Collaborative Learning of Concept Representations for Video Data

Francisco Torres, Hoda Eldardiry, Gaurang Gavai, and Chad Ramos
Palo Alto Research Center, 3333 Coyote Hill Road, Palo Alto, CA USA, torres@parc.com

Abstract

We present an approach for collaborative learning of representations for concepts embodied in unstructured datasets. Our approach learns both machine and expert interpretable representations, where ”expert” refers to expertise in the concept domain, as distinguished from a machine learning expert. In this paper, we focus on mining video data, but the approach is applicable to other types of data as well. Given a large video dataset and an expert-provided example clip that captures some notion of a desired concept, our proposed technology collaborates with the expert to discern the concept, even when it is not initially clear in the expert's mind. Since the expert may be exploring different possibilities, labeling a large set of data and then training a classifier to recognize the concept is not the right approach because it lacks the needed agility. For the learning to happen as the concept evolves, we use “learn-evolve-explain” cycles that generate (1) deep representations of the discerned concept, which can be used by a model for queries, and (2) visual representations that explains the discerned concept to the human expert. We summarize open source software developed to perform a collaborative video query and discuss our proposed road map for future work.

1 Introduction

Key to the impact of many emerging Artificial Intelligence technologies is the collaborative nature of the learning. Many systems leverage a variety of complementary players, and the better these players can collaborate, the more efficient the learning becomes. These collaborating players include machine learning programs, humans, and physical subsystems, each playing various roles. Machine learning methods learn models, inferentially process data streams, mine for interesting patterns, and sometimes generate explanations of results. Physical subsystems sense the environment and capture changing contexts. Human participants manage high level activities and mission development by creating demonstrations for teaching, providing feedback on the output, and strategically reorganizing the machine learning and physical subsystems when necessary. Ideally, human users can delegate much of the more mundane work to machine learning and physical subsystems, allowing the human expert to focus on high level context and goals.

Copyright held by the author(s). In A. Martin, K. Hinkelmann, A. Gerber, D. Lenat, F. van Harmelen, P. Clark (Eds.), Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019). Stanford University, Palo Alto, California, USA, March 25-27, 2019.

1.1 Learning challenges

This work focuses on two players in a collaborative learning approach: (1) the machine learning program, and (2) a human with expertise in the concept under investigation. Our learning methodology enables an agile and exploratory collaboration, while addressing the challenges listed below.

Typical machine learning constraints that can be relaxed by collaborative learning include:

- **Up-front problem specification.** Most learning approaches require the expert to concretely specify the problem they are trying to solve up front. This limits the ability to conduct exploratory learning. Users who are not machine learning practitioners would benefit from an approach where the user presents one to a few instances of some concept of interest and then engages in an intuitive collaboration.
- **Labeled data requirements.** Supervised machine learning typically relies on a lot of labeled data for training. This is expensive and poses a burden on the expert.
- **Unexplained model behaviors.** When an inference model generates an output without some explanation, an expert may not understand why this output was generated. Expecting the expert to fill in this understanding gap by observing model behavior for a large enough set of examples is not practicable at the scale and complexity of many modern AI systems. In the absence of such understanding, using the overall system outside of a narrow context becomes risky, yet broader use may be necessary to accomplish strategic goals or react to a changing context.

1.2 Expert-guided collaborative learning of concepts

Consider a domain expert studying a particular concept that arises throughout a large video dataset. This expert wants a technology that can locate video clips that capture the concept, since manual review of all the video is impractical. Our approach is to build a technology that uses “learn-evolve-explain” cycles; in this paper we report results for the ”learn” and ”evolve” steps and discuss our proposal on how to architect the ”explain” step. In our approach, the expert begins by providing an example video clip. This clip will show some notion of the desired scenario, but it will have activities and objects superfluous to the scope of the target concept as well. Furthermore, the expert may evolve her understanding of concept details and nuances as the algorithm builds representations for recognizing the intended concept and gets feedback from the expert. The learn-evolve-explain cycle works as follows:

- **LEARN: Learn feature representation of the concept depicted by the clip.**
  1) Use an ensemble of deep neural networks pre-trained on action recognition to extract deep feature representations of the example clip.
- **EVOLVE: Collaboratively evolve and clarify the concept.**
  2) Search for similar clips using ensemble scoring.
  3) Present proposed matching clips to the expert. Also present “near misses” so the expert sees actions that the algorithm considers to be outside the scope of the target concept.
  4) Get an expert’s feedback on the search results, and then learn a better machine representation of the concept of interest. By giving feedback, the expert also implicitly clarifies ambiguities and uncertainties, both for the algorithm and herself.
  5) Return to step 1 for another learning cycle, until the expert is satisfied.
- **EXPLAIN: Generate a human-interpretable representation.**
  6) After enough iterations of Learn + Evolve steps, a generative algorithm generates an ”evaluation” video clip that focuses on the target concept, suppressing and minimizing additional content.
  7) The expert either accepts this explanatory clip, or engages in further learn-evolve-explain cycles.

The learning algorithm and the expert jointly discern the target concept of interest starting with an example clip, without requiring a fixed concrete problem specification up front. In steps 2 through 5, the learning of the concept of interest evolves using an expert-guided ongoing collaborative approach. As part of step 5, a target bootstrapping algorithm takes its internal representations of all matching clips validated by the expert and forms a consensus representation, all in terms of machine feature representations. Finally, steps 6 and 7 aim to create a human-understandable representation that explains the internal model representation of the target concept.

1.3 Technology elements

The technology elements of this work can be summarized as follows:

- A video clip representation approach (section 3)
- A search algorithm (section 4)
- A method to refine the search given expert user feedback (section 5)
- A method to refine the machine representation of the target concept given expert user collaborative feedback (section 6)
- A method to generate a clip that captures only the concept, while de-emphasizing other details (future work on our proposed road map, section 9)

2 Related work

Our approach uses transfer learning, in that it uses embeddings from pre-trained deep neural nets in a context other than the original training objective. Using embeddings from a pre-trained deep neural net in a context other than the training objective often proves useful (Goodfellow et al. 2016).

Also, active deep learning is similar in some respects to the expert-guided collaborative learning we are describing in this paper; both aim to make best use of the expert’s time. For example, Gal et al (Gal, Islam, and Ghahramani 2017) take advantage of specialized models such as Bayesian neural networks to construct efficient deep active learning paradigms. Interesting strategies such as deep adversarial active learning (Ducoffe and Precioso 2018) also reduce the amount of expert input required. While active learning approaches focus on efficient use of expert input, they typically assume that the concept of interest has already been discerned and aim to achieve more efficient labeling. Our approach could be used as an active learning tool when the classes are well understood beforehand, but our primary focus in on collaborative learning of a concept that is not specified up front, starting with as little as one example from an expert. We do see potential value in integrating modern active learning algorithms like (Gal, Islam, and Ghahramani 2017) and (Ducoffe and Precioso 2018) with the collaborative learning discussed in this paper.

3 Video clip representation

To prepare data for use, we divide a video dataset into clips, and we compute a signature for each clip using an ensemble of deep learning neural networks. Currently, we define a clip to be 10 seconds long, although exploring this parameter and overlap of clips is on our road map. Below we discuss our ensemble model approach for feature representation. Our ensemble design captures two properties of the content in each video clip: appearance and motion. We pretrain our ensemble using a publicly available video dataset that contains a broad set of human actions.

3.1 Definitions

- Video clip: a short section of video, e.g., a 10 second clip.
- Video clip signature: a set of deep embedded feature vectors that encode clip characteristics.
- Feature vector: a vector of embedded features for a video clip computed using a neural network; in general a function of the clip and the neural network.
- Stream: a deep neural network model that uses video data processed in multiple ways, as part of a collection of streams in a multi-stream architecture.

3.2 Learning an ensemble model of deep neural networks

The ensemble we have been studying comprises three RGB and three warped optical flow deep nets adopted from the Temporal Segment Networks (TSN) work by Wang *et al* (2016). For each mode, the three networks were trained on the three published splits of UCF-101 data (Soomro, Zamir, and Shah 2012). The results reported here correspond to the six 1024-element $global\_pool$ embedded feature vectors in the six deep nets, which are the last feature vectors before the final layers that classify outputs into the 101 UCF classes. Once computed, we store video clip signatures as structured data in a database, enabling structured queries for comparing signatures.

The TSN approach takes a short video, divides it into a specified number of snippets, analyzes each snippet using both a spatial (RGB) convolution neural network and a temporal (optical flow) neural network, and then applies consensus functions to arrive at a final determination of the action. Optical flow shows the velocity of pixels, and warped optical flow attempts to suppress background motion, such as effects of camera, rather than actor, movement. Whereas Wang et al. (2016) report that an optical flow neural net performed nearly as well as a warped optical flow neural net in their work and takes less computation time to prepare, we have been using warped optical flow neural nets in order to increase robustness to camera motion. Our use cases have focused on cameras mounted on moving vehicles, unlike the UCF-101 dataset, and our initial tests suggested that warped optical flow could perform better than optical flow alone.

We chose TSN modeling because the UCF-101 dataset used in the TSN work has a mean clip length of 7.2 sec, comparable to the clip lengths of interest to us. Action concepts like "walking with a dog", "band marching", "riding a bike down a path", and "walking across the street" are the focus in our approach, as opposed to longer, more complicated activities, like a video of someone going through all the steps to bake a cake or build a piece of furniture. TSN emphasizes analyzing snippets of video for short actions, rather than creating a longer term memory of an evolving activity. Our methodology is meant to be used for tasks like identifying a test vehicle stopping for pedestrians crossing in front of it, and we would not expect it to work well in deducing whether a series of actions in a video corresponds to someone doing some shopping before picking up a child. The latter involves a series of actions and an abstract sense of intent, which is a different type of video machine learning task. The methodology proposed here could be a component of a larger machine learning technology that comprehends the latter activity, but it could not achieve that goal alone.

3.3 Choice of stream types

Any appropriate deep nets can be used, and different types of problems will do better with different deep net models. For example, adding an Image Net stream could help if concepts having more to do with images than video action are important, e.g. if someone is interested in finding all examples of crossing a street at a stop sign versus a stop light. The differences between a stop sign and a stop light are image differences, not action differences. In our user studies so far, we have found that users often want to include such "image" features in their search. Another example where a different neural net could be useful is the case of studying facial expressions and head motions. Neural nets trained on facial expressions rather than the actions in the UCF-101 dataset will probably do better, e.g., for videos capturing people's faces while driving cars.

3.4 Choice of deep network embeddings

Depending on the concept of interest, embeddings from different layers other than the final hidden layer in a deep net may be more useful to integrate into an ensemble model. For example, if one is looking for more basic motions, like veering left vs. right, lower layers may better distinguish such actions. In contrast, our currently UCF-101 trained TSN networks may not have that differentiation of left vs. right motion present by the time higher layers are reached, since these networks were trained to predict the correct action regardless of left vs. right motion.

In future work, we plan to investigate including more layers and let the human-algorithm collaboration discern how much to weigh lower vs higher layers in the ensemble.

4 Searching for similar clips

As described above, the search algorithm uses an ensemble of deep neural net embeddings. Similarities of clips are quantified by computing dot products of embedded feature vectors for the example clips and possible matches. The individual dot products are then combined into an ensemble score.

4.1 Choice of similarity measure for a single embedding

To compute the similarity of an embedded feature for a reference clip and the same embedded feature for a second clip i, we use

$$\frac{f_{DNN}^{(i)T} \cdot f_{DNN}^{ref}}{\left\| f_{DNN}^{ref} \right\|_2^2} \tag{1}$$

where f are the feature vectors, T denotes transpose, and DNN represents the deep neural net type (e.g., RGB or warped optical flow). Similarity corresponds to how close the result is to one. In experiments so far, we find this similarity measure works well in an ensemble model, whereas the results for single neural nets alone seem to suffer significantly higher variance.

Note that Eqn 1 differs from a cosine similarity. (The denominator is the square of the $L2$ norm of the embedded feature for the reference frame, not the product of the norms for both features.) We use this "dot product" similarity because it emphasizes what is similar along the hyper-dimensional direction of the reference embedding, not what is different in orthogonal hyper-dimensional directions. Note that two embedded feature vectors can be far apart in their hyper-dimensional space (e.g., according to a Euclidean or other distance metric) and still have a good similarity score.

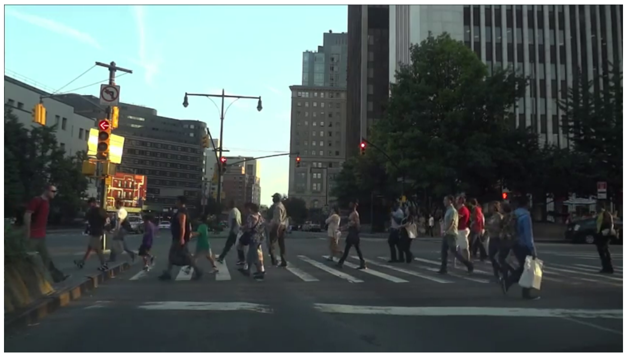


Figure 1: Experimental Evaluations of driving scenarios: Vehicle at intersection with pedestrians crossing. From the Downtown Brooklyn Drive video https://www.youtube.com/watch?reload=9&v=cjs3RxuKo6c at time 3:53.

4.2 Ensemble of similarities

Since the signature of a video clip is the set of features computed using multiple types of deep neural nets (DNN) trained on multiple splits of data, we need to specify how we are ensembling all of the similarity metrics.

Ensemble over data splits For each type of DNN and each candidate clip i, we compute the similarity

$$\varphi^{(i)}_{DNN} = \frac{1}{3}\sum_{j=1}^{3} \frac{f^{(i)T}_{DNN,j} \cdot f^{ref}_{DNN,j}}{\left\| f^{ref}_{DNN,j} \right\|_2^2} \tag{2}$$

where $f^{(i)}_{DNN,j}$ is the feature for split j of the given DNN type, computed for clip i, and $f^{ref}_{DNN,j}$ is the corresponding feature for the reference clip. When clip i is also the reference frame, $\varphi^{(i)}_{DNN} = 1$, and more generally, whenever the projection of $f^{(i)}_{DNN,j}$ on $f^{ref}_{DNN,j}$ equals the squared L^2 norm of $f^{ref}_{DNN,j}$, the summand equals one, even if $f^{(i)}_{DNN,j}$ and $f^{ref}_{DNN,j}$ are not equal. This is the desired behavior we discussed in section 3.1.

Ensemble over DNN streams The next step is to determine an overall similarity score. To do so, we use a Euclidean space in which each DNN corresponds to a dimension along which $\varphi^{(i)}_{DNN}$ is measured. In this space, a value of $\varphi^{(i)}_{DNN} = 1$ for each DNN dimension is the best possible similarity. We do not necessarily want to weigh each DNN dimension the same, but instead want to learn optimal weights. Thus, we compute the overall similarity of clip i to the reference as

$$\theta^i = \frac{\sum_{DNN} w^2_{DNN}\left(1 - \varphi^{(i)}_{DNN}\right)^2}{\sum_{DNN} w^2_{DNN}} \tag{3}$$

where w_{DNN} are the weights for each type of DNN.

5 Refining search

To decide which clips to present to the user for review, similarities θ^i are computed for the latest best guess of the values of w_{DNN} (w_{RGB} and $w_{\text{warped optical flow}}$ in the examples presented here). A small number of clips with similarities better than the current estimate of the threshold are selected, as well as a small number of clips with similarities close to but below the threshold. The user then gives feedback, and the algorithm computes new estimates of w_{DNN} and the threshold for θ^i. In our experiments we tended to see better outcomes when both matches and "near misses" were presented for review. This is perhaps similar to people establishing common understandings by agreeing both on positive and negative examples of concepts.

6 Refining machine representation: Target bootstrapping

In this section, we discuss refining the machine representation of the target concept given expert feedback. The goal is to replace $f^{ref}_{DNN,j}$ with a new bootstrapped set $f^{b}_{DNN,j}$ that is most consistent with all user validated matches, capturing what is similar with all of them and ignoring what is different. Referring back to equation 3, in mathematical terms we want to find $f^{b}_{DNN,j}$ such that $\theta^i \approx 1$ for all user-validated matching clips. Any set of $f^{b}_{DNN,j}$ for which

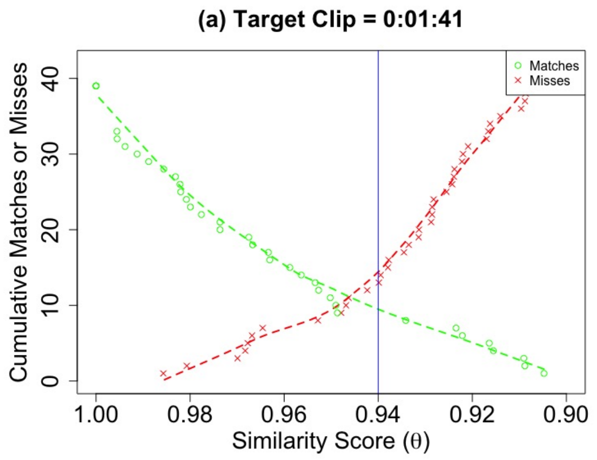
$$f^{(i)T}_{DNN,j} \cdot f^{b}_{DNN,j} = \left\| f^{b}_{DNN,j} \right\|_2^2 \tag{4}$$

for all clips i in the set of user validated matches, all splits j and all DNN types corresponds to a bootstrapped reference that has $\theta^i = 1$. Any such $f^{b}_{DNN,j}$ corresponds to a hyperplane perpendicular to it that contains all the endpoints of $f^{(i)}_{DNN,j}$ for all user-validated matches i.

There are an infinite number of such hyper-planes, since the dimensions of the $f_{DNN,j}$ are much higher (1024 in this work) than the number of matches a user will have validated. As a starting point for the bootstrapping, we compute the largest (in the L^2 norm sense) bootstrapped $f^{b}_{DNN,j}$ that satisfies equation 4 because it is the least restrictive in terms of reducing the impact of any one hyper-dimension on the similarity metric θ^i. In this way, we do not force more restrictions on the user's intent than is necessary for inferring a unifying machine representation. To reduce variance, we use bagging, choosing samples with replacement for each bag and averaging over three bags. Information about non-matches can also be incorporated. The mathematical details for the base version of target bootstrapping are provided in the appendix.

7 Software implementation

We have built Agile Video Query software implementing agile discovery for video datasets and made it available open source at https://github.com/PARC-projects/video-query-home, free for non-commercial use. The software components are (1) a Django API, (2) an Angular browser client, (3) back end python algorithms, and (4) a Postgres database. We are continuing to develop the software and welcome others to participate.



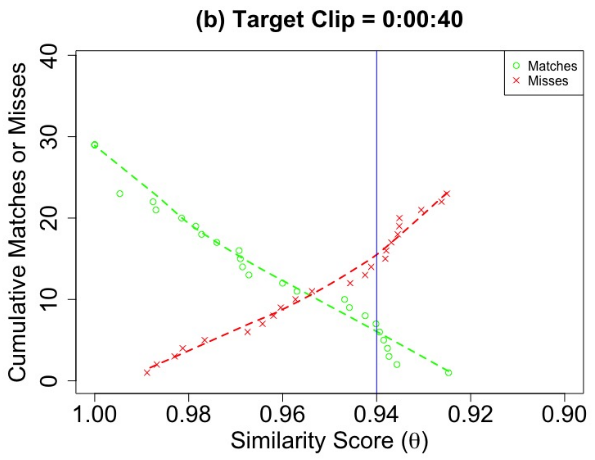


Figure 2: Cumulative Matches or Misses for Pedestrian Crossing Scenario. The vertical lines are learned thresholds separating predicted matches and misses. As θ decreases, the distribution of matches falls off while the one for misses grows.

8 Experimental evaluation

Using video from moving vehicles, we have begun performing user studies of the Agile Video Query software. Scenarios studied so far include: (1) vehicles interacting with pedestrians at intersections, (2) vehicles driving underneath an overpass, and (3) vehicles taking a noticeable left turn. See Figure 1 for a snapshot of the first scenario. (Copyright permission to show snapshots from the video for the second and third scenarios is pending.)

Not surprisingly, different users interact differently with the software. For some users, the software motivates focusing tightly on a particular concept, thereby helping to study a well-defined hypothesis in a disciplined manner. We have also observed cases where the software motivates a user to expand the scope of interest and perform curiosity-driven data exploration, in contrast with the former user type.

Figure 2 shows results for queries of vehicles interacting with pedestrians crossing a street, for two users of the first type discussed above. For Figure 2a, the user accepted any clip where the vehicle interacted with or was stopped for a pedestrian as a valid match. As illustrated, 80% of the matches are correctly located above the learned threshold θ, and 22.5% of the clips above the threshold are false positives (i.e., red x). For Figure 2b, the user only validated clips in which both the vehicle and the pedestrian were moving, rejecting clips in which the vehicle remained stopped. Although 79% of the matches are correctly located above the learned threshold, the false positive fraction is worse at 38%.

9 Technology road map

By analyzing for diverging sample distributions in the embedded feature space, future enhancements will help the human-machine collaboration discern when the user's interest is broadening or conflicted, and act by handling multiple concepts separately. We expect these enhancements to take the form of further development of target bootstrapping. Our road map also includes adding a larger set of deep neural net types, starting with an image-centric deep net to address non-action image recognition needs that have come up repeatedly in our user studies.

We are also working toward a generative algorithm that will use bootstrapped targets to produce video clips that humans can easily and intuitively interpret and evaluate. The goal of this capability is to provide a human-interpretable view of the machine representations. For our intended purposes, the generated video should highlight the actions of interest while also obscuring or subduing irrelevant features. Our current approach is to leverage both LIME (Ribeiro, Singh, and Guestrin 2016) and neural style transfer techniques (Gatys, Ecker, and Bethge 2016).

LIME computes which pixels are important for any one prediction made by a DNN image classifier, presenting a modified image with those pixels replaced with a highlight color. With a modified version of LIME, we expect to be able to show if the machine representation is locking into something unexpected, such as irrelevant trees in the background. If a user sees this happening, then the user can provide further examples without those features in the next round of feedback to the algorithm. Further along the roadmap, we will research adding an explicit ability for users to directly remove irrelevant things revealed by visualizations of the machine representation.

Highlighting important pixels is only a partial solution, however, because it will not reveal what relations among pixels are important versus unimportant. Consider, for example, video clips of pedestrians walking in front of a car at an intersection. Highlighting the pedestrians reveals some information, but it does not tell the user whether the machine representation prioritizes, say, the pattern on a pedestrian's shirt or the reflective stripes on the safety vest of a jogger. We are investigating using neural style imaging as a way to alter a clip in order to subdues patterns and details that are unimportant for the similarity score. Our hypothesis is this approach can provide a richer human-interpretable version of the machine representation; for example, the modified clip may hypothetically show dull, monotone clothing for queries that are not focused on clothing, or conversely, show the details of a jogger's safety vest if the user is intending the query to focus on joggers with such vests.

10 Acknowledgement

We thank the U.S. Department of Transportation Federal Highway Administration and Dr. Ana Maria Eigen for support and funding under EAR grant ID DTFH6115H00006.

References

Ducoffe, M., and Precioso, F. 2018. Adversarial active learning for deep networks: a margin based approach. *CoRR* abs/1802.09841.

Gal, Y.; Islam, R.; and Ghahramani, Z. 2017. Deep bayesian active learning with image data. *CoRR* abs/1703.02910.

Gatys, L. A.; Ecker, A. S.; and Bethge, M. 2016. Image style transfer using convolutional neural networks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, 1135–1144.

Soomro, K.; Zamir, A. R.; and Shah, M. 2012. Ucf101: A dataset of 101 human actions classes from videos in the wild. *arXiv preprint arXiv:1212.0402*.

Wang, L.; Xiong, Y.; Wang, Z.; Qiao, Y.; Lin, D.; Tang, X.; and Val Gool, L. 2016. Temporal segment networks: Towards good practices for deep action recognition. In *ECCV*.

11 Appendix: Target Bootstrapping

In target boostrapping, we want to choose the least restrictive bootstrapped $f^{ref}_{DNN,j}$ that satisfies equation 4, in order to not force more restrictions on the user's intent than is justified. Accordingly, we choose the target bootstrapping to be

$$\max \ \frac{1}{2} f^{bT} \cdot f^b \tag{5}$$

$$\text{such that} \ \ f^{(i)T} \cdot f^b = \left\| f^b \right\|_2^2 \tag{6}$$

for all clips i that the user has validated to be a match. When using bagging, the set of all clips is replaced by a set randomly chosen from the entire set with replacement.

As written, this maximization problem is in a form that is difficult to handle. To put it in a nicer form, we introduce the scaled target

$$t = \frac{f^b}{\left\| f^b \right\|_2^2} \tag{7}$$

In terms of this scaled target, equations 5 and 6 become, for each choice of DNN and j

$$\min \ \frac{1}{2} t^T \cdot t \tag{8}$$

$$\text{such that} \ \ f^{(i)T} \cdot t = 1 \tag{9}$$

This is a straightforward quadratic minimization problem with linear equality constraints.

Using the method of Lagrange multipliers, the Lagrangian for equations 8 and 9 is

$$L = \frac{1}{2} t^T \cdot t + \lambda^T \left(F \cdot t - 1 \right) \tag{10}$$

where λ is a vector of Lagrange multipliers, and

$$F = \begin{bmatrix} \leftarrow f^{(1)^T} \rightarrow \\ \leftarrow f^{(2)T} \rightarrow \\ \vdots \\ \leftarrow f^{(J)^T} \rightarrow \end{bmatrix} \tag{11}$$

1_J is a vertical vector of m ones, and J is the number of user-validated matches corresponding to the $f^{(i)}$ in equation 9.

Setting $\frac{\partial L}{\partial t_i} = 0$ to find the minimum yields

$$t + F^T \cdot \lambda = 0 \tag{12}$$

Since $F \cdot t = 1_J$ (equation 9 and $\frac{\partial L}{\partial \lambda} = 0$), it follows that

$$1_J + F F^T \cdot \lambda = 0, \tag{13}$$

implying

$$\lambda = -\left(F F^T \right)^{-1} \cdot 1_J \tag{14}$$

Substituting back into equation 12, we derive the solution for t:

$$t = F^T \cdot \left(F F^T \right)^{-1} \cdot 1_J \tag{15}$$

Since t, the scaled form of f^b given by equation 7, is the quantity needed for the bootstrapped version of equation 1, there is no need to convert t back to f^b.