CSE 847 Intermediate Report

Discovering the underlying topological space of extracted features

Hamid Karimi
karimiha@msu.edu

Harrison LeFrois
lefroish@math.msu.edu

ACM Reference format:
Hamid Karimi and Harrison LeFrois. 2016. CSE 847 Intermediate Report. In *Proceedings of The Best Conference, East Lansing, MI, 04/28/2017,* 6 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 PROBLEM DESCRIPTION

Manifold learning is a common approach to dimensionality reduction. The idea is to find an underlying space (a manifold) that actually describes the data in a lower dimension than the initial data representation. For our project, we have decided to examine what is happening in the hidden layers of neural networks. Apart from the simplest examples, the maps that a neural network is learning become complicated very quickly. We are particularly interested to see if it is possible to find an underlying topological space (potentially a manifold) that parametrizes our feature space for a layer in our network. In other words, what is the space that spans our selected features? Is it possible to relate the representation of our data at each layer to something geometric? Data analysis generally does not leverage the geometry and topology of the data [1], which is what we are aiming towards. It may not be possible to find an underlying manifold for our data, but parametrizing the feature space by a topological space can help us better understand the particular data set and what is happening in our network.

2 SURVEY AND RELATED WORK

Dimensionality reduction is a well-established and often used strategy to analyze high dimensional data and enables the user to visualize data that would otherwise be inaccessible. This is done by extracting features that best represent the data and mapping the data to a lower dimensional space based on these features, hopefully without too much loss of information.

Some common methods of dimensionality reduction and manifold learning are Principal Component Analysis (PCA), Linear Discriminant Analysis (LDA), Multidimensional Scaling (MDS), and ISOMAP. PCA, LDA, and MDS are linear approaches that are not very useful for non-linear structures. While ISOMAP is a non-linear approach, it fails to retrieve the underlying embedding in complicated structures. Nevertheless, deep neural networks have shown to be quite successful in extracting the underlying manifold of data feature space.

One of the common networks for manifold learning is autoencoders. Autoencoders essentially seek to learn the identity map, i.e. they have the same number of input nodes as output nodes. However, by varying the number of nodes in the hidden layers (say less than the input layer) or imposing sparsity, one can extract a number of features which is less than the dimension of the data's ambient space. If the data was sampled from a manifold embedded in the ambient space, the hope is that the extracted features can tell us about the manifold and/or potentially allow us to reconstruct the manifold. This is the general goal of manifold learning. But what if the data is not sampled from a manifold? Then maybe it is possible to examine the extracted features and deduce an underlying space for the features. We intend to utilize sparse autoencoders during our investigation. Sparse coding has seen great success in feature learning [2], and our hope is that utilizing the geometry/topology of the feature space may yield new insights.

This project is inspired by the work of Dr. José Perea in [3]. A motivating example in his work involved finding the underlying topological space of 7×7 pixel grey-scale images displaying a fixed-width line. Let $X \subset \mathbb{X}$ denote a sample of these images from the set $\mathbb{X}$ of all such images. Each image could be represented as a 7×7 matrix whose entries, consisting of -1 for black and 1 for white, sum to 0. By concatenating the columns of each matrix, we could view this data set X as a subset of $\mathbb{R}^{49}$. He noticed that despite being embedded in a 49 dimensional space, each image was governed by the distance of the line from the center and the angle of the line compared to the horizontal. Locally, $\mathbb{X}$ appears to be 2 dimensional. Using a combination of MDS and a strategy to discover the orientability of $\mathbb{X}$, Perea was able to conclude that X was actually sampled from the real projective plane $\mathbb{R}\mathbf{P}^2$. Each image in X was then able to be assigned a pair of coordinates in $\mathbb{R}\mathbf{P}^2$ by using an extension of PCA called Principal Projective Components. In this particular instance, the data was sampled from a manifold and the important features of each image, namely the angle and distance, provided some of the information needed to find the underlying topological space. The goal of this work, possibly in conjunction with algebro-topological tools, is to utilize machine learning to make similar discoveries.

3 PLAN AND MILESTONES

With an eye towards Dr. Perea's work, we would like to see if autoencoders can provide a way towards parameterizing the feature space of other data sets. The data set he used provides an opportunity to test and see if an autoencoder can detect features that would hint at $\mathbb{R}\mathbf{P}^2$ being the underlying topological space. We will utilize different sparse autoencoder architectures to extract features from our data sets to see if each autoencoder focuses on different features (and hence a potentially different space spanning the features). Using data from [3], MNIST [4], and potentially natural images, we will apply our algorithms to each data set and attempt to find an underlying topological space that would parametrize the feature space. Our projected timeline and milestones are in the following chart.

The Best Conference, East Lansing, MI
2016. 978-x-xxxx-xxxx-x/YY/MM…$15.00
DOI: 10.1145/nnnnnnn.nnnnnnn

Week 1-2 (02/17-03/03)	Literature survey and complete sparse autoencoder tutorial [5]
Week 3-4 (03/03-03/17)	Construct autoencoders and test on data from [3]. Prepare Intermediate Report
Week 5-6 (03/17-03/31)	Run autoencoders on data from MNIST and natural images. Analyze feature space.
Week 7-11 (03/31-04/28)	Continue feature space analysis. Prepare final report.

4 TOPOLOGY BACKGROUND

In topology, we assign invariants (such as numbers or algebraic objects like groups) to a space as a way to help distinguish different spaces from each other. By construction, these invariants do not change when we make small deformations to our original space, for example denting a sphere. This makes topology particularly well-suited to applications in data analysis, where robustness to noise is desirable. However, there is not much that is topologically interesting or illuminating about a collection of points, the invariants we use are trivial (homology groups, homotopy groups, etc). It is only when we construct a more complex topological space that we are then able to use the tools from topology effectively to deduce properties of our data set such as shape.

The topological tool we use to study our data sets in this project is called persistent homology [6]. Homology tells us about the number of "holes" in our space. For example, if X is a topological space, the zeroth homology group $H_0(X)$ gives the number of connected components, $H_1(X)$ gives the number of one-dimensional holes (like the hole in a donut), and $H_2(X)$ gives the number of holes, as in a cavity (like the interior of a beach ball). What persistent homology computes is how the homology groups of a space (in this case a simplicial complex constructed from our data) change as we vary a particular parameter. In order to compute the persistent homology of a data set, we need to first give the data a topological structure since the homology of a set of points tells us nothing. We do this by constructing what is called the Rips complex, which is done in the following way. Let D be our data set, $\epsilon > 0$, and $R_\epsilon(D)$ denote our Rips complex. Then we call $\{x_0, x_1, ..., x_k\} \subset D$ a set of vertices and say that this set spans a k-simplex $\sigma \in R_\epsilon(D)$ if the distance $d(x_i, x_j)$ is less than or equal to ϵ for every i and j. If $\epsilon < \epsilon'$, then we get an inclusion of the complexes $R_\epsilon(D) \to R_{\epsilon'}(D)$.

Now that we have constructed a topological space from our data set, we can begin to discuss the homology of this space. The definition and details of simplicial homology (simplicial since our topological space $R_\epsilon(D)$ is a simplicial complex) are contained in [7]. From the inclusion map mentioned above, we get an induced map between the homology groups $H_i(R_\epsilon(D)) \to H_i(R_{\epsilon'}(D))$, and this leads us to the persistent homology. As a technical side note, all of our homology groups have coefficients in a field, often $\mathbb{Z}/2\mathbb{Z}$ or $\mathbb{Z}/3\mathbb{Z}$. When viewed as a module over the coefficient ring, we see that our homology groups are actually now vector spaces with linear maps between them.

The persistent homology of our collection of complexes $\{R_\epsilon(D)\}_{\epsilon \geq 0}$ measures when a homological class (a connected piece in zeroth dimension, a loop in the first dimension, etc) is born and when it dies. The birth time is the ϵ for which the class first appears, and the death time is defined analogously. As an example, suppose that $D = \{(0,0),(0,1)\} \subset \mathbb{R}^2$. For each $\epsilon > 0$, we construct $R_\epsilon(D)$. Notice that $R_\epsilon(D)$ only changes when $\epsilon = 1$. For $\epsilon < 1$, we see that $R_\epsilon(D)$ has two vertices and no other simplices. When $\epsilon = 1$, we see that $d((0,0),(0,1)) = 1$, so $\{(0,0),(0,1)\}$ spans a 1-simplex, i.e. we draw an edge between those two points. Now there is only one connected component instead of two. So one of the zeroth homology classes no longer exists (it died!). The birth and death of homology classes can be represented by intervals, and these can be drawn as barcodes. Generally, the longer intervals in the barcode represent the topological features which persist longest, and are hence thought to represent the salient topological features and shape of our data. All of our homology computations are done using the Javaplex package for Matlab [8].

5 MOTIVATION

We have decided to slightly change the direction of our project from that originally proposed at the beginning of this report. Large amounts of data are needed to train a neural network, so we need a large data set before doing anything else. We have decided to work with the CASIA-Webface data set. This data set contains close to 500,000 images of faces, which is larger than any idea for data we have had thus far. We will examine the feature space associated to this data set from a Convolutional Neural Network (CNN), the motivation for which is explained in this section.

The motivation for our idea of investigating the feature space of images of faces came from the persistent homology of images of a rubber duck.

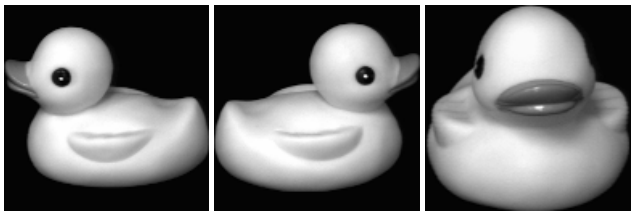


Figure 1: Rotations of a rubber duck

The Columbia Object Image Library (COIL) contains a collection, COIL-20, of gray-scale images of 20 everyday objects [9]. There are 72 images of each object, where each image differs from the next by rotating the object five degrees. Images can be analyzed by viewing them as matrices whose entries are pixel values (integers between 0 and 255). To analyze the images of the rubber duck, each 128×128 image is represented as a 128×128 matrix. We then take the first column of this matrix and concatenate each subsequent column to create a column vector in $\mathbb{R}^{16384}$. By repeating this process with each image, we get 72 vectors (or data points) in $\mathbb{R}^{16384}$ associated to each object. This point cloud can then be used to construct the Rips complex and compute the persistent homology.

From the barcode shown, particularly the first homology, we see that there is one prominent 1-cycle or loop in our data. Despite living in a very high dimensional space, our data points roughly form a circle, as far as the topological perspective is concerned.

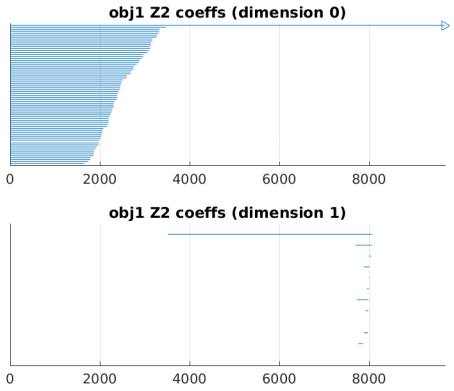


Figure 2: The barcode associated to the rubber duck images. The top is the zeroth homology and bottom is the first homology.

Note that in topology, a square is viewed as identical to a circle because one can continuously deform the square to make it look like a circle (and vice versa). Continuous deformations can be thought of as wiggling and moving pieces of your space without tearing or cutting. Periodic behavior in data, i.e. starting at one point, changing locations in space then returning to the starting point, can be thought of as a loop. These loops then show up as a 1-cycle in the barcode associated to the data. When viewed from this perspective it is not surprising that we have a 1-cycle appearing in our data because we are starting at one angle of rotation, rotating the duck 360°, hence returning to our starting point.

Figure 3: Examples from the CASIA-Webface dataset

Our original goal was to look at the feature space from a hidden layer of a simple autoencoder to see if we could find an underlying topological space that would parametrize our feature space. In this instance, our data roughly forms a circle in high dimensional space, so one would hope that the feature space would be parametrized by a circle as well. However, for each object we only have 72 data points. This is not remotely enough data to train a network. It was then brought to our attention that there is a large database of faces called CASIA-Webface [10]. Since many of these photos show different angles of the same person as seen in Figure 3 (i.e. rotations of the person!), analyzing the feature space of this data space for different network architectures may provide some insights about the data set as a whole. We hope that these techniques and subsequent analysis will shed more light on how neural networks work.

6 DATASETS

Face datasets used to be collected in controlled environments wherein images had high quality and have least distortion. Facial recognition algorithms on these datasets have already achieved excellent results. However, images in the real-world situations are not that perfect. In order to have better performance in more practical scenarios, facial recognition has moved toward working on images genered in uncontrolled manner. One of the popular unconstrained facial recognition benchmarks called LFW (Labeled Faces in the Wild) [11] emerged in 2007. It consists of 5749 subjects crawled from Internet. We will use this dataset for **evaluation set**.

As a rule of thumb, the more training samples we have at our disposal, the better performance we can achieve. In addition, the more diversity and variability exist in the samples, the more we can generalize the trained model. CASIA-Webface [10] enjoys both of these properties. It consists of 494,414 images of 10,575 subjects. The images are crawled out of profile of celebrities from the IMDB website [1]. This dataset is the largest publicly available face dataset, and the second largest dataset after Facebook's private face dataset, SFC [12]. CASIA-Webface is our **training set** for this project.

7 PREPROCESSING

The first thing we need to do is aligning the images. We aligned the images utilizing the recently published method called Multitask Cascaded Convolutional Networks (MTCCN) [13]. We cropped the training images to 182×182 and the evaluation images to 160×160. Since there were some Python package and Tensorflow issues with HPCC, we ran the alignment program on a laptop which took two days long.

8 CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Networks (CNN) have shown to be successful in facial recognition and verification. Thanks to deep networks and larger datasets, we can achieve incredibly high accuracy in facial recognition [10] [14]. Therefore, we use deep CNN to train the model. Our base implementation for this project is from [14], where the authors have implemented a basic training model, data augmentation, as well as several deep CNN architectures. We will implement the CNN architecture in Fig 4 as well. This simple model has been shown to achieve high accuracy on the CASIA-Webface dataset [10].

9 PROPOSED LOSS FUNCTION

In addition to examining the embeddings of these images into the feature space and looking for an underlying topological space, we also wish to expand upon the success of this CNN by proposing a new loss function.

[1] http://www.imdb.com

Name	Type	Filter Size /Stride	Output size	Depth	#Params
Conv11	convolution	3×3 / 1	100×100×32	1	0.28K
Conv12	convolution	3×3 / 1	100×100×64	1	18K
Pool1	max pooling	2×2 / 2	50×50×64	0	
Conv21	convolution	3×3 / 1	50×50×64	1	36K
Conv22	convolution	3×3 / 1	50×50×128	1	72K
Pool2	max pooling	2×2 / 2	25×25×128		
Conv31	convolution	3×3 / 1	25×25×96	1	108K
Conv32	convolution	3×3 / 1	25×25×192	1	162K
Pool3	max pooling	2×2 / 2	13×13×192	0	
Conv41	convolution	3×3 / 1	13×13×128	1	216K
Conv42	convolution	3×3 / 1	13×13×256	1	288K
Pool4	max pooling	2×2 / 2	7×7×256	0	
Conv51	convolution	3×3 / 1	7×7×160	1	360K
Conv52	convolution	3×3 / 1	7×7×320	1	450K
Pool5	avg pooling	7×7 / 1	1×1×320		
Dropout	dropout (40%)		1×1×320	0	
Fc6	fully connection		10575	1	3305K
Cost1	softmax		10575	0	
Cost2	contrastive		1	0	
Total				11	5015K

Figure 4: CASIA CNN architecture [10]

9.1 Related works

What we are basically interested in this project is a function f which embeds our data X in d-dimensional Euclidean space. As far as clustering performance is concerned, we are interested in embedding in which samples from the same class are closer to other while samples from different classes are far apart. To this end, some authors have defined loss function of their deep network to achieve aforementioned property (i.e., less inter-cluster similarity and less intra-cluster variability). In other words, $d(f(x_1), f(x_2))$ should be less than $d(f(x_1), f(x_3))$ where x_1 and x_2 belong to the same class while x_3 class is different and d is a distance function such Euclidean or cosine. Authors in [14] have proposed triple loss :

$$\sum_{i=1}^{n}[\| f(x_i^a) - f(x_i^p) \|_2^2 - \| f(x_i^a) - f(x_i^n) \|_2^2 + \alpha]_+ \qquad (1)$$

x_i^a is an image of a person (anchor image), x_i^p is another image of the same person (positive), x_i^p is an image of different person (negative image) and α is a margin that is enforced between positive and negative pairs.

In [15] the authors have proposed the following loss function.

$$\min_P \lambda \frac{\sum_M d(a_i, b_j)}{|M|} + (1-\lambda)\frac{\sum_V [1 + d(a_i, b_j) - d(a_i, c_k)]_+}{|V|} \qquad (2)$$

Where P is the learned mapping, y is class of a sample, $M = \{(i,j)|y_{a_i} = y_{b_j}\}$, and $V = \{(i,j,k)|y_{a_i} = y_{b_j}, y_{a_i} \neq y_{c_k}\}$. As noted in [15] the first term in Eq. 2 is a pull term that is minimized when points that belong to the same cluster are close, while the second term is a push term that is minimized when the points that belong to different classes are farther than points that belong to the same class. These two are controlled by the parameter $0 \leqslant \lambda \leqslant 1$.

Authors in [16] have proposed a center loss as follows:

$$\sum_{i=1}^{N} \| x_i - c_{y_i} \|_2^2 \qquad (3)$$

where x is a sample and c_{y_i} is the centroid of y_i th class.

9.2 Motivation for new loss function

The objective of each clustering algorithm is to make elements with the same label together and different label apart. Therefore, the target embedding of data should also follow the similar criteria i.e. same label data should be closer than different label data. Previous loss functions somewhat follows these criteria but not fully. Triplet loss function in Eq 1 doesn't consider centroids of clusters while they could be helpful in disjointing the clusters. Loss function proposed in 3 has employed centroid of a cluster but not its relation to other cluster centroids or samples. In Eq 2 distances among the centroids of different clusters are handled implicitly by pushing dissimilar instances away. This pushing is arbitrary and has no direction of convergence. If we incorporate distance from the centroid (of own cluster) to be one of the objectives, then it tries to push dissimilar objects to the centroid of its own cluster. Moreover, we also want distances between cluster centroids to have margin more than 1. Fig 5 shows the graphical representation of different components of proposed loss function.

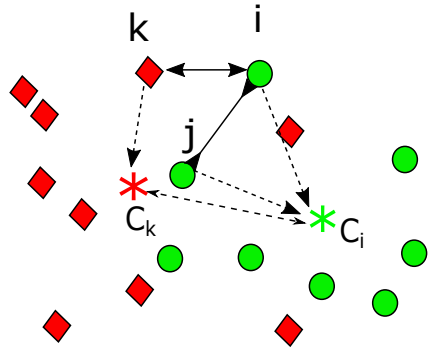
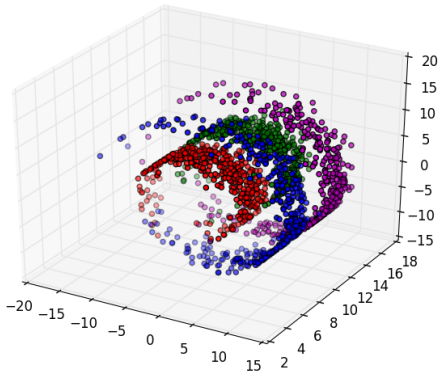


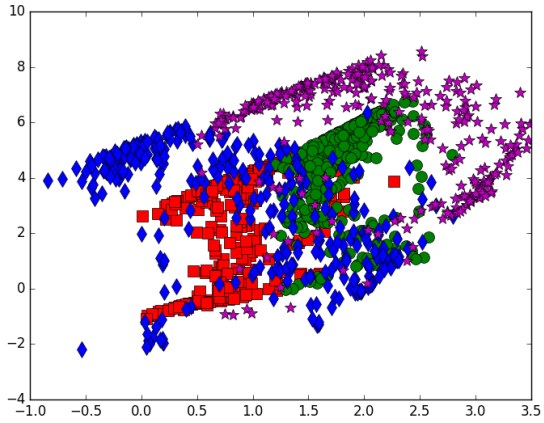
Figure 5: Graphical representation of loss function in the embedded space

Effectiveness of new loss function have been shown for a toy example in below. Data points are following a Swiss-roll distribution shown in Fig 6a which contains 4 different classes. Fig 6b shows initial embedding. Fig 6c shows embedding achieved by applying Eq 2 loss function after training for 150 epochs . Fig 6d shows the embedding of proposed loss function after 150 epochs.

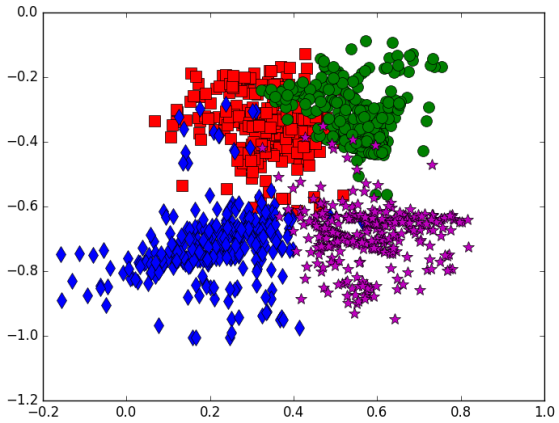
The new loss function will be implemented in conjunction with CNN of Fig 4. We hope and anticipate to achieve a better embedding which results in more meaningful topology representation.



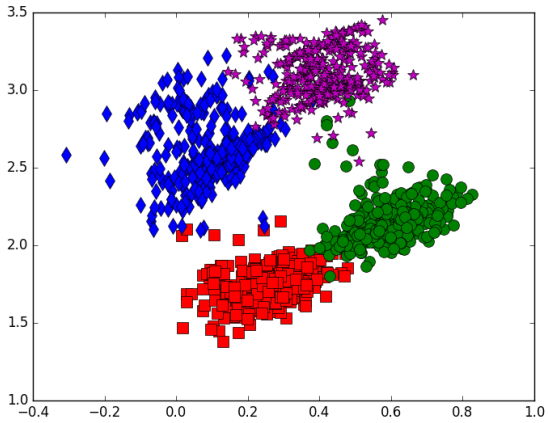
(a) Data shape in 3D (1600 instances)



(b) Initial embedding(same for both)



(c) Loss proposed in [15]



(d) New loss function

Figure 6: Performance of two embedding loss functions in Swiss-roll data

REFERENCES

[1] Woong Bae, Jae Jun Yoo, and Jong Chul Ye. Beyond deep residual learning for image restoration: Persistent homology-guided manifold simplification. *CoRR*, abs/1611.06345, 2016. URL http://arxiv.org/abs/1611.06345.

[2] Yunlong He, Koray Kavukcuoglu, Yun Wang, Arthur Szlam, and Yanjun Qi. Unsupervised feature learning by deep sparse coding. In *Proceedings of the 2014 SIAM International Conference on Data Mining*, pages 902–910. SIAM, 2014.

[3] J. A. Perea. Multi-Scale Projective Coordinates via Persistent Cohomology of Sparse Filtrations. *ArXiv e-prints*, December 2016. URL http://adsabs.harvard.edu/abs/2016arXiv161202861P.

[4] Yann Lecun and Corinna Cortes. The MNIST database of handwritten digits. URL http://yann.lecun.com/exdb/mnist/.

[5] Andrew Ng, Jiquan Ngiam, Chuan Yu Foo, Yifan Mai, and Caroline Suen. UFLDL Tutorial, 2013. URL http://ufldl.stanford.edu/wiki/index.php/UFLDL_Tutorial.

[6] Afra Zomorodian and Gunnar Carlsson. Computing persistent homology. *Discrete Comput. Geom.*, 33(2):249–274, February 2005. ISSN 0179-5376. doi: 10.1007/s00454-004-1146-y. URL http://dx.doi.org/10.1007/s00454-004-1146-y.

[7] James R. Munkres. *Elements of Algebraic Topology*. Addison-Wesley, 1993.

[8] Andrew Tausz, Mikael Vejdemo-Johansson, and Henry Adams. JavaPlex: A research software package for persistent (co)homology. In Han Hong and Chee Yap, editors, *Proceedings of ICMS 2014*, Lecture Notes in Computer Science 8592, pages 129–136, 2014. Software available at http://appliedtopology.github.io/javaplex/.

[9] S.A. Nene, S.K. Nayar, and H. Murase. Columbia Object Image Library (COIL-20), 1996. URL http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.

[10] Dong Yi, Zhen Lei, Shengcai Liao, and Stan Z. Li. Learning face representation from scratch. *CoRR*, abs/1411.7923, 2014. URL http://arxiv.org/abs/1411.7923.

[11] Gary B Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical report.

[12] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708, 2014.

[13] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23 (10):1499–1503, Oct 2016. ISSN 1070-9908. doi: 10.1109/LSP.2016.2603342.

[14] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015.

[15] S. Siena, V. N. Boddeti, and B. V. K. V. Kumar. Maximum-margin coupled mappings for cross-domain matching. In *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 1–8, Sept 2013. doi: 10.1109/BTAS.2013.6712686.

[16] Yandong Wen, Kaipeng Zhang, Zhifeng Li, and Yu Qiao. A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer, 2016.