Deep Metrics for Visual Retrieval Tasks

Zhao Yuan
School of Computer Science, UCD
zhao.yuan@ucdconnect.ie

Ning Tao
School of Computer Science, UCD
ning.tao@ucdconnect.ie

Yili Lai
School of Computer Science, UCD
yili.lai@ucdconnect.ie

Hanyuxi Zhou
School of Computer Science, UCD
hanyuxi.zhou@ucdconnect.ie

Yikai Wang
School of Computer Science, UCD
yikai.wang@ucdconnect.ie

Yuping Tian
School of Computer Science, UCD
yuping.tian@ucdconnect.ie

***Abstract*—Metric learning endeavors to map data to an embedding space where similar data are close to each other, and dissimilar data are far from each other. This article aims to reveal and compare different metric learning algorithms' performance by implementing an image retrieval web application that has a standard pipeline for multiple algorithms. There are two general pipelines in this project, one for training and the other for image retrieval. The web application of this project is built on a Flask web framework and Apache2 web server. As a result of the project, two algorithms, multi-similarity, and soft-triple appeared to over-perform under the same experimental setting.**

***Index Terms*—Deep Metric Learning, Deep Learning, benchmark**

I. Introduction

Deep metric learning(DML), a way that applies machine learning to learn a distance function that measures similarity among samples, has attracted a lot of interest in the last 10 years. Although its origins can be traced back to many earlier work, metric learning is believed that really emerged in 2002 with the pioneering work of Xing *et al*[1][2]. It has wide applications in many fields, for example, face recognition, image retrieval and etc. Deep metric learning has indeed brought progress in these areas. Google's FaceNet model uses Triplet-Loss to refresh the record of face recognition at that time[3]. In the past two years, the loss functions that have been successfully applied in face recognition such as Angular-Softmax and Additive Margin Softmax improved by Softmax have introduced the idea of metric learning. As improvement in deep learning, deep metric learning areas proposed many methods and metrics that claim their progress. Although there is no comparison between them under similar training and hyper parameter optimization scenarios.

In our project, we propose to conduct a study of the DML area to evaluate some algorithms. After some research in this field, we reproduced 7 existing deep metric learning methods: Contrastive loss, Triplet loss, Cosface, ArcFace, Softtriple, Multi-similarity and proxy-NCA, and tried to benchmark them in a relative fair setting: training and testing in the same dataset(Caltech-UCSD Birds-200-2011[4] and Cars-196[5]) and evaluating in the same metrics. We applied two evaluation metrics: recall@k and precision@k in these algorithms for comparison. Except for this work, we built a website for users so that our reproduced models are accessible. We also introduced an evaluation method based on users' ratings.

This report is organized as follows. We first discuss the related works in this area with detailed introduction of the algorithms we implemented. In section 3, we demonstrate the user stories of this project. After that, we cover the implementation of models and websites. Section 5 deals with the evaluation of these reproduced algorithms. Lastly, we conclude our work in section 6, and discuss the limitations and the future work of this project.

II. Related Work

According to research papers published by researchers from Facebook AI and Cornell Tech on the preprint platform arXiv, they claim that the research progress in the field of deep metric learning in the past ten years is actually non-existent[6]. At the same time, it shows no substantial improvement in the comparison between more than a dozen algorithms that have emerged in recent years and baseline methods (Contrastive and Triplet) thirteen years ago. The paper first proposed the concept of Unfair Comparisons, which is essentially a mathematical control variable idea. Like a car race, different competitors should have the same distance in the race to compare the speed. Therefore, Unfair Comparison emphasizes that to compare algorithms' performance, as many parameters should be uniform as possible. Among these parameters, network architecture and evaluation measures occupy the highest position. Based on this idea, we selected 7 algorithms that are representative in time (primitive and emerging) and have certain commonalities in the network structure, then research their respective performance through training and evaluation.

A. Why Choose these 7 Algorithms

- Time Span: From Contrastive and Triplet to the latest algorithms (the latest is in 2019), it covers an extensive time range, which is more representative of the comparison between the primitive algorithms and the emerging algorithms.
- Network Structure Commonalities: Proxy-NCA, Multi-Similarity, and Softtriple all use the BNInception network. Both Cosface and Arcface use Resnet 50. Contrastive and Triplet are the most representative pair-

training and triplet-training origins. Common network structures will save configuration time and have the opportunity to make horizontal comparisons.
- These 7 algorithms have corresponding papers detailing their processes, which saves us much time to re-implement these algorithms, and more research can be implemented in a limited time. The challenges we face include replacing a unified data set and ensuring the unity of more parameters to the greatest extent.

B. Introduction to Each Algorithm

- Contrastive[7]: The Contrastive loss model is a method called Dimensionality Reduction by learning an invariant mapping using Contrastive loss, for which learning a function that maps the data to output manifold. The technique overcomes two main drawbacks of previous dimension reduction methods. One is that it does not need the relation between the new data and input space; the other is that it does not require any specific distance metric. It can learn an invariant mapping to a low dimensional manifold using prior knowledge, and the complexity of the invariances that can be learned are only limited by the power of the parameterized function (structure of the neural network). The function maps inputs that evenly cover a manifold, as can be seen by the experimental results. It also faithfully maps new, unseen samples to meaningful locations on the manifold.
- Triplet[3]: The Triplet loss model used triplets of images to transform the image's original dimension into a new embedding dimension based on the triplet-based loss function. Triplet loss is a loss function where a baseline (anchor) input is compared to a positive input and a negative input.
- Proxy-NCA[8]: Proxy-NCA is a model based on the triplet-based method. However, Proxy-NCA is finding proxies to represent the whole datasets instead of finding informative triplets. The target of the triplet loss is to make similar points closer to the anchor point than different points. Therefore, the sampling is essential for the performance of the loss function, Proxy-NCA is to redefine Triplet based losses over a different space of points, which is called proxies. Each point in the original space has a proxy point that is close to it, so this is a space that is small enough and similar to the training set.
- Softtriple[9]: SoftTriple loss learns the embeddings directly from deep neural networks and without triplet constraints sampling. Qi Qian et al. found SoftMax loss is equivalent to a smoothed triplet loss where each class has a single center and proposed SoftTriple loss, which extends the SoftMax loss with multiple centers for each class. So SoftTriple loss is demonstrated on the fine-grained visual categorization tasks which optimize the geometry of local clusters[10], provides more flexibility for modeling intra-class variance like triplet constraints, but reduces the total number of triplets to alleviate the challenge from a large number of triplets like using proxies. Moreover, optimizing SoftTriple loss can learn the embedding without the sampling phase by mildly increasing the size of the last fully connected layer, since most samples cannot represent data well and would lead to sub-optimal embedding.
- Multi-similarity[11]: In deep metric learning, a series of pair-based loss functions have been proposed. But none of them offer a unified framework to understand these loss functions. The structure called General Pair Weighting(GPW) regards the sampling problem in deep metric learning as a pair weighting problem through gradient analysis. Multi-similarity loss is within the framework of GPW. This method focuses on Self-similarity, Negative relative similarity, and Positive relative similarity[12][13]. It is mainly divided into two iterative steps, namely mining and weighting.
- Cosface[14]: CosFace is essentially a classification based on margin. It sets the margin on the result of cosine loss, that is, $cos(\theta) - m$, so the optimization is relatively simple. We consider the situation when the model initially starts to minimize the LMCL. Given a feature vector x, let $cos(\theta_i)$ and $cos(\theta_j)$ denote cosine scores of the two classes, respectively. Without normalization on features, the LMCL forces $\|x\|(cos(\theta_i) - m) > \|x\|cos(\theta_j$. Note that $cos(\theta_i)$ and $cos(\theta_j)$ can be initially comparable with each other. Thus, as long as $(cos(\theta_i) - m)$ is smaller than $cos(\theta_j)$, $\|x\|$ is required to decrease for minimizing the loss, which degenerates the optimization.
- Arcface[15]: Arcface is an algorithm training based on additive angular margin loss, which tries to divide the classification boundary directly in the angular space. Arcface and Sphereface and Cosface have certain points in common, but in ArcFace, the classification limit is maximized directly in the angular space, while Cosface maximizes the classification limit in the cosine space. ArcFace normalizes feature vectors and weights, and θ With the addition of the angular interval m, the angular interval has a more direct effect on the angle than the cosine interval. There is a constant linear angle margin geometrically.

C. Comparisons between Algorithms

To meet the control variables requirements, we designed a uniform evaluation measure and applied it to all seven algorithms. By selecting these algorithms reasonably and formulating a standard evaluation measure, the following two comparisons can be achieved by information summarized in table 1.

- Comparison between primitive algorithms and emerging algorithms
- Comparison of different network structures (BNInception and Resnet 50)

Although there are two different comparisons, in essence, each algorithm is trained on the same data set, and its performance is explored with the standard evaluation measure, and the respective evaluation results are used for comparison.

TABLE I
THE COMPARISON BETWEEN DIFFERENT ALGORITHM

Algorithms	Primitive Algorithms	Emerging Algorithms	BNInception	Resnet 50
Contrastive	✓	-	✓	-
Triplet	✓	-	✓	-
Proxy-NCA	-	✓	✓	-
Softtriple	-	✓	✓	-
Multi-similarity	-	✓	✓	-
Cosface	-	✓	-	✓
Arcface	-	✓	-	✓

After that, to show the pros and cons of each algorithm more intuitively, and compare them comprehensively, the process and results of this series of research are made into an end-to-end web application to meet the different needs of image retrieval enthusiasts and deep metric learning researchers.

Besides, to prevent unsolvable problems in the construction of these 7 algorithms, we have also selected several emerging algorithms as alternatives, including Normalized Softmax, FastAP, Signal to Noise Ratio Contrastive (SNR) , etc.

III. USER STORIES

The system consists of two main functions, the first one is simply retrieving the similar images of the input image, and another one is comparing the performance of different models.

The system has two kinds of user-oriented, the first one is researchers or fans on a field; another is researchers on deep metric learning. Currently, the models in the system have trained by datasets of birds and cars; therefore, the first scenario of the system is to find similar images of a kind of bird or a car that user input to the web service. The user's identity could be an ornithology student, car designer, or just a fan of one of these two fields, and the user might like to research the feature of a kind of bird or collect pictures of a car.

Our models are trained to focus on the datasets of birds and cars, while particular labels separate the datasets. The meaning of separating by label is that the images of a kind of bird or cars are trained together and the features of them are recorded. Which means our learning models could be more accurate. Besides, the size and amount of datasets can be more significant, it makes the system scalable.

The second function is usable for people who are studying deep metric learning models or researching it. The website of the system can show the retrieved results of different models at the same time, it is a direct view of the performance of those models. Therefore the second scenario is that if those people are trying to compare or record the performance of different models with different datasets, they could easily observe them on the website, they can estimate the performance themselves, and decide which model retrieves the result they like the most, and then determine their research direction.

The traditional ways of showing the comparison of learning models are displaying tables of data such as accuracy and recall, or a line chart or bar chart of different models. In this system, the retrieved results are shown to the users, and they can have a more intuitionistic sense of the performance, it is better than looking at numbers.

Due to the model in the system can be changed by upload or change the files of trained model parameters, it makes the system more flexible while changing of the models is much easier than running and deploying models by hand, this feature makes the system flexible. In this case, the system can implement different models to satisfy users' demands, which also makes the work of people who work on deep metric learning more comfortable and more convenient.

IV. IMPLEMENTATION

The implementation of our project is divided into two parts.

- Image retrieval system: The image retrieval system is essentially a web application, which introduces seven metric learning models trained by two chosen datasets and compares the performances of these models. It also implements functions to retrieve the top k most similar images from a query image using these different models and datasets.
- Models training and deploying: Training and deploying of our image retrieval metric learning models include building and training models with chosen image datasets as well as generating image embedding vector space files for these image datasets.

A. Image retrieval system

1) System Architecture: The image retrieval system mainly contains three parts, which are browsers, web servers and file data storage. Our Apache2 web server is deployed in a virtual machine on the Microsoft Azure cloud platform and all data is stored in the server as files.

All data includes model states, image embedding vectors generated by models, image path maps and image files. The data is not typically relational and is not modified frequently. Besides, data occupying the most space in our system is image datasets, which are binary image files, so relational databases are not used in this system.

For the front-end, we choose Bootstrap, a popular front-end toolkit with powerful JavaScript plugins to achieve adaptation to multiple devices of different sizes and dynamic effects, such as page rolling, mouse capturing and so on. To present our model comparison data more interactively, Vega-lite, an interactive data visualization tool, is used to generate dynamic charts from data defined in JSON objects.

For the back-end, we choose the Apache2 server, Flask and mod_wsgi server as the Web Server Gateway Interface to connect the apache server and flask python web application. Apache is suitable for handling our dynamic retrieving requests because of its dynamic computing and concurrency capabilities. It creates a new thread for each request and processes dynamic content within the server. Since our system

does not concentrate on high concurrency, Nginx is not our choice.

Flask is a micro web framework, which can be combined with many powerful Python libraries. It is compact, suitable for small or medium-sized projects with moderate complexity, and has strong scalability. Django and tornado are both large-scale frameworks and there is no need for us to use complex frameworks like them.

2) Website introduction: Our website contains four pages, which are search, algorithm, download, and evaluation.

The search page(Fig.1) is our home page. Users choose different datasets or models and upload a query image to retrieve similar images from our datasets or to evaluate all seven models.

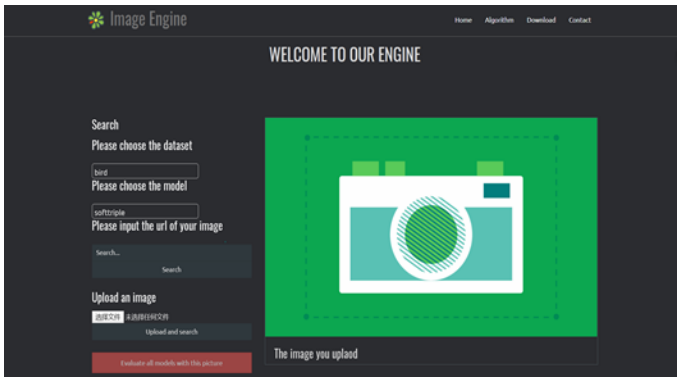

Fig. 1. Search page of website.

The evaluation page(Fig.2) is the page for users to submit their rates based on the top three most similar images retrieved by our seven models without knowing which result is returned by which model.

The algorithm page(Fig.3) contains text introductions of the seven models and interactive charts of model performance comparison data.

The download page provides links to the datasets we used and to our Github page.

The image retrieval workflow on the search page presents in the Figure 4.

First, the image uploaded by a user stores in the server, then after preprocessing, it is transformed into an embedding vector by our model. Later, we calculate the distance between this vector and all other vectors in the embedding space and find k closest ones. Finally, the corresponding k image files are retrieved and sent back to our website.

B. Models training and deploying

Our models trained in online labs, such as Azure machine learning or Google CoLab. And these online labs make training models in parallel possible.

1) Model training pipeline:

- Datasets: The two datasets used in this project, CUB-200 and Cars-196, give various performances in image retrieval.

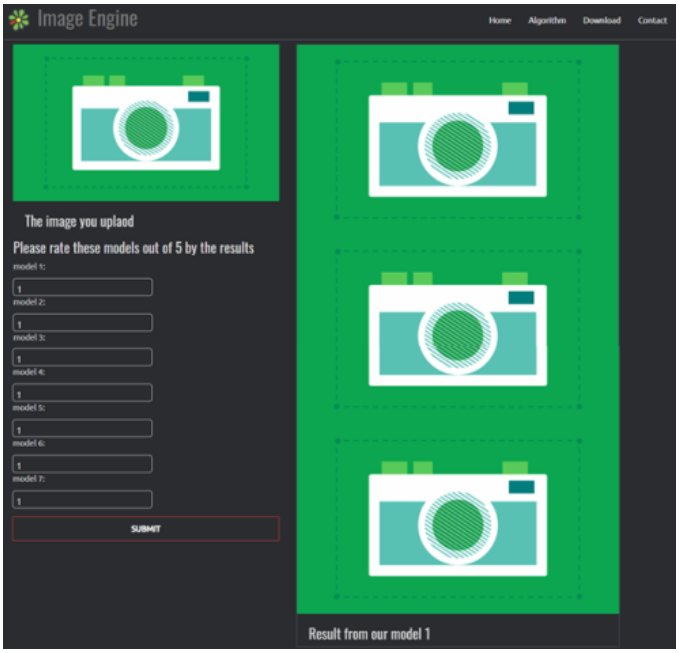

Fig. 2. Evaluation page of website.

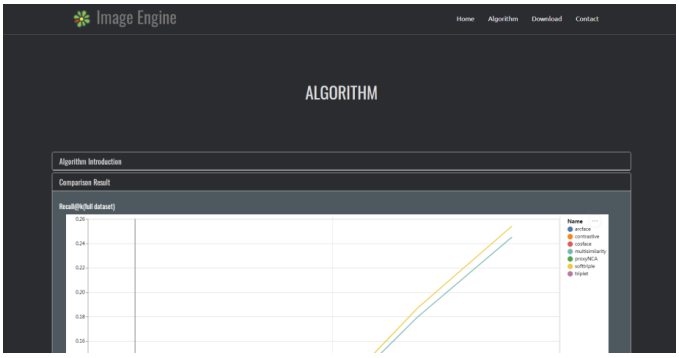

Fig. 3. Algorithm page of website.

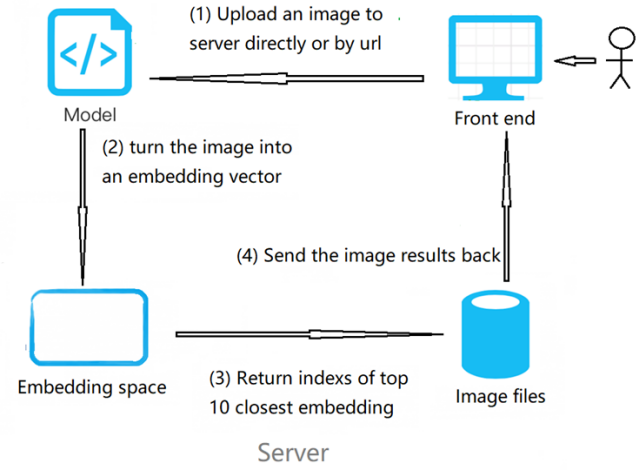

Fig. 4. The image retrieval workflow.

It is more challenging to train the Cars-196 dataset to get good results. Cars of the same class can look very different from different angles of view and images of different classes may look very similar, such as different series of the same brand.Therefore, CUB-200 is applied to all seven models but Cars-196 is used merely on models that deliver better performances.
Moreover, Cars-196 does not provide us with the labels or ground-truths for the testing image set, so we cannot calculate evaluation metrics, such as recall and precision, for the Cars-196 dataset. Also, for the same reason, the user rating evaluation is based on the CUB-200 dataset only as well.

- Dataset splitting: In the papers which propose models used in this project, those authors generally use half of the classes for training and the left classes for testing to get better generalization results when models are applied to new classes they have never come across. However, we split all images in each class almost equally into training and testing sets since we are using images in our datasets as retrieval results and do not require a high generalization performance. After testing, the splitting method we are using gives better results in our system.
- Data loading and transformation: We use torchvision to read datasets from image folders and load data in batches. And the transformations of images during datasets reading are chiefly color changes (RGB to BGR/Grayscale), resizing, cropping, vectorization (image to tensor) and normalization.
- Implementations of loss functions and network architectures: The three main network architectures used in our project are BN-Inception, ResNet50 and Siamese network. The basic loss functions are contrastive loss, triplet loss, cross-entropy loss, SoftMax loss or their variants.
- Model training and validating: We tuned all model parameters into the best versions according to original papers introducing each model.
- Model saving: We use the torch package to save the dictionaries of states for each model, which occupies smaller storage space and causes fewer bugs.
- Embedding space extracting: Firstly, we build the network architectures again and read the model states in. And we read full image datasets according to the sequence of image folders and do the transformations again. Then, put each image vector into our model to get the embedding vector in sequence. Finally, save image embedding vectors and image path maps separately to files.

2) Model deploying: Figure 5 shows the objects deployed as well as the relationship between them and how we scale more models and datasets in this system.

All light blue objects in Figure 5 are replaceable, which means we can generate diverse models and embedding spaces with different image datasets. When we want to extend more models, we just need to put all replaceable objects into our server as files and update the website to provide more selections to users. That's how we scale more models on our web server. For scaling more datasets, it is similar. We put more datasets on our web server, and then we can add models trained by these new datasets.

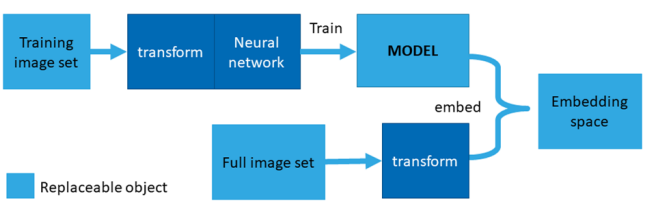

Fig. 5. Model deploying pipeline.

V. Evaluation

We evaluate our models' performance from two different aspects: evaluation metric and user evaluation.

A. Evaluation metric

Evaluation by mathematical indicators, that is, by randomly extracting a considerable number of pictures (200) from the test set, then input these pictures into the system and counting the number of correct pictures (pictures with the same label) in the result to see the model's performance. Among them, we have selected two more intuitive parameters, namely precision@k and recall@k. By comparing these parameters between models, we can get the mathematical evaluation of the corresponding models.

In practice, we have found that some better models can often return a large number of correct images of the same type. For example, soft-triple and multi-similarity, their accuracy and recall rate remain high in many cases. Therefore, in this case, we introduced the concept of 'accuracy and recall based on the test set'.

When establishing the embedding space, we input all images from the entire data set into the model and complete the embedding, including the training set and the test set. However, for the model itself, images in the test set are not involved in the training, so they are unknown images. The image used for the test also comes from the test set. Therefore, when images returned by the model include images from the same label and also belonging to the test set, we can regard that the model has completed the correct matching of two completely unknown images, compared to correcting an unknown picture to a known picture, the former is obviously more valuable. Therefore, we introduce the model's evaluation based on the accuracy and recall of the test set in the Figure 6.

We implement our work from two aspects that we mentioned above.

The first is the comparison of primitive algorithms and emerging algorithms. From the table 2, we could find that the primitive algorithms, Triplet and contrastive receive a

TABLE II
COMPARISON BETWEEN EACH ALGORITHM WITH FULL DATASET

Algorithm	precision@5	precision@10	precision@20	precision@30	recall@5	recall@10	recall@20	recall@30
Multi-similarity	0.692	0.6105	0.531	0.483	0.0586	0.1033	0.1798	0.2452
Triplet	0.22	0.116	0.0638	0.0468	0.0186	0.0197	0.0216	0.0238
Contrastive	0.213	0.114	0.0633	0.0478	0.0181	0.0193	0.0214	0.0243
Cosface	0.007	0.005	0.0063	0.0062	0.0006	0.0008	0.0021	0.0031
Softtriple	0.7	0.6295	0.5533	0.5012	0.0592	0.1064	0.1872	0.2542
Arcface	0.006	0.0055	0.0048	0.0045	0.0005	0.0009	0.0016	0.0023
Proxy-NCA	0.094	0.072	0.055	0.0465	0.0079	0.0121	0.0185	0.0235

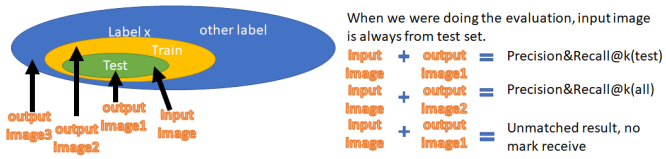

Fig. 6. Evaluation of model.

medium performance compared with emerging algorithms. If we ranked them vertically, these two are much better than Cosface, Arcface and Proxy-NCA, but their performance is poor compared with Multi-similarity and Softtriple. So the conclusion is, emerging algorithms is not always better than the primitive algorithms.

The second is from the aspect of different network structures. Arcface and Cosface are based on Resnet50, while proxy-NCA and Softtriple, Multi-similarity, are based on the BNInception network. Also from the table 2, we can say that the BNInception network's performance is much better than the Resnet 50. If we put our eyes on the worst algorithm, the proxyNCA, its performance is ten times better than Arcface. So here we can say, the advantage of the BNInception network is evident in our work.

Which one is the best? We need a more convincing evidence——the precision and recall on the test set. Overall, all results in table 3 were lower than on the full dataset, but we could find something interesting.

The first is the comparison between Multi-similarity and Softtriple. We see that the former has overtaken the latter on the test set. The performance of multi-similarity is better than Softtriple on the two indicators of precision@5 and recall@5. This shows that the former is better than the latter in processing unknown pictures and has higher robustness.

So the conclusion here is, multi-similarity has a better generalization ability, but Softtriple gives better performance on numbers.

What is interest here is the performance of Triplet and Contrastive. Reset 5 algorithm's' performance dropped on the test set, but these two have remained still. After discussions about this phenomenon, it could be explained in two aspects: First is the overfitting of other models, like Multi-similarity and Softtriple. The second is the underfitting of these two models. The network of contrastive and Triplet are quite simple compared with BNInception and Resnet50, and it is the combo of Siamese network and simple CNNs. The limit of the network leads to this situation.

B. User evaluation

We collected a total of fewer than 30 people's evaluation opinions(blind evaluation) and then standardized their scores (1-5) one by one and averaged them as the model's user score (in range 0-1). The performance on recall and precision is also treated in the same way(shown in figure 7).

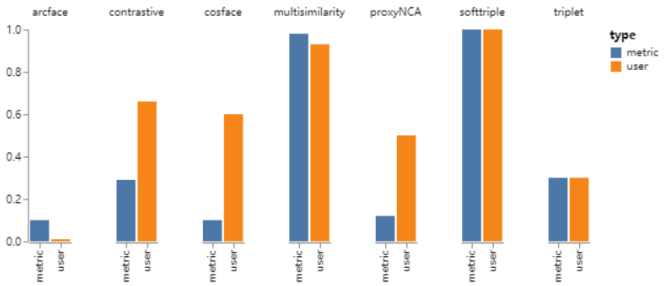

Fig. 7. The users' opinion has some differences compared with the precision and recall.

For the most ideal situation, the angle of user evaluation should be consistent with our statistics, that is, the higher the precision and recall of the model, the higher the user score. However, the reality is different from this idea. We observed that compared with Triplet, Contrastive is almost the same in terms of accuracy and recall, but the user scores gained by Contrastive are significantly higher than the Triplet. Besides, Proxy-NCA and Cosface also have the same situation, which is worth discussing.

After testing and going through the image uploaded by our user, we summarized some of the reasons for this phenomenon and found some way to explain it:

- We believe that precision can be the user's first evaluation element; for example, users give the highest scores for Multi-similarity and Softtriple. However, it has a threshold, that is, when the precision is high enough, the user

TABLE III
COMPARISON BETWEEN EACH ALGORITHM WITH TEST DATASET

Algorithm	precision@5	precision@10	precision@20	precision@30	recall@5	recall@10	recall@20	recall@30
Multi-similarity	0.417	0.3255	0.2615	0.2302	0.0716	0.1116	0.1792	0.2362
Triplet	0.207	0.1075	0.0563	0.0392	0.0357	0.0372	0.0391	0.0409
Contrastive	0.202	0.1055	0.0558	0.0402	0.0349	0.0364	0.0385	0.0415
Cosface	0	0.001	0.0017	0.0018	0	0.0003	0.0012	0.0019
Softtriple	0.414	0.33	0.2648	0.2343	0.0711	0.113	0.1817	0.241
Arcface	0.004	0.004	0.0033	0.003	0.0007	0.0014	0.0022	0.003
Proxy-NCA	0.055	0.043	0.0305	0.0252	0.0093	0.0145	0.0206	0.0255

will give a higher evaluation to the model. Nevertheless, when the accuracy is low to a specific value, other factors will replace precision as the evaluation element under the human perspective. User evaluation and precision will no longer be positively correlated.
The concept of 'threshold' here is just a conjecture. We did not accurately confirm this number because it is beyond the scope of the project.

- The images Contrastive, Cosface and Proxy-NCA shown to users contain many extra features beyond the bird species. Such as bird's posture and background color. So maybe inside these models, the features mastered by the model are different from other models.
  We use the images uploaded by the user to test and see what happens when they are doing the evaluation.
  We noticed a bird in a specie that is not included in our dataset. The Cosface returned several similar pictures of birds with the same shape and color, proxyNCA and contrastive also do the same thing. In this situation, Multi-similarity and Softtriple did not show any advantages, and the pictures they returned were quite different (if it was a picture included in the test set, the pictures they returned were very similar.)
- Some user uploaded some images from car dataset and other strange images which might contain any objects but a bird. In this situation, all the models are at the same start line, the model which returned similar tones scored higher.

So the conclusion here is: The model not only has the generalization ability relative to birds but also has similar generalization ability for other objects. These capabilities are not enough to be measured by the precision and recall of the bird data set, but we see the potential of these models through users' strange behavior.

VI. CONCLUSION AND FUTURE WORK

In recent years, deep metric learning based on neural networks seems to be a required study field for computer vision researchers. The goal of deep metric learning is to learn a similarity metric that computes two or more objects' similarity or dissimilarity while using sample data. Most of the recent studies in the literature were inspired by Siamese and Triplet networks in deep metric learning, and as the comparison result for seven models that we implemented in our web application, soft-triple and multi-similarity algorithm perform better than other models both in numeric indicator evaluation and online user comparative evaluation. We strongly believe that the deep metric learning algorithm's power depends on the deep neural network structure, which means the better performance of the two algorithms mentioned above related to the use of inception networks. Furthermore, the new model works better than the old one based on using a more complicated network.

Our project's critical success is building the highly extendable standard training pipeline for deep metric learning algorithms to get it ready for a fair comparison with the same training environment setting. Except for the algorithms implemented in the web application, a new metric learning algorithm could be added in the system by providing the network's core structure and pre-processing image transformations.

In future work, we could extend the dynamic comparison function that allows deep metric researchers to add and compare their model's performance by uploading the parameter file of the model and core functionality code. Besides, we can change the static comparison result graph into a dynamic graph, which could automatically update based on new user evaluation data. In addition, we could add more datasets to get rid of the dataset bias, which means some algorithms could perform correctly in a specific dataset but very bad in the other one. The user needs to select the dataset that they want to retrieve, in the future work, we can combine the datasets to make the system retrieve related pictures no matter what object the user upload. Besides, we could add a feature that makes users available to choose how many pictures they want to retrieve instead of the default ten pictures at the current stage.

ACKNOWLEDGMENT

We would strongly like to acknowledge to our mentors: Mo Karzand and Filipe Figueiredo, the completion and success of this project is inseparable from their suggestions and guidance. We also express same acknowledgment to our mentors of UCD, who provided valuable suggestions and directions for this project. What's more, this project is supported by Microsoft Team.

References

[1] A. Bellet, A. Habrard, and M. Sebban, "A survey on metric learning for feature vectors and structured data," *arXiv preprint arXiv:1306.6709*, 2013.

[2] E. P. Xing, M. I. Jordan, S. J. Russell, and A. Y. Ng, "Distance metric learning with application to clustering with side-information," in *Advances in neural information processing systems*, 2003, pp. 521–528.

[3] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 815–823.

[4] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie, "The caltech-ucsd birds-200-2011 dataset," 2011.

[5] J. Krause, M. Stark, J. Deng, and L. Fei-Fei, "3d object representations for fine-grained categorization," in *Proceedings of the IEEE international conference on computer vision workshops*, 2013, pp. 554–561.

[6] K. Musgrave, S. Belongie, and S.-N. Lim, "A metric learning reality check," *arXiv preprint arXiv:2003.08505*, 2020.

[7] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 1735–1742.

[8] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 360–368.

[9] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6450–6458.

[10] Q. Qian, R. Jin, S. Zhu, and Y. Lin, "Fine-grained visual categorization via multi-stage metric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3716–3724.

[11] X. Wang, X. Han, W. Huang, D. Dong, and M. R. Scott, "Multi-similarity loss with general pair weighting for deep metric learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 5022–5030.

[12] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese, "Deep metric learning via lifted structured feature embedding," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4004–4012.

[13] D. Yi, Z. Lei, S. Liao, and S. Z. Li, "Deep metric learning for person re-identification," in *2014 22nd International Conference on Pattern Recognition*, IEEE, 2014, pp. 34–39.

[14] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou, Z. Li, and W. Liu, "Cosface: Large margin cosine loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5265–5274.

[15] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4690–4699.