of the continuous classifier outputs:

$$\begin{aligned}\mathscr{L}_H(Y,\hat{Y},\hat{M}) = \quad & \textstyle\sum_{tb} C_b[\underbrace{I(y_b^t=1)(I(\hat{m}_b^t=0)C_{fn}}_{\text{FN}} - \\ & \underbrace{I(\hat{m}_b^t>0)(\tau-|t-m_b^t|)\hat{y}_b^t)}_{\text{TP}} + \\ & \underbrace{\hat{y}_b^t I(\bar{m}_b^t=0)C_{fp}}_{\text{FP}}]\end{aligned} \tag{4.2}$$

where C_b is a weight for behavior b (usually set to one over the number of true starts of that behavior). C_{tp}, C_{fn}, and C_{fp} are parameters for weighing the importance of true positives, false negatives, and false positives respectively.

Note that the loss can be applied to any matching, but in this case the optimal matching is chosen. With this loss, the importance of predicting action starts near the true behavior start frame while avoiding spurious predictions can be directly enforced. A correct prediction is penalized by the distance to the true behavior start frame and the confidence of the network output. Any prediction that is not matched, will be penalized by the network's output score.

Given the matching $\hat{M}$, this loss is differentiable and thus can be minimized using gradient descent. However, selecting the optimal matching $\hat{M}$ is not differentiable. Following [SAN16, WWN+18], the training will iteratively fix the network and select the the optimal matching, and then fix the matching and apply gradient descent to optimize the network. Experimentation found that, using this training procedure, the networks were able to learn to localize behavior start locations. One concern for this loss function is that it is not fully differentiable. For example, suppose there is a predicted start matched to a true start. While the total loss would decrease if the prediction were closer to the true start frame, gradient descent on the Matching loss with fixed matching will not do this.

4.3.2 Wasserstein/EMD Loss

The Matching Loss relies on an assignment of predictions to ground truth labels. The assignment problem can be approximated with the Wasserstein Distance [PC+19]. Similar to [WWN+18], the squared EMD loss, a variant of the 1-Wasserstein Distance, is used as an alternative structured cost for the sequence. Unlike [WWN+18], no matching is applied before computing the loss. Additionally, the work in this chapter applies the Wassertein loss to all predictions for a behavior simultaneously, rather than to each prediction separately. The EMD loss is completely differentiable. The predicted label sequence and the ground truth label sequence are first normalized:

$$\begin{aligned} y'^{i}_{b} &= \frac{1}{\sum_{i=0}^{T}(y^{i}_{b}+\varepsilon)}(y^{i}_{b}+\varepsilon) \\ \hat{y}'^{i}_{b} &= \frac{1}{\sum_{i=0}^{T}(\hat{y}^{i}_{b}+\varepsilon)}\hat{y}^{i}_{b} \end{aligned} \quad . \tag{4.3}$$

ε is added in case there are no labels or predictions in a given behavior class for a video. A Gaussian blur with standard deviation, σ_b, and window size, w_b is applied the groundtruth labels to allow more robustness to small temporal offsets. The Wasserstein structured loss is then defined as the sum of the cumulative differences over all behaviors:

$$\mathscr{L}_W = \sum_{b=0}^{B}\sum_{i=0}^{T}\left[\sum_{j=0}^{i} y'^{j}_{b} - \sum_{j=0}^{i}\hat{y}'^{j}_{b}\right]^2 \quad . \tag{4.4}$$

The Wasserstein Loss enforces the goal of reducing false positives by linking multiple predictions. Minimizing Wasserstein Loss tends to transfer probability mass closer to sharp spike labels. This maps all network predictions to true action start locations and when multiple predictions contribute to a true action start, the loss will penalize the extra mass (Figure 4.1c). In contrast, a per-frame loss will treat spurious predictions separately, and penalize multiple predictions less aggressively.

4.3.3 Per-Frame Loss

The per frame loss, $\mathscr{L}_f$, is defiend as the mean squared error (MSE) between Y and $\hat{Y}$. A Gaussian blur is also applied to the groundtruth labels Y to allow more robustness to small temporal offsets.

$$\mathscr{L}_f = \sum_{b=0}^{B} \sum_{i=0}^{T} \left[y_b^t - \hat{y}_b^t \right]^2 \ . \tag{4.5}$$

4.3.4 Combined Loss

Similar to [WWN+18], it was helpful to combine the per-frame loss with the structured losses ($\mathscr{L}_H$ and $\mathscr{L}_W$) to improve optimization.

$$\mathscr{L}(Y,\hat{Y}) = \lambda \mathscr{L}_f(Y,\hat{Y}) + (1-\lambda)\mathscr{L}_s(Y,\hat{Y}) \tag{4.6}$$

where $\mathscr{L}_f$ is the per-frame loss, the structured loss $\mathscr{L}_s$ is either $\mathscr{L}_W$ or $\mathscr{L}_H$, and λ is a hyper parameter between 0 and 1. This is especially true for the Matching Loss, since there may not be any initial predictions that pass the network classifier score thresholding. Additionally, including the per-frame loss reduces false negatives for the Matching Loss. Decreasing the weight of the per-frame loss during training was also tested.

4.4 Visualization

A visualization tool was developed to help understand performance on video data. When debugging various network architectures, it was difficult to review an individual frame associated with a network prediction and understand what may have caused the network score. Additionally it was difficult to seek to specific frames/seconds of a video manually. In order to help visualize results, a web-based viewer, see Fig. 4.2, that synchronizes the network output score and video frame was created. The viewer has two main components. A line graph, where the x-axis is

video frames and the y-axis is the network output score. The other component is a video viewer, where the frame being shown is the currently selected frame. A frame can be selected by either playing the movie or mousing over the line graph. Being able to slowly, or quickly, mouse over consecutive video frames around curious network outputs helped find software bugs and explore network architectures.

The viewer was created using a JavaScript library called d3.js [Bos]. Network outputs are stored as a CSV file, where the first column is the frame number and any following columns are any set of scores, such as the ground truth of a label and network predictions for that label. Movie files can be any format supported by HTML5 such as MP4 or WebM.

4.5 Datasets

4.5.1 Mouse Reach Dataset

The Mouse Reach Dataset is a new video dataset which was carefully annotated with action bout starts by experts. Neuroscientists collected this dataset for studying the neural control of behavior, which involves examining recorded neural activity prior to a behavior change [GGG+15]. Unlike most action detection datasets, in which the duration of the bout is labeled, only the action start was relevant. In addition, only the action starts, not the action ends, were well-defined and could be labeled consistently (Figure 4.1), thus only action starts were annotated. The dataset can be downloaded at http://research.janelia.org/bransonlab/MouseReachData/.

The dataset contains 1165 recordings of four mice attempting to grab and eat a food pellet. Only the mice's limbs are free to move. They were recorded many times a day for several days from two fixed, time-synchronized cameras. The videos were recorded at 500 frames per second in near infrared.

The biologists labeled the start of six different actions. "Lift" occurs when the mouse begins to lift its paw from the perch. "Hand-open" occurs when the paw begins to open before

grabbing the pellet. “Grab” occurs when the paw begins to close around the pellet. “Supinate” occurs when the begins to turn toward the mouse’s mouth. “At-mouth” occurs when the pellet is first placed in the mouse’s mouth. “Chew” occurs when the mouse begins to eat the pellet. Note that grab and supinate are annotated when the mouse misses and the paw does not contain a pellet, but at-mouth and chew are not. The most common behavior is the “Hand-open” behavior with 2227 labels – on average 1.91 labeled instances per video. The least common behavior is “Chew”, with 664 labeled instances.

In these videos, the temporal sequence of actions is highly structured. For example, the mouse cannot eat a food pellet without grabbing it first. Thus, long-range temporal context is important. The most common sequence of labels is “Lift”, “Hand-open”, “Grab”, “Supinate”, “At-mouth”, and “Chew”. However, the temporal sequence is also flexible, as when the mouse misses a pellet it will backtrack and repeat actions. For example, multiple instances of grab can occur in a row. This also produces an imbalance across action categories.

This dataset will provide computer vision researchers an opportunity to work with high quality labels of action key frames. As mentioned previously, bout boundary detection has gained interest in the vision research community, and this provides a dataset for comparing and spurring algorithm development while providing useful tools for neuroscientists.

Table 4.1 and Table 4.2 provide more details on the dataset. The behaviors: Hand-open, Grab and Supinate, occur more often because the mouse will fail to grab the food pellet and try to grab food pellet again. The number of chew frames are low because the mouse will also fail to eat the food pellet. Figure 4.5 and Figure 4.6 show sample frames of each of the behaviors.

4.6 Experiments

4.6.1 Mouse Experiments

The proposed loss functions are tested on the Mouse Reach Dataset. The goal of this task is to detect the start of a behavior within τ frames. For these experiments $\tau = 10$ frames

Table 4.1. Number of labelled frames with the Mouse Reach Dataset.

Behavior	Total	Average Per Video
Lift	1175	1.01
Hand-open	2227	1.91
Grab	2096	1.79
Supinate	1392	1.19
At-mouth	921	0.79
Chew	664	0.57
Background	830939	71081

Table 4.2. The Mouse Reach Dataset contains a total of 1165 videos of mice performing the reaching task.

Mouse	Total Videos
M134	217
M147	97
M173	492
M174	359

Table 4.3. F1 scores for each loss, feature type, and behavior. Matching and Wasserstein losses outperform the per-frame MSE

Algorithm	Lift	Hand-open	Grab	Supinate	At-mouth	Chew
MSE+HOGHOF	0.79	0.78	0.84	0.65	0.44	0.45
Matching+HOGHOF	0.88	0.77	0.84	0.75	0.43	0.45
Wasserstein+HOGHOF	0.91	0.78	0.83	0.73	0.49	0.46
MSE+Canned I3D	0.81	0.77	0.83	0.68	0.46	0.41
Matching+Canned I3D	0.83	0.74	0.80	0.70	0.44	0.33
Wasserstein+Canned I3D	0.83	0.74	0.80	0.71	0.44	0.33
MSE+Finetuned I3D	0.61	0.51	0.53	0.45	0.28	0.29
Matching+Finetuned I3D	0.90	0.80	0.84	0.74	0.51	0.35
Wasserstein+Finetuned I3D	0.88	0.81	0.84	0.72	0.48	0.26
Finetuned I3D+Feedforward	0.38	0.25	0.37	0.30	0.17	0.15
ODAS	0.35	0.45	0.59	0.40	0.27	0.13

(0.02 seconds). 10 frames was chosen from a study of annotation consistency on a subset of the videos. For each of the four mice, a test mouse is chosen and training is done with all other mice's videos and the first half of the test mouse's videos. Then tested on the second half of the test mouse's videos. Training videos from the test mouse are included because, without this, for all algorithms tested, generalization across mice was poor. Test sets consisted of 125, 55, 274, and 192 videos for the four mice.

As mentioned before, a correct prediction, TP, is one that is within $\tau = 10$ frames from the ground truth frame start. These represent true positive results. All other network predictions are false positives (FP) and missed ground truth starts are false negatives (FN). The F1-Score, the harmonic mean of precision and recall, is calculated using the TP, FP, and FN.

4.6.2 THUMOS'14 Experiments

[SPC$^+$18] detects action starts in an online setting. Although the work in this chapter is not designed to be used in an online setting, their evaluation protocol can be used to test the proposed algorithm on a known action detection dataset, THUMOS'14 [JLRZ$^+$14]. [SPC$^+$18] defines point-level average precision, p-AP, as a way to evaluate the performance of start frame detection. For each class, the predicted action starts are ordered by their confidences. Each prediction is counted as a correct action start if it matches the ground truth action start class, the temporal distance between the prediction and true start is less than a threshold, and no other prediction has been matched to this start. With the matches, the point-level average precision, p-AP, can be computed and averaged over classes to compute the p-mAP.

Following [SPC$^+$18], two metrics are used to evaluate the algorithm on THUMOS'14. The first is p-mAP, computed over different temporal distance thresholds. The temporal offsets are set at every second from 1 to 10 seconds, or 30 to 300 frames. These offsets can provide insight on how precise an action start can be detected, which is useful for tuning algorithms to different applications. The second is *AP depth at recall X%* which is the average of precision points on the P-R curve between 0% and X% recall. The *AP depth at recall X%* can be useful to

evaluate the precision of top predictions at low recall.

Although THUMOS'14 has been used as a baseline of online action start detection, the Mouse Reach Dataset is better suited for action start detection. In Figure 4.3, example frames of action starts of two long jump behaviors are shown. In this example, it does not seem the same key start frame is labeled, although it is the same behavior. Because THUMOS'14 is labeled for action detection, the start of a bout may not represent the start of the action. Some labeled sequences are of highlights of the action and only show a subset of the action. For example, a long jump action clip may only show the portion of the action where the athlete jumps into the sand pit. However, the start of the action should be when the athlete begins their run towards the sand pit.

4.6.3 Implementation Details

For all experiments, pre-computed or fine-tuned features are used as inputs to an RNN. The base model is a two layer bi-directional LSTM with 256 hidden units. The inputs to the LSTM pass through a fully connected layer, ReLU, and Batch Normalization. The ouputs are transformed by a fully connected layer with a sigmoid activation layer. Fig. 4.4 shows the model. ADAM [KB14] was used for optimization. The learning rate varied by loss function. The network was trained for 400 epochs with a batch size of 10.

For the Mouse Reach dataset experiments two types of input features were used: HoG+HOF [DT05] and I3D [CZ17]. The HoG+HOF are hand-designed features that capture image gradients and motion gradients. The features were computed on overlapping windows on each view point, resulting in an 8000 dimensional feature vector. I3D is a state-of-the-art action recognition network that uses sets of RGB and optical flow frames as input. The output from the last average pooling layer before the $1 \times 1 \times 1$ convolutional classification layer was used as the I3D feature representation. For each frame in the video sequence, I3D was applied to a 64 frame window, centered around the input frame. The features used from the I3D network are 14336 dimensions. The features from the model trained on the Kinetics dataset [CZ17] will

be refered to as Canned I3D. The I3D network was also fine-tuned by training the feedforward, per-frame I3D network on the Mouse Reach Dataset. This feature set is referred to as Finetuned I3D. HoG+HOF features will be provided with the dataset.

RNNs were trained with each of the three feature types (HOGHOF, Canned I3D, and Finetuned I3D) and each of three losses: Matching (Section 4.3.1), Wasserstein (Section 4.3.2), and MSE (Mean-squared error, Section 4.3.3).

When using the HOG+HOF and Canned I3D features as inputs with the Matching Loss, $\mathscr{L}_H$, in the combined loss (4.6) the weight of the per-frame loss λ was reduced from an initial value 0.99 to 0.5 with an exponential step size of 0.9 every five epochs until $\mathscr{L}_f$ and $\mathscr{L}_s$ were weighted equally. For the Finetuned I3D features, λ was reduced to 0.25. The following parameters were set $C_{tp} = 4$, $C_{fp} = 1$, $C_{fn} = 2$. For the Wasserstein Loss, $\lambda = 0.5$. In order to help all the losses deal with the scarcity of positive samples, the ground truth label sequence was blurred with a Gaussian kernel with $\sigma_b = 2$ and $w_b = 19$ frames. Like the choice of τ, the window size was chosen based off of the annotation consistency study.

For the THUMOS'14 experiments features from an I3D network pre-trained on the Kinetics Dataset [CZ17] were created. Like the Mouse Reach experiments, outputs before the classifications layer were used as a feature representation. Unlike the Mouse Reach experiments, only the pre-trained RGB 3D convolutional branch of the I3D network were used, because the RGB branch is highly similar to the C3D network used in the ODAS work. The same hyper parameters for the losses as the Mouse Reach experiments were used, except the maximum matching offset was $\tau = 30$ frames (1 second). In the mouse videos, the researchers were interested in start detections within frames of the behavior starting, however for THUMOS'14, results were evaluated at the level of seconds. For these reasons the ground truth sequence was blurred with a kernel $w_b = 59$ frames and $\sigma_b = 8$. In addition to the bi-directional LSTM, a forward only LSTM was trained. A forward only LSTM could be used in an online fashion, which should be more comparable to the results from [SPC$^+$18].

Table 4.4. p-mAP at depth *Rec*=1 shows the performance of the proposed loss functions on THUMOS'14 at different offset thresholds. The *+FWD networks were trained as forward only LSTM's, whereas the *+BIDIR networks were bi-directional LSTM's.

Offsets	1	2	3	4	5	6	7	8	9	10
SceneDetect	0.01	0.02	0.02	0.03	0.03	0.04	0.04	0.05	0.05	0.05
ShotDetect	0.01	0.02	0.02	0.03	0.03	0.03	0.04	0.04	0.04	0.05
ODAS [SPC$^+$18]	0.03	0.04	0.04	0.05	0.05	0.06	0.06	0.07	0.07	0.08
MSE-FWD	0.06	0.10	0.11	0.12	0.13	0.14	0.15	0.15	0.15	0.15
Matching-FWD	0.06	0.10	0.11	0.12	0.13	0.14	0.15	0.15	0.15	0.15
Wasserstein-FWD	0.05	0.09	0.10	0.11	0.12	0.12	0.13	0.13	0.13	0.13
MSE+BIDIR	0.08	0.14	0.15	0.16	0.16	0.17	0.18	0.18	0.18	0.18
Matching+BIDIR	0.09	0.14	0.16	0.17	0.18	0.19	0.19	0.20	0.20	0.20
Wasserstein+BIDIR	0.05	0.09	0.11	0.12	0.12	0.13	0.13	0.14	0.14	0.14

4.6.4 Mouse Reach Results

The network trained with MSE loss will be considered as a baseline and will be compared to a networks trained with the proposed structured loss functions. RNNs trained with the MSE loss are equivalent to the most standard framework for action detection, if the bout lengths are just one frame. Table 4.6 shows the precision, recall and F1 score for each of these losses, using the HoG+HOF, pre-trained, and fine-tuned I3D features. Results of the fine-tuned feed-forward I3D network are also shown. The structured losses have a better F1 score because they have higher precision, implying an improved false positive rate. This matches one of the goals of structured losses, to penalize spurious start detections. The Matching loss explicitly penalizes false positive predictions and the Wasserstein Loss attempts to match the number of predicted behavior starts with the ground truth. Overall, the Wasserstein Loss performs best regardless of the input features. Table 4.3 shows the performance breakdown with respect to each behavior. The action categories that benefited most from the structured losses were lift and supinate.

The MSE+Finetuned I3D performs far worse than expected and it maybe due to finetuned feature training. Only the labeled frames were used as positive samples. No frames in a window with radius 10 around the positive sample are sampled. Additionally no hard negatives at the border of this window were used for training. Because MSE does not penalize false positive

Table 4.5. Average p-mAP at different depths on the THUMOS'14 dataset.

Depth@X	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	10
SceneDetect	0.30	0.18	0.12	0.09	0.07	0.06	0.05	0.04	0.4	0.03
ShotDetect	0.26	0.15	0.11	0.08	0.07	0.06	0.05	0.04	0.04	0.03
ODAS [SPC$^+$18]	0.42	0.27	0.19	0.14	0.11	0.10	0.08	0.07	0.06	0.05
MSE+FWD	0.55	0.48	0.46	0.45	0.45	0.45	0.44	0.44	0.44	0.44
Matching+FWD	0.64	0.57	0.55	0.54	0.54	0.53	0.53	0.53	0.53	0.53
Wasserstein+FWD	0.58	0.55	0.53	0.53	0.53	0.53	0.53	0.53	0.53	0.53
MSE+BIDIR	0.46	0.36	0.30	0.26	0.24	0.24	0.23	0.23	0.23	0.23
Matching+BIDIR	0.59	0.52	0.48	0.45	0.44	0.43	0.42	0.42	0.42	0.42
Wasserstein+BIDIR	0.42	0.35	0.33	0.31	0.30	0.29	0.29	0.29	0.29	0.29

as harshly as the structured losses, it is happy to predict many extra action starts due to similar feature representation of neighboring frames, as seen by the poor precision score. It is possible the feature representation and the MSE performance can be improved by implementing hard negative mining.

Figures 4.7, 4.8, 4.9, 4.10, 4.11, and 4.12 show the distribution of predictions for each behavior. Within the $\tau = 10$ frames of the true start, the structured losses predict far fewer false positives without missing true positives. Thus, the losses are more likely to produce a single prediction for an action start than MSE. Figure 4.13 shows how predictions far from the action start eventually are matched. Perhaps surprisingly, there does not seem a big difference in performance between HOG/HOF features and the fine-tuned I3D features learned for this task.

At the time of this writing, source code and the trained model for the ODAS [SPC$^+$18] algorithm was unavailable, so reimplementation their algorithm was used. To make comparisons as fair as possible, the I3D network was used as the backbone network, instead of the reported C3D. The network was trained with adaptive sampling and temporal consistency, but without a GAN to generate hard negatives. The implementation of ODAS without a GAN model was slightly better than the I3D+Feedforward model. However the recurrent models all outperform the implementation of ODAS.

Table 4.6. For each loss and feature type, the F-score, precision and recall are reported. The Matching and Wasserstein Losses have an improved F-score and precision over MSE, implying fewer false positives.

Algorithm	F1 Score	Precision	Recall
MSE+HOGHOF	0.69	0.62	0.79
Matching+HOGHOF	0.73	0.72	0.74
Wasserstein+HOGHOF	0.75	0.78	0.71
MSE+Canned I3D	0.70	0.64	0.76
Matching+Canned I3D	0.69	0.72	0.66
Wasserstein+Canned I3D	0.73	0.77	0.70
MSE+Finetune I3D	0.48	0.37	0.69
Matching+Finetune I3D	0.75	0.80	0.70
Wasserstein+Finetune I3D	0.75	0.85	0.66
I3D+Feedforward	0.27	0.16	0.88
ODAS	0.22	0.12	0.94

4.6.5 THUMOS'14 Results

Table 4.4 shows the p-mAP performance of the algorithms with different offset thresholds. The Matching+BIDIR algorithm matches or outperforms all other algorithms for each threshold offset. Interestingly, the Wasserstein+BIDIR loss performs worse then both MSE+BIDIR and Matching+BIDIR on THUMOS'14. Across each cost function, using bidirectional networks provides about a 0.02 improvement at most threshold offsets. Forward only for the Mouse Reach dataset will be tested for future work.

The baseline MSE+FWD algorithm performs better than ODAS. It's possible that lack of GAN hard negative training is causing the baseline forward algorithms to perform better than ODAS. As mentioned previously label balance was improved through sampling positive samples more often. Applying the label re-weighting may help in this example as well.

The average p-mAP *at depth X%* results are shown in Table 4.5. All of the algorithms perform better than ODAS at each depth. Interestingly, the FWD only algorithms perform worse at each offset threshold, but they perform better on the average p-mAP at each depth.

4.7 Discussion

In this work an algorithm for predicting the frame a behavior begins was devleoped and a new dataset for action start detection research, the Mouse Reach Dataset, was provided. Due to the nature of the task, a structured loss functions that reduces the number of false positive predictions was designed. The two different losses perform better than the baseline on the Mouse Reach Dataset. However on the THUMOS'14 dataset the Matching loss performed the best. In the future tests on the importance of τ and smoothing of the ground truth labels are planned. By modifying τ or the size of the Gaussian kernel on the ground truth labels, the temporal localization accuracy can be adjusted.

The model will also be compared to existing techniques used by biologists for automatically detecting the start of behaviors. Currently biologists use JAABA [KRRA$^+$13] for detecting action starts. Because JAABA uses a different type of labels, it is not trivial to make direct comparisons between action start detection and bout start detection. Additionally JAABA does not require every bout to be labeled, but the work in this chapter requires every start to be labeled. Properly comparing the different algorithms will require some work.

Finally behavior discovery will be investigated. Currently labels created by humans are being used, but there is no guarantee that these behaviors are connected to neurological events. Ideally, using SNaCK 3 and learned models of video could be combined. By combining SNaCK embeddings with the video analysis described in this chapter, it may be possible to explore mouse behavior space.

4.8 Acknowledgments

Chapter 4 is based on "Detecting the Starting Frame of Actions in Video," I. S. Kwak, D. Kriegman, K. Branson and is currently being prepared for submission for publication of the material. The dissertation author was the primary investigator and author of this paper.

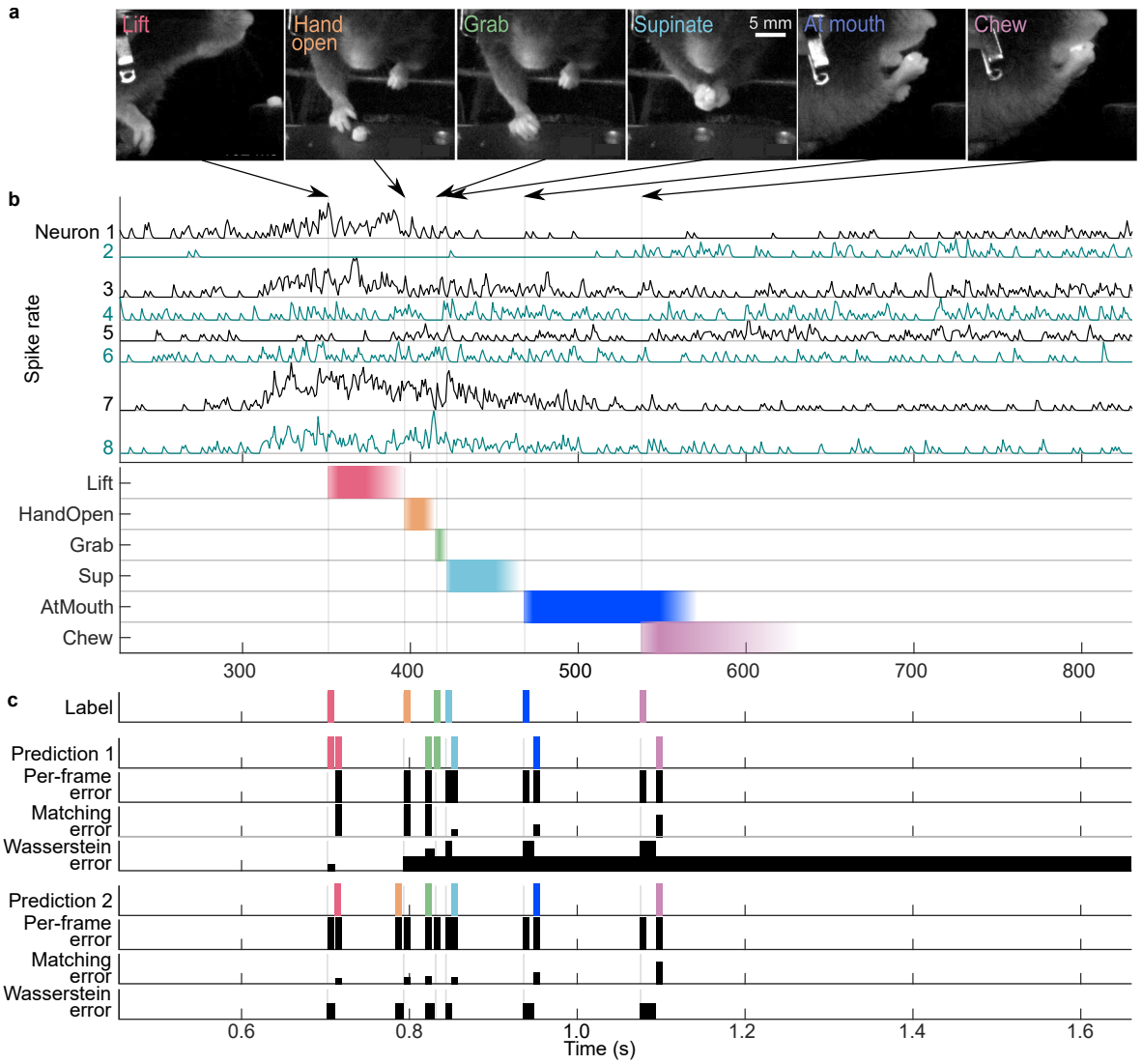


Figure 4.1. (a) In this neuroscience experiment, a mouse has been trained to reach for a food pellet. This movement consists of a sequence of actions: lift, hand-open, grab, supinate, pellet at-mouth, and chewing [GGG+15]. (b) A fundamental goal in systems neuroscience is to associate patterns of neural activity (top) with the behaviors it causes (bottom), e.g. spiking in several of the recorded cortical neurons precedes the onset of lift. Colors indicate different behaviors, and saturation indicates annotator confidence. Confidence changes are sharper at action starts than ends, as starts are usually associated with large accelerations, e.g. pinpointing the start of a lift is much easier than pinpointing its end. (c) Given a sequence of labeled frames (Labels), a per-frame loss prefers multiple or missed detections (Prediction 1) to a small temporal offset in the predictions (Prediction 2). The structured losses proposed in this work are designed to instead heavily penalize extra or missed detections. Error plots (black) show the error accrued on each frame.

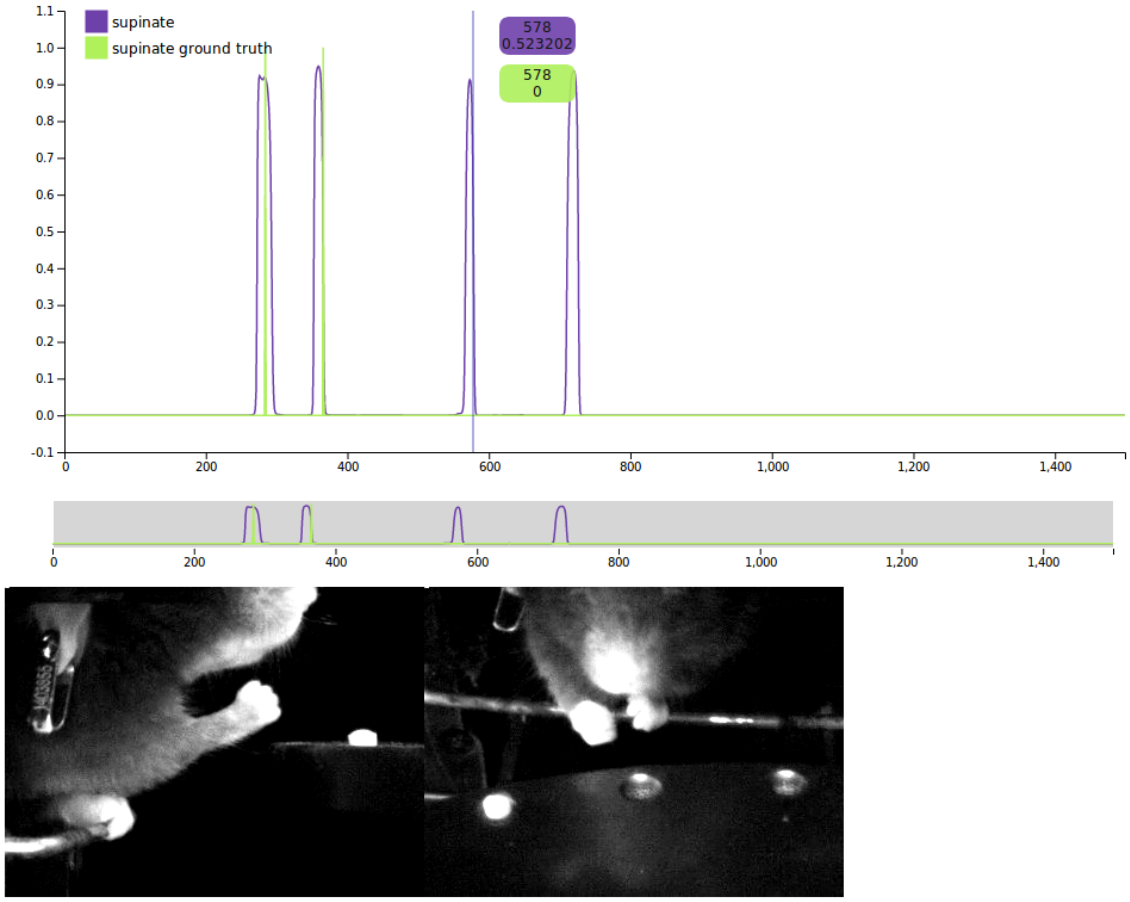


Figure 4.2. An example screen shot of the web based network output viewer for videos. The green line is ground truth and purple is the network's predictions. The mouse can hover over the frames that caused the false positive predictions. The vertical blue line near 600 frames denotes the current visible frame in the video. The purple and green rectangles shows the frame number and scores of that frame. In this case, frame 578 is being viewed and the ground truth supinate score is 0 and the network prediction of the behavior is 0.52. The side of the mouse paw can be seen, which is something visible in the mouse supinate behavior, but the paw is quite far from the food pellet.

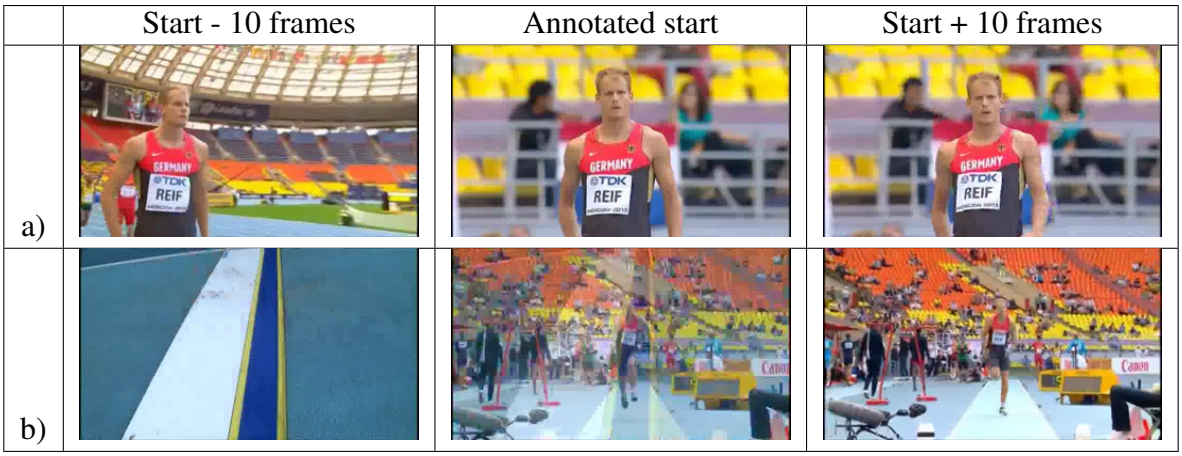


Figure 4.3. Examples of labeled action starts from THUMOS'14 showing ambiguity in the annotations. Each row shows annotated long jump starts (middle column), 10 frames before the start (left column), and 10 frames after (right column). The frames representing the start of the long-jump behavior do not seem to correspond to the same key frame in these examples. Row a) shows an example without scene transitions, and the action is annotated to start when the athlete leans back before running. Row b) includes a scene transition to a replay of the action. The annotated action start is after the athlete has already started his run. These examples show very different action start frames, and thus the proposed dataset is a better test of key frame detection.

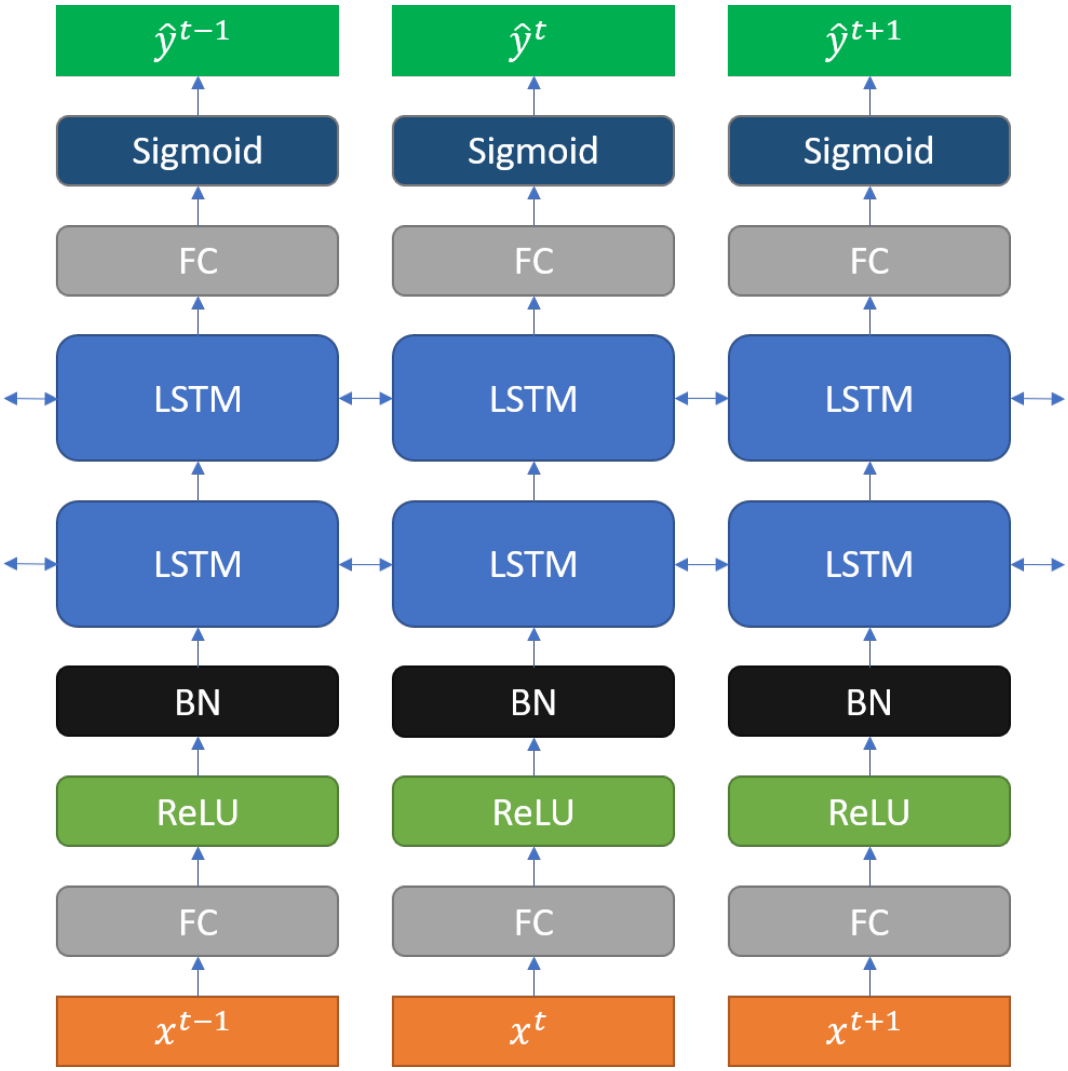


Figure 4.4. The complete model consists of a fully connected layer, ReLU, Batch Normalization, two Bi-directional LSTM layers, a fully connected layer then a sigmoid activation layer. The LSTMs each have 256 hidden units.

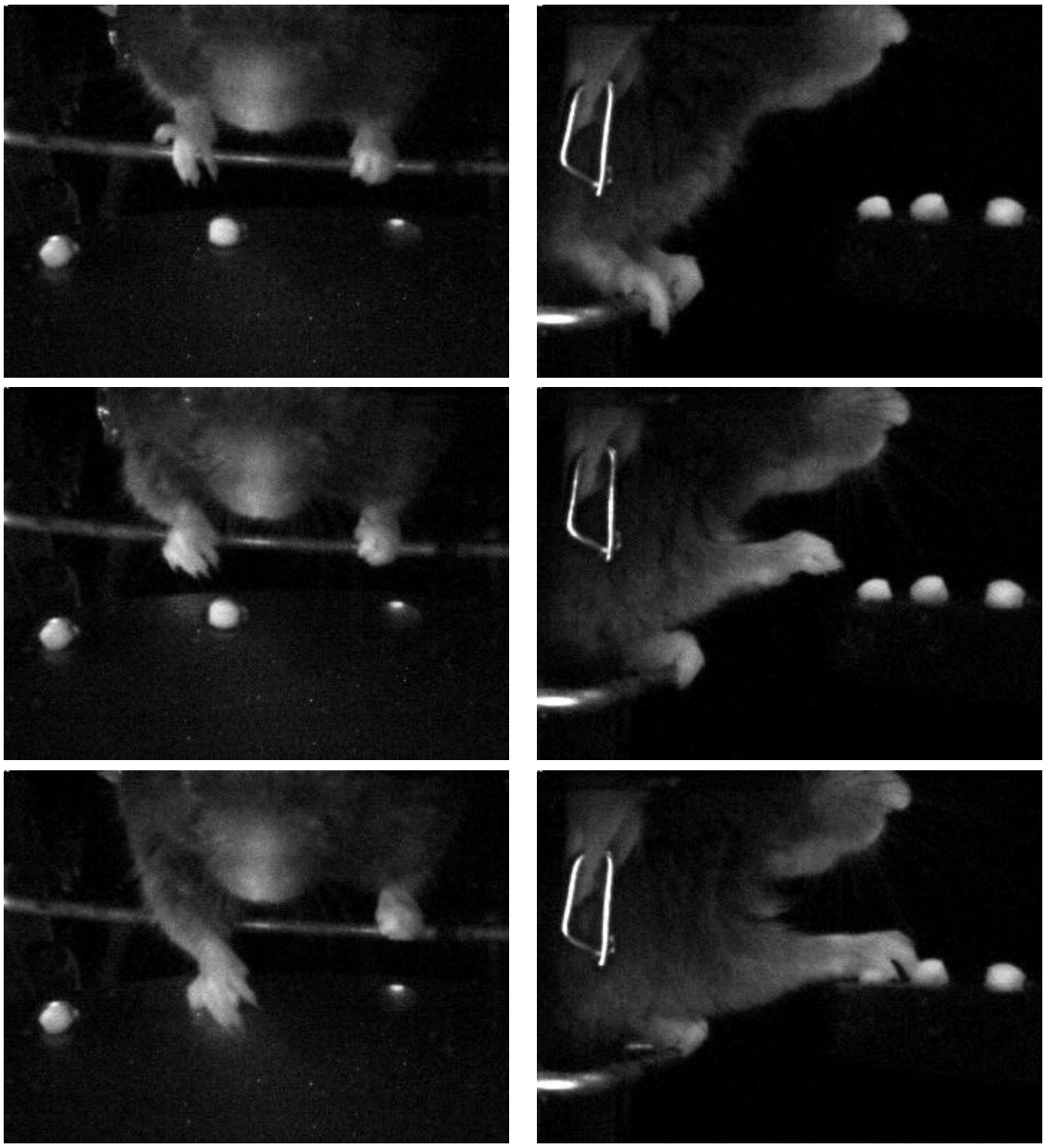


Figure 4.5. Example frames of behaviors the Mouse Reach Dataset. The first row is the lift behavior. Here the mouse paw is beginning to move off of the perch. The next row is the Hand-open behavior. Here is the mouse beginning to open his paw to grab a pellet. The third row is the grab behavior. The mouse beginning to close his paw around a food pellet.

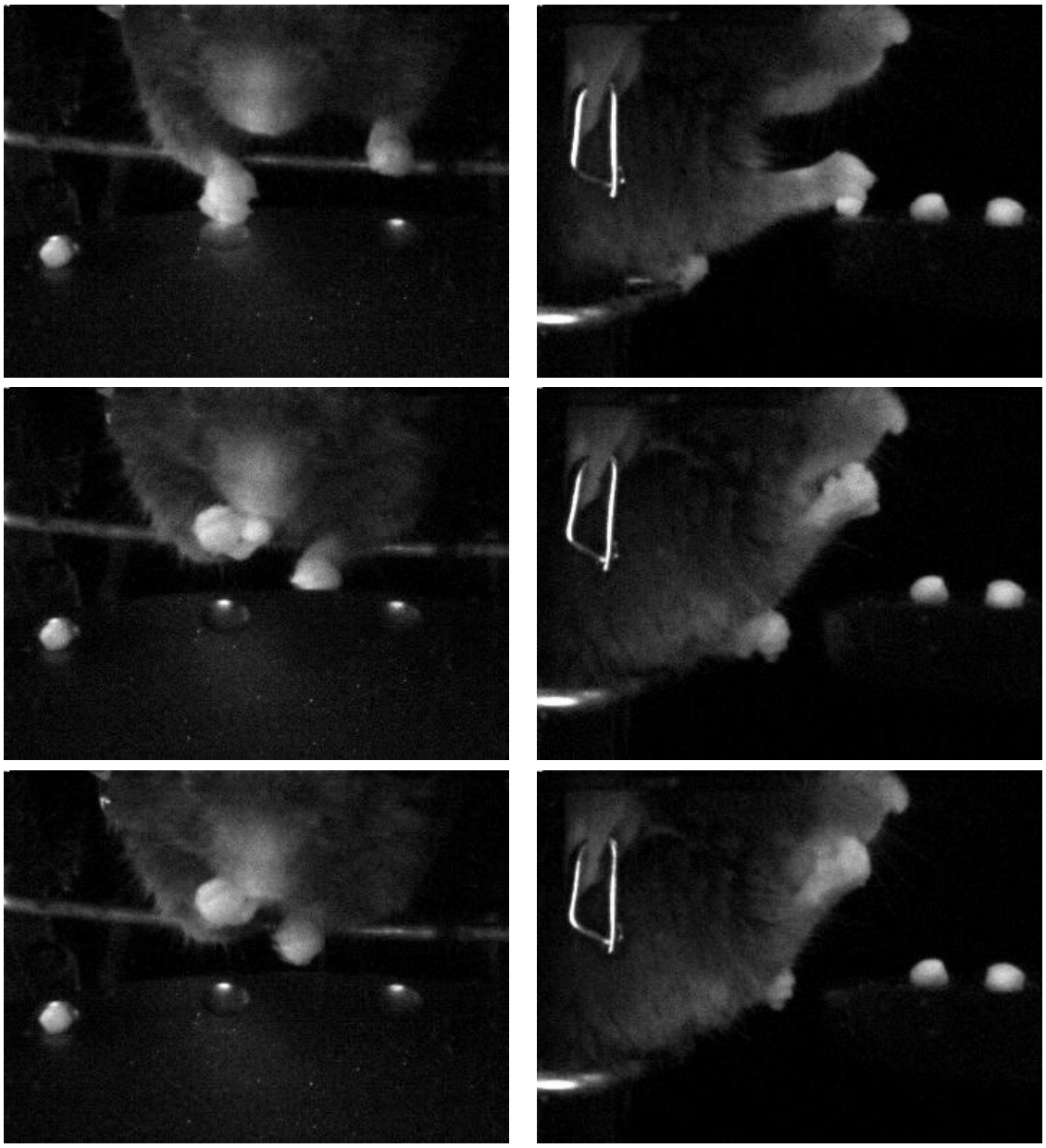


Figure 4.6. The supinate behavior is shown in the first row. The mouse is beginning to turn its paw towards its mouth. The second row shows the At-mouth behavior. The At-mouth behavior occurs when the food pellet is starting to be placed into the mouth. The last row shows the chew behavior, where the food pellet in the mouth and the mouse is starting to eat the pellet.

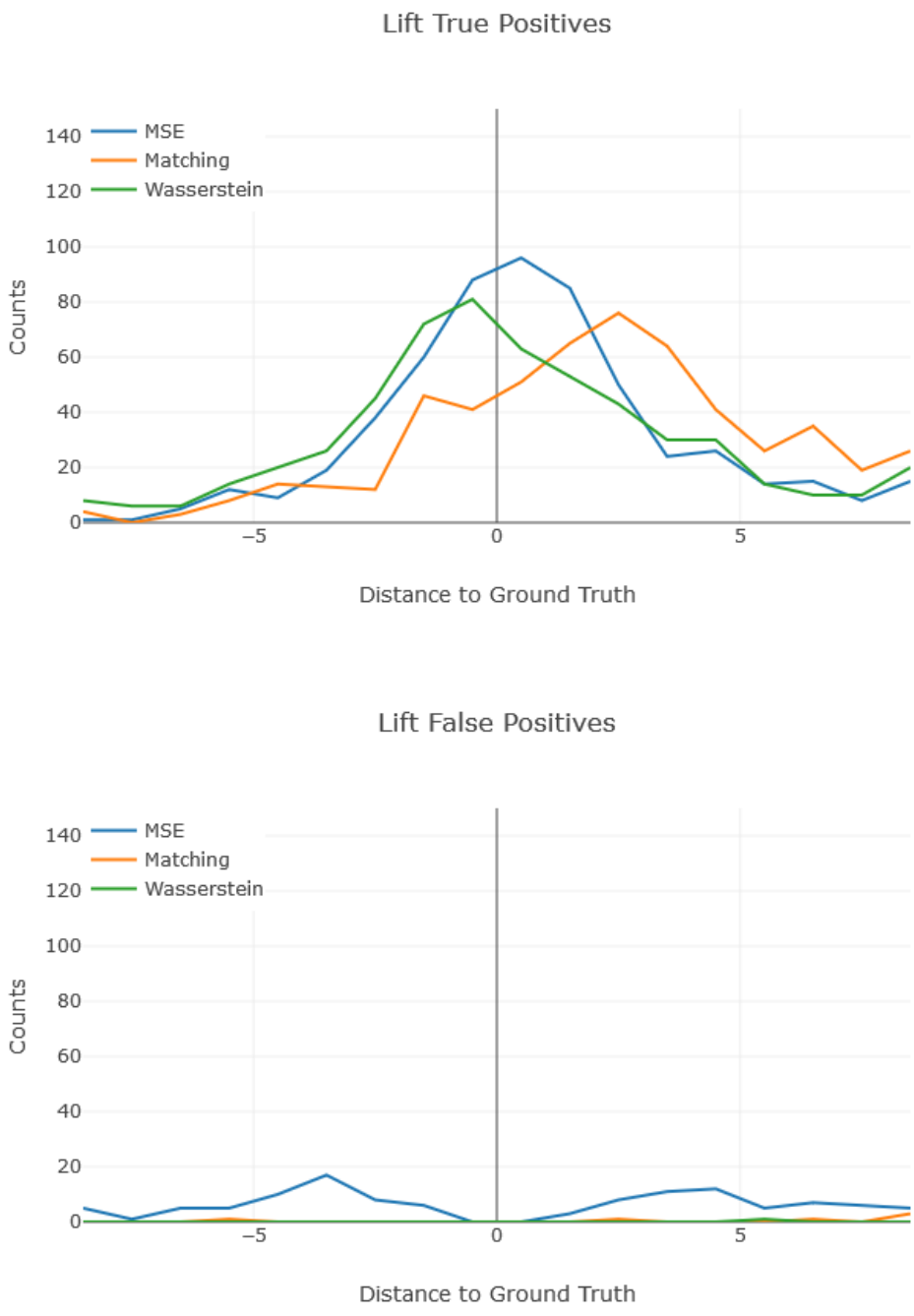


Figure 4.7. Each of these graphs shows the number of predictions at a certain frame distance from the ground truth behavior location for the Lift behavior. The networks were trained with HOGHOF features. The top shows the distribution of true positives and the bottom the false positives.

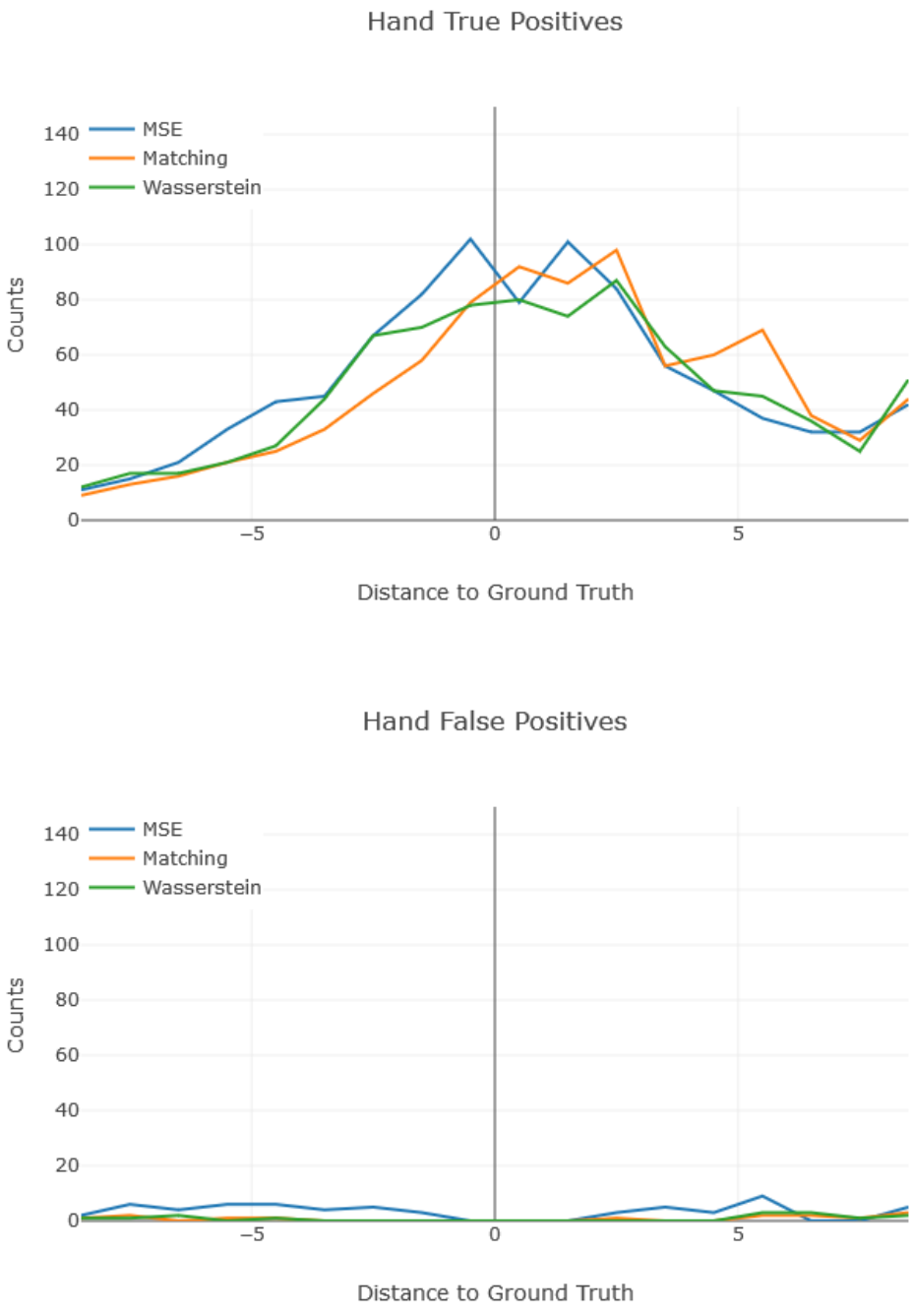


Figure 4.8. These graphs show the distribution of predictions for the Hand-open behavior.

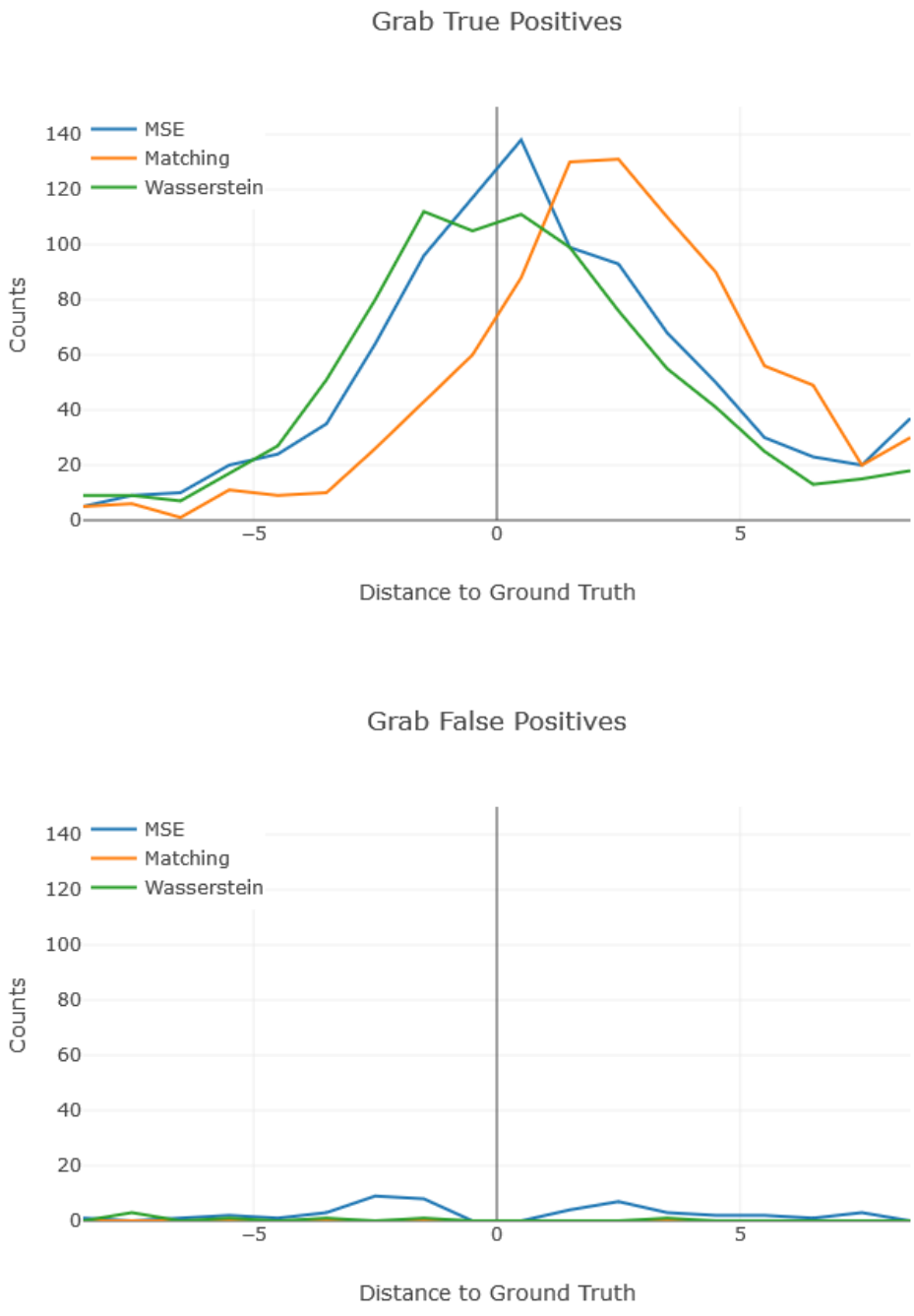


Figure 4.9. These graphs show the distribution of predictions for the Grab behavior.

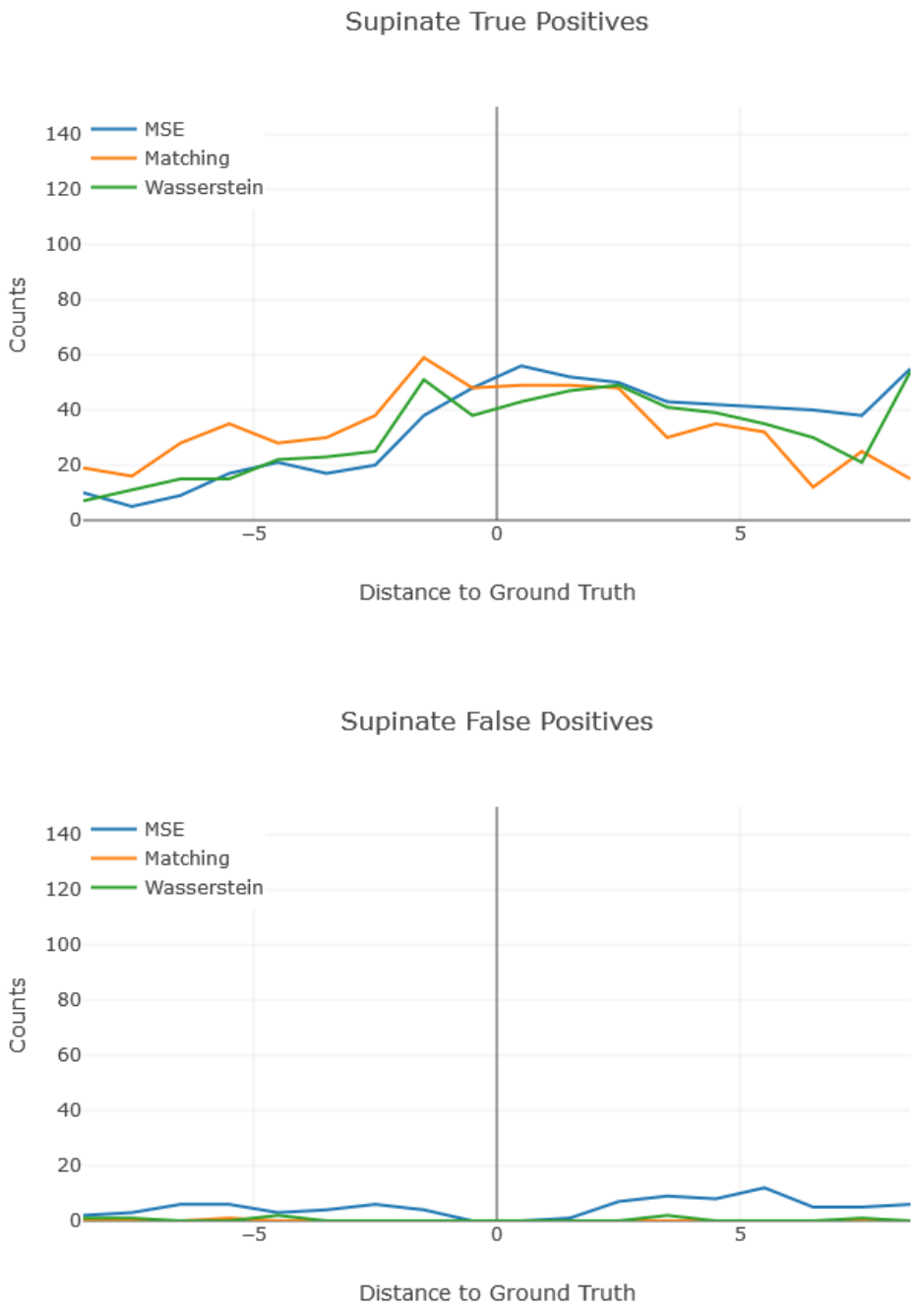


Figure 4.10. These graphs show the distribution of predictions for the Supinate behavior.

Figure 4.11. These graphs show the distribution of predictions for the At-mouth behavior.

Figure 4.12. These graphs show the distribution of predictions for the Chew behavior.

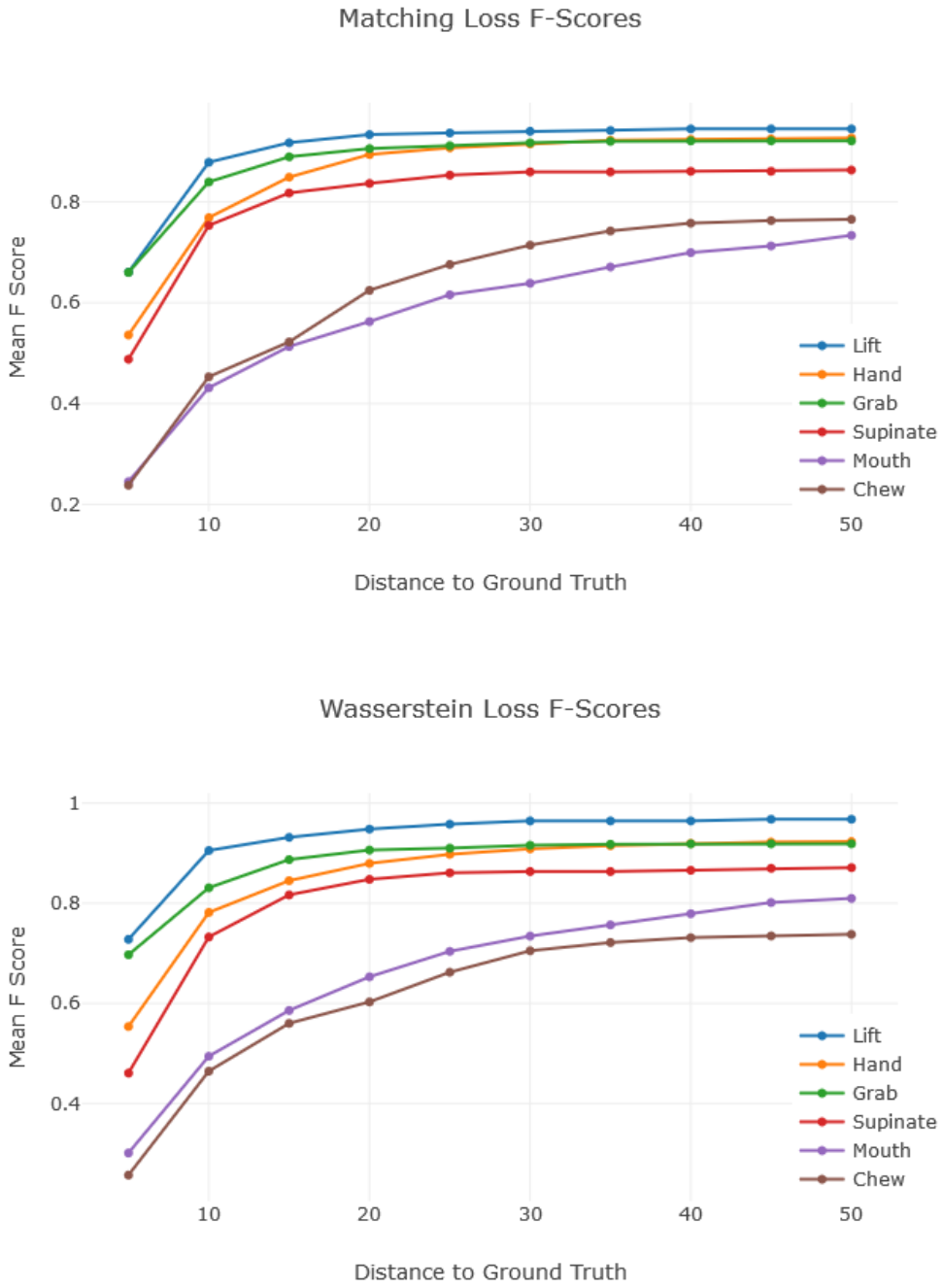


Figure 4.13. In all the other results in this paper, a match is considered correct when the predicted behavior is within ten frames of the true action start. Here τ is varied between 5 and 50 frames. The Matching loss F-score is shown on the top and the Wasserstein Loss is bottom. The networks were not retrained for these results, but simply re-analyzed. For the first 3 behaviors, the network detects the start within twenty frames. However, for the At-mouth and Chew behaviors, the network needs a much larger window.

Chapter 5

Conclusion

This thesis presents three different topics, each exploring parts of the supervised pipeline and showing improved results by tailoring parts of the pipeline to the needs of the application. Chapter 2 explores a new problem space, and a new dataset designed for the problem was created. In chapter 3, the process of the dataset labeling was explored, which improved the quality of constructed perceptual embeddings. Finally, Chapter 4 focuses on a problem that is not well studied. The quality of results was improved drastically with the creation of a well-designed dataset and algorithm for the problem.

Chapter 2 introduced the problem of urban tribe classification. This chapter introduced a technique for modeling the scene and group of individuals for the purpose of social category classification. The importance of the modeling group structure was explored and shown to be important for the task. Additionally, with deep learning, future work corroborated the intuition discovered in Chapter 2. The models in Chapter 2 were tested on a novel dataset collected for the task.

In Chapter 3, triplet constraint collection for building perceptual embeddings was explored. UI modifications for collecting triplets were tested both synthetically and with human mechanical turkers. The UI improvements show that with efficient triplet collection high quality perceptual embeddings can be created. In addition to the changes to the UI, chapter 3 introduces a method for combining a learned machine kernel with human constraints for reducing the

number of constraints needed and improving the quality of the embedding. The improvements in the algorithm are shown quantitatively through a variety of tasks such as discovering mislabeled birds and new categories. This work also provides a large food dataset of human taste similarities.

Finally, the precise localization of key frames of an action is explored. The work in this chapter focuses on detecting the exact starting frame of a behavior, which is a previously, but not heavily researched task. A new structured loss, that penalizes extra and missed action starts heavily, was designed for this task. This loss was optimized with recurrent neural networks (RNNs) and was compared to existing approaches. The model and loss were tested both on an existing dataset as well as a newly collected dataset. Past work relied on evaluating performance on datasets ill-suited for action onset detection. The new dataset, the Mouse Reach Dataset, is more suited for the task of action start detection because experts have labeled only the starting frame of a behavior. On both datasets, the proposed model outperforms related approaches and baseline methods using an unstructured loss.

Bibliography

[AHSG17] Ziad Al-Halah, Rainer Stiefelhagen, and Kristen Grauman. Fashion forward: Forecasting visual style in fashion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 388–397, 2017.

[Arm] R. Armburst. *Capturing Growth: Photo Apps and Open Graph*. http://developers.facebook.com/blog/post/2012/07/17/capturing-growth--photo-apps-and-open-graph/.

[AWC+07] Sameer Agarwal, Josh Wills, Lawrence Cayton, Gert Lanckriet, David J Kriegman, and Serge Belongie. Generalized non-metric multidimensional scaling. In *International Conference on Artificial Intelligence and Statistics*, 2007.

[BBM04] Mikhail Bilenko, Sugato Basu, and Raymond J. Mooney. Integrating constraints and metric learning in semi-supervised clustering. In *ICML*. ACM, 2004.

[BDR06] Yoshua Bengio, Olivier Delalleau, and Nicolas Le Roux. *Label Propagation and Quadratic Criterion*, pages 193–216. MIT Press, 2006.

[BEK+12] Oscar Beijbom, Peter J Edmunds, David I Kline, B Greg Mitchell, and David Kriegman. Automated annotation of coral reef survey images. In *CVPR*. IEEE, 2012.

[BGVG14] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–Mining Discriminative Components with Random Forests. In *ECCV*. Springer, 2014.

[BHBP14] Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. In *BMVC*, Nottingham, 2014.

[BHW+14] Steve Branson, Grant Van Horn, Catherine Wah, Pietro Perona, and Serge Belongie. The Ignorant Led by the Blind: A Hybrid Human-Machine Vision System for Fine-Grained Categorization. *IJCV*, 108(1-2):3–29, February 2014.

[BM09] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In *ICCV*, 2009.

[BMM11] L. Bourdev, S. Maji, and J. Malik. Describing people: Poselet-based attribute classification. In *ICCV*, 2011.

[Bos] Mike Bostock. *D3.js*.

[BS02] Mukund Balasubramanian and Eric L Schwartz. The isomap algorithm and topological stability. *Science*, 295(5552):7–7, 2002.

[Car12] M. Carroll. How Tumblr and Pinterest are fueling the image intelligence problem. *Forbes*, January 17 2012. Web http://onforb.es/yEfDmM.

[CBC+10] D. J. Crandall, L. Backstrom, D. Cosley, S. Suri, D. Huttenlocher, and J. Kleinberg. Inferring social ties from geographic coincidences. *PNAS*, 2010.

[CGG12] H. Chen, A. Gallagher, and B. Girod. Describing clothing by semantic attributes. In *ECCV*, 2012.

[CGZ+16] De Cheng, Yihong Gong, Sanping Zhou, Jinjun Wang, and Nanning Zheng. Person re-identification by multi-channel parts-based cnn with improved triplet loss function. In *Proceedings of the iEEE conference on computer vision and pattern recognition*, pages 1335–1344, 2016.

[CHEGCN15] Fabian Caba Heilbron, Victor Escorcia, Bernard Ghanem, and Juan Carlos Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015.

[CL11] C. Chang and C. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2011. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[CZ17] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017.

[DBH14] C.D. Demiralp, M.S. Bernstein, and J. Heer. Learning Perceptual Kernels for Visualization Design. *IEEE Trans. on Visualization and Computer Graphics*, 20(12):1933–1942, December 2014.

[DCSH15] Qieyun Dai, Peter Carr, Leonid Sigal, and Derek Hoiem. Family member identification from photo collections. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 982–989. IEEE, 2015.

[DJRG12] A. Dhall, J. Joshi, I. Radwan, and R. Goecke. Finding happiest moments in a social context. In *ACCV*, 2012.

[DSK+14] C. Demiralp, C.E. Scheidegger, G.L. Kindlmann, D.H. Laidlaw, and J. Heer. Visual Embedding: A Model for Visualization. *IEEE Computer Graphics and Applications*, 34(1):10–15, January 2014.

[DT05] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005.

[DY11] L. Ding and A. Yilmaz. Inferring social relations from visual concepts. *ICCV*, 2011.

[EHNG16] Victor Escorcia, Fabian Caba Heilbron, Juan Carlos Niebles, and Bernard Ghanem. Daps: Deep action proposals for action understanding. In *ECCV*, 2016.

[FG09] M. Ferecatu and D. Geman. A statistical framework for image category search from a mental picture. *IEEE TPAMI*, June 2009.

[FOZ+11] Ryan Farrell, Om Oza, Ning Zhang, Vlad I. Morariu, Trevor Darrell, and Larry S. Davis. Birdlets: Subordinate categorization using volumetric primitives and pose-normalized appearance. In *ICCV*, 2011.

[FSSM07] A. Frome, Y. Singer, F. Sha, and J. Malik. Learning globally-consistent local distance functions for shape-based image retrieval and classification. In *IEEE ICCV*, 2007.

[GC09] A. Gallagher and T. Chen. Understanding images of groups of people. In *CVPR*, 2009.

[GGG+15] Jian-Zhong Guo, Austin R Graves, Wendy W Guo, Jihong Zheng, Allen Lee, Juan Rodriguez-Gonzalez, Nuo Li, John J Macklin, James W Phillips, Brett D Mensh, et al. Cortex commands the performance of skilled movement. *Elife*, 2015.

[GWKP11] Ryan Gomes, Peter Welinder, Andreas Krause, and Pietro Perona. Crowdclustering. *NIPS*, 2011.

[GYC+17] Jiyang Gao, Zhenheng Yang, Kan Chen, Chen Sun, and Ram Nevatia. Turn tap: Temporal unit regression network for temporal action proposals. In *ICCV*, 2017.

[HBL17] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017.

[HBSH15] Eric Heim, Matthew Berger, Lee M. Seversky, and Milos Hauskrecht. Efficient Online Relative Comparison Kernel Learning. *arXiv preprint arXiv:1501.01242*, 2015.

[HKHL+15] M Hadi Kiapour, Xufeng Han, Svetlana Lazebnik, Alexander C Berg, and Tamara L Berg. Where to buy it: Matching street clothing photos in online shops. In *Proceedings of the IEEE international conference on computer vision*, pages 3343–3351, 2015.

[HWJD17] Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S Davis. Learning fashion compatibility with bidirectional lstms. In *Proceedings of the 25th ACM international conference on Multimedia*, pages 1078–1086. ACM, 2017.

[IZJ+17] Haroon Idrees, Amir R Zamir, Yu-Gang Jiang, Alex Gorban, Ivan Laptev, Rahul Sukthankar, and Mubarak Shah. The thumos challenge on action recognition for videos "in the wild". *Computer Vision and Image Understanding*, 2017.

[JGC+10] X. Jin, A. Gallagher, L. Cao, J. Luo, and J. Han. The Wisdom of Social Multimedia: Using Flickr for Prediction and Forecast. In *ACM Multimedia Int. Conf.*, 2010.

[JLRZ+14] Y.-G. Jiang, J. Liu, A. Roshan Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar. THUMOS challenge: Action recognition with a large number of classes, 2014.

[JN11] K.G. Jamieson and R.D. Nowak. Low-dimensional embedding using adaptively selected ordinal data. In *Allerton Conference on Communication, Control, and Computing*, 2011.

[JSD+14] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[KB14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[Ken48] Maurice George Kendall. *Rank correlation methods.* Griffin, 1948.

[KHX+19] John W Krakauer, Alkis M Hadjiosif, Jing Xu, Aaron L Wong, and Adrian M Haith. Motor learning. *Comprehensive Physiology*, 9:613–663, 2019.

[KMB+13] Iljung S Kwak, Ana Cristina Murillo, Peter N Belhumeur, David J Kriegman, and Serge J Belongie. From bikers to surfers: Visual recognition of urban tribes. In *BMVC*, volume 1, page 2, 2013.

[KRRA+13] Mayank Kabra, Alice A Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. Jaaba: interactive machine learning for automatic annotation of animal behavior. *Nature methods*, 10(1):64, 2013.

[Kru64] Joseph B Kruskal. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964.

[KSH12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[KTS+14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014.

[Kuh55] Harold W. Kuhn. The Hungarian Method for the assignment problem. *Naval Research Logistics Quarterly*, 1955.

[KvL14] Matthus Kleindessner and Ulrike von Luxburg. Uniqueness of Ordinal Embedding. *JMLR*, 2014.

[KW78] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*, volume 11. Sage, 1978.

[KYBB14] M Hadi Kiapour, Kota Yamaguchi, Alexander C Berg, and Tamara L Berg. Hipster wars: Discovering elements of fashion styles. In *European conference on computer vision*, pages 472–488. Springer, 2014.

[LCY13] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv:1312.4400 [cs]*, December 2013. arXiv: 1312.4400.

[LG11a] Yong Jae Lee and K. Grauman. Learning the easy things first: Self-paced visual category discovery. In *CVPR*, pages 1721–1728, June 2011.

[LG11b] Yong Jae Lee and Kristen Grauman. Face discovery with social context. In *BMVC*, pages 1–11, 2011.

[LHLF15] Tianqiang Liu, Aaron Hertzmann, Wilmot Li, and Thomas Funkhouser. Style compatibility for 3d furniture models. *ACM Transactions on Graphics (TOG)*, 34(4):85, 2015.

[LLQ+16] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1096–1104, 2016.

[LLWT15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. Deep learning face attributes in the wild. In *Proceedings of the IEEE international conference on computer vision*, pages 3730–3738, 2015.

[LSL+12] S. Liu, Z. Song, G. Liu, C. Xu, H. Lu, and S. Yan. Street-to-shop: Cross-scenario clothing retrieval via parts alignment and auxiliary set. In *CVPR*, 2012.

[Maf96] M. Maffesoli. *The Time of the Tribes: The Decline of Individualism in Mass Society*. Sage Publications, 1996.

[MCCD13] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[McF12] Brian McFee. *More like this: machine learning approaches to music similarity*. PhD thesis, University of California, San Diego, May 2012.

[Mil56] George A Miller. The magical number seven, plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956.

[MKB+12] A. C. Murillo, I. S. Kwak, L. Bourdev, D. Kriegman, and S. Belongie. Urban tribes: Analyzing group photos from a social perspective. In *CVPR Workshop on Socially Intelligent Surveillance and Monitoring (SISM)*, 2012.

[MKBK11] O. K. Manyam, N. Kumar, P. N. Belhumeur, and D. Kriegman. Two faces are better than one: Face recognition in group photographs. In *Int. Joint Conference on Biometrics (IJCB)*, 2011.

[MTSVDH15] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–52. ACM, 2015.

[OT01] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *IJVC*, 2001.

[PC+19] Gabriel Peyré, Marco Cuturi, et al. Computational optimal transport. *Foundations and Trends® in Machine Learning*, 11, 2019.

[PG11] D. Parik and K. Grauman. Relative Attributes. In *ICCV*, 2011.

[PMB18] Omid Poursaeed, Tomáš Matera, and Serge Belongie. Vision-based real estate price estimation. *Machine Vision and Applications*, 29(4):667–676, 2018.

[RDS+14] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Li Fei-Fei. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.

[RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, 2015.

[SAN16] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016.

[SCZ+17] Zheng Shou, Jonathan Chan, Alireza Zareian, Kazuyuki Miyazawa, and Shih-Fu Chang. Cdc: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In *CVPR*, 2017.

[SGCC13] Henry Shu, Andrew Gallagher, Huizhong Chen, and Tsuhan Chen. Face-graph matching for classifying groups of people. In *2013 IEEE International Conference on Image Processing*, pages 2425–2429. IEEE, 2013.

[SGZ+18] Britton Sauerbrei, Jian-Zhong Guo, Jihong Zheng, Wendy Guo, Mayank Kabra, Nakul Verma, Kristin Branson, and Adam Hantman. The cortical dynamics orchestrating skilled prehension. *bioRxiv*, page 266320, 2018.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin. FaceNet: A Unified Embedding for Face Recognition and Clustering. *arXiv:1503.03832 [cs]*, March 2015. arXiv: 1503.03832.

[SLF13] Ming Shao, Liangyue Li, and Yun Fu. What do you do? occupation recognition in a photo via social context. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3631–3638, 2013.

[SLJ+14] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going Deeper with Convolutions. *arXiv:1409.4842 [cs]*, September 2014. arXiv: 1409.4842.

[SMJ+16] Bharat Singh, Tim K Marks, Michael Jones, Oncel Tuzel, and Ming Shao. A multi-stream bi-directional recurrent neural network for fine-grained action detection. In *CVPR*, 2016.

[SPC+18] Zheng Shou, Junting Pan, Jonathan Chan, Kazuyuki Miyazawa, Hassan Mansour, Anthony Vetro, Xavier Giro-i Nieto, and Shih-Fu Chang. Online detection of action start in untrimmed, streaming videos. In *ECCV*, 2018.

[SSF17] Qianru Sun, Bernt Schiele, and Mario Fritz. A domain based approach to social relation recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3481–3490, 2017.

[SWHY11] Z. Song, M. Wang, X. Hua, and S. Yan. Predicting occupation via human clothing and contexts. *ICCV*, 2011.

[SZ14] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014.

[SZS12] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*, 2012.

[TBF+15] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *ICCV*, 2015.

[TLB+11] Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively learning the crowd kernel. In *ICML*, 2011.

[UVDSGS13] Jasper RR Uijlings, Koen EA Van De Sande, Theo Gevers, and Arnold WM Smeulders. Selective search for object recognition. *IJCV*, 2013.

[VBK17] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional similarity networks. *Computer Vision and Pattern Recognition (CVPR)*, 2017.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *JMLR*, 9(2579-2605):85, 2008.

[vdMW12a] L. van der Maaten and K. Weinberger. Stochastic triplet embedding. In *IEEE MLSP*, 2012.

[VdMW12b] Laurens Van der Maaten and K. Weinberger. Stochastic triplet embedding. In *IEEE Int. Workshop on Machine Learning for Signal Processing*, 2012.

[VMN+16] Andreas Veit, Tomas Matera, Lukas Neumann, Jiri Matas, and Serge Belongie. Coco-text: Dataset and benchmark for text detection and recognition in natural images. *arXiv preprint arXiv:1601.07140*, 2016.

[VPB09] A. Vinciarelli, M. Pantic, and H. Bourlard. Social signal processing: Survey of an emerging domain. *Image and Vision Computing*, 2009.

[VSP+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, 2017.

[WBW+11] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011.

[WC15] Yufei Wang and Garrison W Cottrell. Bikers are like tobacco shops, formal dressers are like suits: Recognizing urban tribes with caffe. In *2015 IEEE Winter Conference on Applications of Computer Vision*, pages 876–883. IEEE, 2015.

[WHB+14a] Catherine Wah, G. V. Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. Similarity Comparisons for Interactive Fine-Grained Categorization. *CVPR*, 2014.

[WHB+14b] Catherine Wah, Grant Van Horn, Steve Branson, Subhransu Maji, Pietro Perona, and Serge Belongie. Similarity comparisons for interactive fine-grained categorization. In *IEEE CVPR*, 2014.

[Wic03] Florian Wickelmaier. An introduction to mds. *Sound Quality Research Unit, Aalborg University, Denmark*, 2003.

[WKB14a] Michael J Wilber, Iljung S Kwak, and Serge J Belongie. Cost-effective hits for relative similarity comparisons. In *Second AAAI conference on human computation and crowdsourcing*, 2014.

[WKB14b] Michael J Wilber, Iljung S Kwak, and Serge J Belongie. Cost-effective hits for relative similarity comparisons. In *AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[WKKB15] Michael Wilber, Iljung S Kwak, David Kriegman, and Serge Belongie. Learning concept embeddings with combined human-machine expertise. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 981–989, 2015.

[Wol94] Jeremy M. Wolfe. Guided search 2.0 a revised model of visual search. *Psychonomic Bulletin & Review*, 1, June 1994.

[WP90] Ian Q Whishaw and Sergio M Pellis. The structure of skilled forelimb reaching in the rat: a proximally driven movement with a single distal rotatory component. *Behavioural brain research*, 41(1):49–59, 1990.

[WQD12] Xiang Wang, Buyue Qian, and Ian Davidson. On constrained spectral clustering and its applications. *Data Mining and Knowledge Discovery*, 28(1):1–30, September 2012.

[WTL11] Y. Wang, D. Tran, and Z. Liao. Learning hierarchical poselets for human parsing. *CVPR*, 2011.

[WWN+18] Zijun Wei, Boyu Wang, Minh Hoai Nguyen, Jianming Zhang, Zhe Lin, Xiaohui Shen, Radomír Mech, and Dimitris Samaras. Sequence-to-segment networks for segment detection. In *Advances in Neural Information Processing Systems*, 2018.

[WXW+16] Limin Wang, Yuanjun Xiong, Zhe Wang, Yu Qiao, Dahua Lin, Xiaoou Tang, and Luc Van Gool. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*, 2016.

[XDS17] Huijuan Xu, Abir Das, and Kate Saenko. R-c3d: Region convolutional 3d network for temporal activity detection. In *ICCV*, 2017.

[YJHL11] J. Yu, X. Jin, J. Han, and J. Luo. Collection-based sparse label propagation and its application on social group suggestion from photos. *ACM Trans. Intell. Syst. Technol.*, 2011.

[YKOB12] K. Yamaguchi, M. H. Kiapour, L. E. Ortiz, and T. L. Berg. Parsing clothing in fashion photographs. In *CVPR*, 2012.

[ZCZ+18] Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

[ZMT15] Liwen Zhang, Subhransu Maji, and Ryota Tomioka. Jointly Learning Multiple Perceptual Similarities. *arXiv:1503.01521 [cs, stat]*, March 2015. arXiv: 1503.01521.

[ZR12] X. Zhu and D. Ramanan. Face detection, pose estimation, and landmark localization in the wild. In *CVPR*, 2012.