Exploring Deep Neural Networks for Cat Face Embeddings

Elizabeth Witten
Northeastern University
Boston, USA
witten.e@northeastern.edu

***Abstract*—This paper presents an implementation of a deep neural network for learning cat face embeddings, utilizing transfer learning from the pre-trained GoogLeNet model. The proposed model is trained using triplet margin loss to learn a 256-D embedding space where images of the same subject's face are located close together. Finally, the resulting embedding space is analyzed from two perspectives to identify potential areas for improvement for the task of cat identification. First, the entire space is visualized using the t-SNE algorithm to project it into two dimensions. Second, the trained model is used to create a labeled database of cat face embeddings. A command-line interface (CLI) application is developed to interface with the database, allowing for the addition of new images and performing top-k retrieval for query images of cats.**

***Index Terms*—computer vision, deep neural networks, learned image embeddings**

I. Introduction

Face recognition has gained significant attention in recent years, with applications ranging from security systems to social media. Even pets have not been excluded from this field of study, with products already available in the market that use face recognition for individual pets. These products not only provide entertainment but also have practical applications in monitoring pet health and safety. For example, a feeding station could enforce a special diet for specific individuals, or access gates can block off restricted areas of the home. These applications can be valuable for individual pet owners as well as for larger settings such as animal shelters or sanctuaries.

The goal of this project is to explore the use of deep neural networks for learning cat face embeddings, which are representations of cat faces in a lower-dimensional space. Once generated, these embeddings can be used for various retrieval or identification tasks. The proposed embedding model makes use of transfer learning from the GoogLeNet model, a well-known deep neural network architecture pre-trained on the large-scale ImageNet dataset, to produce 256-D embeddings for cat faces. Transfer learning leverages the knowledge learned from ImageNet for the task of cat identification, potentially improving the performance of the model.

In addition to transfer learning, the training process of the proposed embedding model uses triplet margin loss, which is a loss function that helps the model learn to differentiate between subjects in the embedding space more efficiently. This ensures that the learned embeddings effectively capture the unique features of different cat faces, and allows for more accurate retrieval and identification.

Once the embedding space is learned, several tasks are performed to analyze the quality of the learned embeddings. This includes visualizing the entire embedding space using techniques such as the t-SNE algorithm, which can provide insights into the distribution and clustering of the embeddings. Additionally, retrieval tasks are performed to evaluate the effectiveness of the learned embeddings in real-world scenarios. To aid in these evaluations, a command-line interface (CLI) application was developed, allowing for easy interaction with the learned embeddings and performing retrieval tasks.

The rest of this paper is structured as follows. Section II provides an overview of relevant research. Section III describes the proposed embedding model implementation and training process in detail. Section IV presents evaluation and analysis of the learned embedding model. Lastly, Section V concludes the paper with a summary of findings and directions for future research.

II. Related Work

A. Learning Visual Similarity

One method for learning visual similarity was proposed by Bell and Bala [1], for the application of product design search. One of their approaches uses convolutional neural networks (CNNs) with a siamese architecture, where pairs of product images are input into two parallel copies of a CNN with shared model weights. They trained CNNs on a large dataset of product images and similarity ratings, using a contrastive loss function to compute similarity scores. Their experiments showed that CNNs can effectively learn visual similarity, with the best architecture being a siamese GoogLeNet trained using cosine distance as the metric.

GoogLeNet [2] is a well-known CNN architecture that uses Inception, which uses multiple parallel convolutional pathways of different filter sizes in a single layer, whose outputs are merged together. This allows for efficient feature extraction, reducing computational cost while maintaining high accuracy. The GoogLeNet architecture was trained on ImageNet and achieved top performance in ILSVRC 2014.

Inspired by these works, this project aims to learn cat face embeddings using GoogLeNet and cosine distance as the similarity metric.

B. FaceNet

While [1] focus on furniture and other household products, Schroff et al. [3] address the problem of learned embeddings for face recognition and clustering tasks with their proposed model, FaceNet. FaceNet aims to map face images into a compact and discriminative embedding space, where the distances between the embeddings can measure facial similarity. The first key component of FaceNet is the use of triplet loss, which encourages similar faces to have close embeddings, and dissimilar faces to have distant embeddings. During training, the hardest triplets are selected, making the task of discriminating between them more challenging for the neural network.

In addition to triplet loss, FaceNet also employs a CNN architecture and a face alignment technique as other key components. The learned embeddings can be utilized for various tasks, such as face verification, recognition, and clustering, and have achieved state-of-the-art results on benchmark face recognition datasets. Similarly, in the training process for cat embeddings in this project, triplet loss is also used as the training criterion.

C. Cat Head Detection

Related to the downstream task of cat identification, a Microsoft research team [4] has developed an approach to detect cat heads in images by effectively combining shape and texture features. Their proposed shape model captures the geometrical structure of cat heads, while their texture model captures characteristic texture patterns. These two models are combined into a single joint detection framework, which has been shown to be effective at detecting cat heads with varying poses, lighting conditions, and textures.

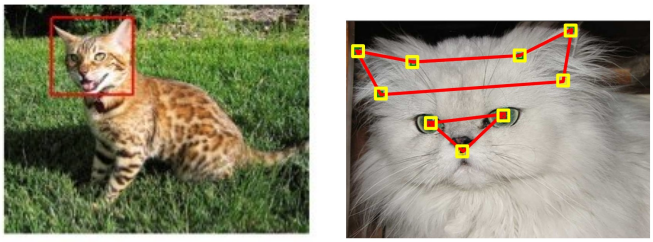
Furthermore, in addition to proposing these models, [4] has also released a large database of annotated cat faces, primarily focusing on frontal views. This database has been used as training data for the OpenCV cat face detector, implemented by Joseph Howse [5]. Similarly, in this project, data will be drawn from the Microsoft cat dataset for training and evaluation purposes.

III. Methodology

This section details the implementation of the learned cat face embedding. It covers data selection and processing, as discussed in Section III-A, and the training implementation, described in Section III-B. Source code for this project is available on GitHub[1], and the dataset can be made available upon request.

A. Data

1) Data Sources: The dataset utilized for training the embedding model was obtained from two existing cat datasets. These datasets were chosen based on their large size, which increases the likelihood of multiple images of the same subject being available, as well as their provided face annotations. The first dataset used is the Oxford-IIIT PET dataset [6], comprised of 2,371 cat images, each annotated with a tight bounding box around the head (see Figure 1(a)). The second dataset is the Microsoft 2008 cat dataset [4], which includes 10,000 cat images primarily in frontal view, annotated with nine landmark points for the mouth, eyes, and ears (see Figure 1(b)). To reduce labeling effort, only the CAT_00 and CAT_01 splits of the Microsoft 2008 cat dataset were used, resulting in a total of 3,320 cat images. Both datasets were accessed in April 2023 and downloaded from Kaggle [2][3].



(a) Oxford-IIIT annotations [6] (b) Microsoft cat annotations [4]

Fig. 1. Annotation examples from each data source.

2) Subject Labeling: From the two data sources described above, each of the 5,691 images was manually labeled with a unique cat subject identifier produced using a UUID v4 generator. Subjects that only appeared in one image were discarded to ensure a sufficient number of samples per subject. The resulting labeled dataset consists of 407 subjects and a total of 1,208 images. Specifically, 632 images of 204 subjects are sourced from the Microsoft cat dataset, while the remaining 576 images of 203 subjects are from the Oxford-IIIT PET dataset.

3) Annotation Standardization: The face annotations from the separate datasets had to be standardized into a single CSV file, where each row contains the image file name and four bounding box values (*xmin, ymin, xmax, ymax*).

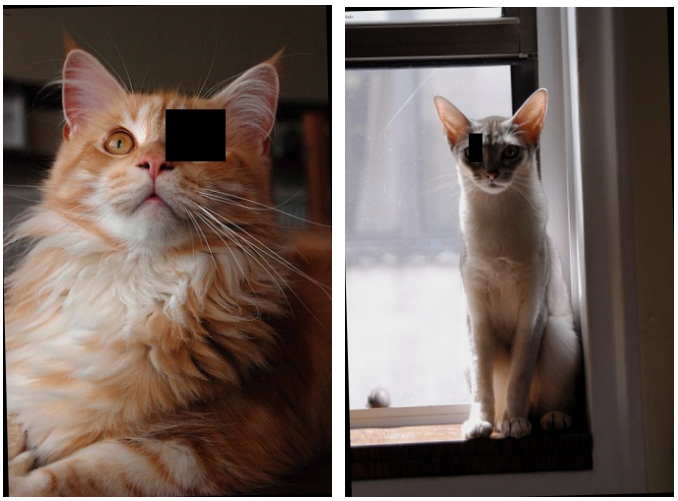
For the Oxford-IIIT PET dataset, the bounding box information was obtained from the provided XML annotation files. However, it was found that 265 images in the Oxford-IIIT PET dataset were missing annotation files. To address this, these images were manually annotated with face bounding boxes using Label Studio, which produced a JSON-formatted annotation file. The bounding box information was then extracted from the JSON file.

For the Microsoft cat dataset, the locations of the nine landmark points are stored in plain text files. Since the lower extent of the face is unspecifed by the landmarks, the face is estimated to be a square centered between the eyes, excluding the ear tips. The implementation was inspired by that of Joseph Howse's ISMAR 2014 demonstration, which also used the Microsoft cat dataset to train the cat face detector in OpenCV [5]. Since the Oxford-IIIT PET dataset annotations include

[1]https://github.com/wittene/cat-face-embeddings

[2]https://www.kaggle.com/datasets/zippyz/cats-and-dogs-breeds-classification-oxford-dataset

[3]https://www.kaggle.com/datasets/crawford/cat-dataset



(a) (b)

Fig. 2. Examples of data augmentation output.

the ears, the resulting face bounding box was extended on the top and sides to also include the ears. It should be noted that unlike [4] and [5], no rotation was performed, and the bounding boxes are axis-aligned.

4) Train/Validation/Test Splitting: To achieve balanced train/validation/test splits, each subject is individually divided. The main criterion for splitting the data is that each subject must have at least two images in the training set. This ensures that at least one positive-anchor-negative training triplet can be created for each subject. Once two images are allocated to the training set, the remaining images are evenly distributed among the three splits in the following order: test, train, validation. As a result, the training set contains 879 images for all 407 subjects, the validation set contains 110 images for 78 subjects, and the test set contains 219 images for 177 subjects.

5) Image Augmentation: To augment the training set, which is relatively small for a face identification task, data augmentation techniques are applied to generate three additional images from each original training sample. For each augmented image, three different techniques are applied to the original image. The first technique involves random rotation within 15 degrees. The second technique randomly scales the pixel values by a small amount to vary the brightness and contrast. The third technique simulates occlusion by erasing a randomly sized rectangle and setting the pixels to black. Example outputs are shown in Figure 2. The purpose of this augmentation process is to increase the diversity and size of the training set, enhancing the model's ability to generalize to unseen data.

6) Online Image Processing: During training or evaluation, the image data is loaded and resized to 224x224 with padding. This is achieved by first cropping the image to the face boundary box as defined in the annotation file. Then, the cropped image is resized to a maximum of 224x224 dimensions, while preserving the aspect ratio of the cat face. Finally, zero padding

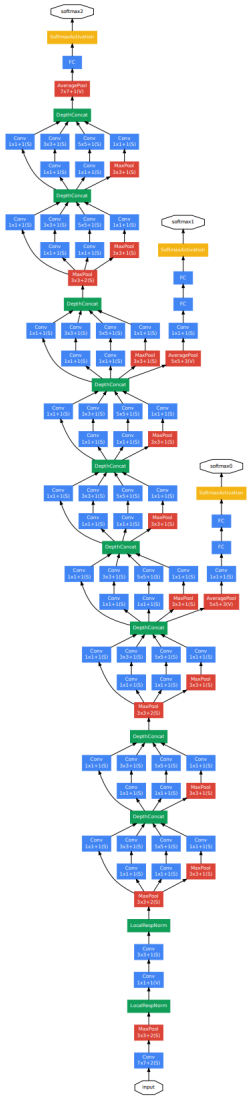


Fig. 3. GoogLeNet architecture (image sourced from [2]).

is applied to fill the 224x224 square, while ensuring the cat face remains centered. The use of padding allows for the preservation of spatial information, even when cat faces are of varying shapes.

B. Training the Model

1) Model Architecture: The proposed cat face embedding model is based on GoogLeNet [2], which serves as the base model. The full model architecture is shown in Figure 3. To leverage transfer learning, the pre-trained weights from training on ImageNet are loaded into the embedding model. The original 1000-D fully-connected layer is replaced with a 256-D fully-connected layer, which serves as the final image embedding vector, and the weights for all layers up to the final layer are frozen, allowing only the embedding layer to be updated during the training process. This enables the embedding model to take advantage of the sophisticated

Fig. 4. Training performance over 100 epochs.

learned features from training on the large ImageNet database, which improves training efficiency, particularly with a small training set.

2) Triplet Mining Technique: The criterion for learning the embedding is the triplet margin loss function. Each triplet is composed of an anchor a, a positive example p (an image of the same subject as the anchor), and a negative example n (an image of a different subject from the anchor). The triple margin loss function attempts to minimize the distance between the anchor and positive example to be as small as possible, while aiming for the distance between the anchor and negative example to exceed a given margin μ. As a result, the embedding model is trained to learn compact representations that effectively encode the distinctive features of individual cat faces, while also enabling discrimination between different subjects.

The loss function for a single triplet is given by:

$$L(a, p, n) = max\{d(a, p) - d(a, n) + \mu, 0\}$$

where d represents the distance metric. The details of this equation are explored by [7]. In this implementation, cosine distance is used as the distance metric, which measures the angular similarity between two vectors.

In order to compute this loss function, an online triplet mining technique is applied to sample the hardest triplets in a batch. This technique ensures that the model focuses on the most challenging examples, improving training efficiency. This is done by first computing a pairwise cosine distance matrix from the predicted embeddings. For each training sample as the anchor, the *hardest positive example* and the *hardest negative example* are chosen form the set of hardest triplets. The *hardest positive example* is chosen as the sample with the same subject identifier as and the greatest distance from the anchor. The *hardest negative example* is chosen as the sample with the a different subject identifier from and the least distance from the anchor. The batch loss is computed as the mean triplet margin loss across the hardest triplets.

3) Training Process: The training process involved standard backpropagation with the Adam optimizer and triplet margin loss as the criterion. The model was trained for 100 epochs using an NVIDIA Tesla V100 SXM2 GPU, with checkpoints saved after each epoch to track the best model, the latest model, and training performance data (see Figure 4). The best model was determined based on top-k retrieval accuracy scores using the validation set. To perform retrieval, the training data embeddings were compiled into a single database tensor, and the closest 100 matches were retrieved for each validation sample using cosine similarity. Validation metrics were computed for different threshold values of k (1, 5, 10, 25, 50, 100) by tracking whether the correct label appeared in the top-k matches. The model with the highest top-1 accuracy was selected for comparison. In case of ties, top-5 accuracy was considered, and so on.

Additionally, a simple exhaustive search was performed to compare different training parameters, including batch size (128 or 256), learning rate (0.1 or 0.01), and loss margin (0.1 or 0.01). The validation performance of the best model was not significantly affected by the learning rate, as the validation performance plateaued before completing 100 epochs. Among the tested parameters, a batch size of 128 and a loss margin of 0.1 performed the best, although the difference was marginal. Thus, the final model used for evaluation was trained with a batch size of 128, a learning rate of 0.01, and a loss margin of 0.1.

IV. Experiments and Results

This section provides an evaluation of the trained model. It begins with an overview of the learned embedding space, visualizing the distribution of embeddings in Section IV-B. Next, the model's performance on a top-k retrieval task is assessed in Section IV-C, reporting both top-k accuracy and precision@k.

A. Database Generation

To conduct evaluation, the embedding space must be populated with labeled data points. This is achieved by generating a database of embeddings from the training set, where each image is passed through the model to obtain its embedding vector. The subject identifier and embedding are recorded, creating a ready-to-use database for evaluation purposes. The same database is used for all evaluation tasks.

B. Visualization

To visualize the proposed embedding space, the t-SNE (t-distributed Stochastic Neighbor Embedding) algorithm is employed, following the approach used in [1]. This technique reduces the dimensionality of the embeddings from 256-D to 2-D, while preserving local similarities among data points [8]. The resulting 2-D scatter plot is annotated with representative images of associated subjects to add meaning to the plot. It's important to note that the representative image is chosen as the first located image of a particular subject, which aids in

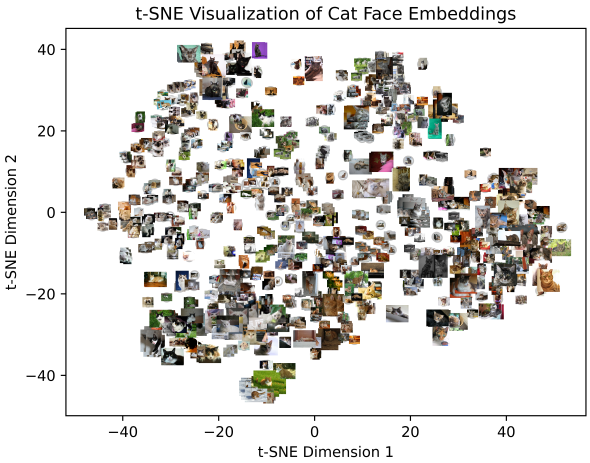


Fig. 5. 2-D embedding visualization with representative images, generated using t-SNE [8]. Best viewed on a monitor for full resolution.

identifying clusters of single subjects or outliers in the scatter plot. Figure 5 shows the final visualization.

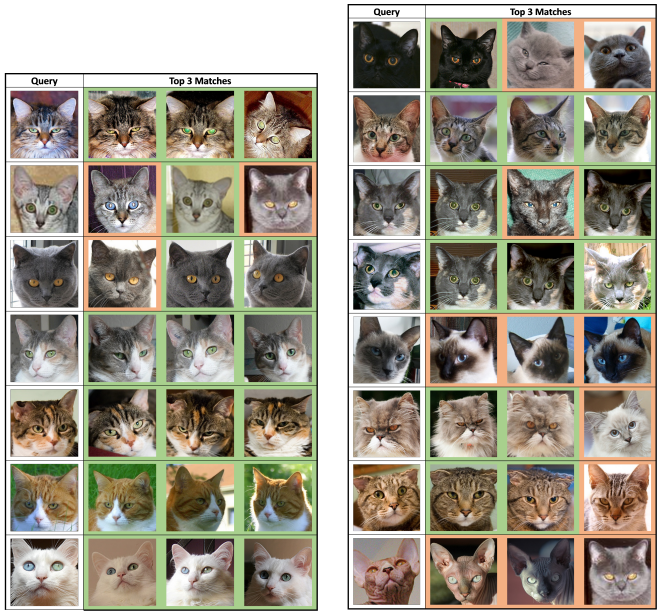
Upon examination of the scatter plot, it is observed that the largest clusters are color-based, with texture and coat pattern also playing a significant role, as seen by the proximity of the orange tabby and brown tabby clusters. On the other hand, facial structure–such as the shape of the face and head–appears to be a less important feature, as viewed in the 2-D projection. This may aid in identifying cats in different poses. As expected, subjects with unique facial markings tend to cluster more tightly together, while those without, such as the gray British shorthair cats and Siamese cats, tend to blend into other subjects of the same breed.

C. CLI program

To evaluate retrieval performance, a simple CLI application was developed to interact with the database. This application offers three features: (1) top-k subject retrieval given a single query image, (2) addition of new labeled images to the session database, and (3) saving the session database to a new or existing file.

Expanding on feature (1), top-k retrieval is implemented very similarly to the validation described in Section III-B and returns a list of subject identifiers. To obtain the query embedding, a face must be located in the input image. The OpenCV Haar cascade cat face detector implemented by [5] first attempts to return a face bounding box. If no cat faces are detected by the classifier, then the face is naively estimated to be in a centered square 3/4 the length of the image's shortest side. From there, image is resized and padded as described in Section III-A and passed to the model to obtain the query embedding.

Instructions for setting up and running this application can be found in the project repository. The rest of this section evaluates performance on different retrieval tasks using query images not seen during training.



(a) Retrieval results for query images with detected faces

(b) Retrieval results for query images with estimated faces

Fig. 6. Retrieval results for known subjects of varying colors and patterns, including queries with detected faces (left) and estimated faces (right). The first column displays the query image, while the remaining columns show the top-3 results. Green background indicates a subject match, while orange background indicates a non-match. Please note that the cropping applied in this chart is approximate and for illustrative purposes.

Face detected?	Top-1 Accuracy	Top-3 Accuracy	Precision@3
Yes	0.71	1.00	0.86
No	0.75	0.75	0.54

TABLE I
RETRIEVAL METRICS FOR KNOWN SUBJECTS.

1) Retrieval of Known Subjects: In this retrieval task, a set of query images are drawn from the test set at random, with the goal of evaluating the top-3 retrieval for a wide range of cats. Because these subjects are drawn from the test set, the model has seen other images of the same subjects during both training and validation. In this query set, faces could be automatically detected in half of the images, while the other half had to be estimated. The results of this task can be seen in Figure 6.

The metrics evaluated for this task are top-1 accuracy, top-3 accuracy, and precision@3 (which measures the proportion of relevant documents among the top k retrieved documents), and the results are shown in Table I. For the detected faces, five out of seven subjects are correctly identified in the top result, and all subjects are correctly identified in the top three matches. The precision@3 for the detected faces is 0.86. For the estimated faces, six out of eight subjects are correctly identified in the top result, and the same six subjects are correctly identified in the top three matches. The precision@3 for the estimated faces is 0.54.

As expected, the precision@3 for the estimated faces is lower compared to the detected faces. However, it is worth noting that the top-1 accuracy is comparable for both subsets,

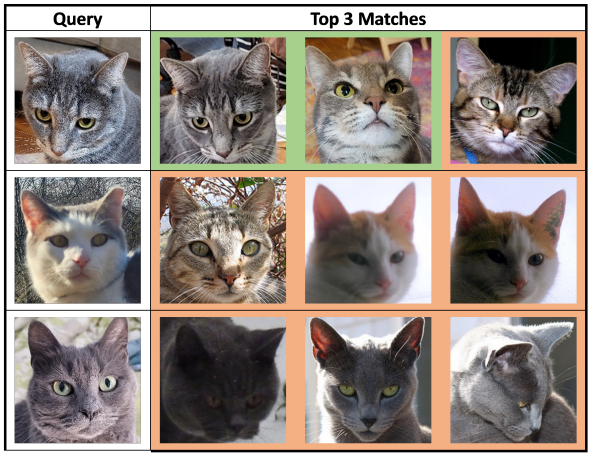


Fig. 7. Retrieval results for new subjects. The first column displays the query image, while the remaining columns show the top-3 results. Green background indicates a subject match, while orange background indicates a non-match. Please note that the cropping applied in this chart is approximate and for illustrative purposes.

Face detected?	Top-1 Accuracy	Top-3 Accuracy	Precision@3
Yes	-	-	-
No	0.33	0.33	0.22

TABLE II
RETRIEVAL METRICS FOR NEW SUBJECTS.

indicating that the model performs well even with partial faces due to a misaligned bounding box. It is also possible that the estimated bounding box misses the face but captures the body, suggesting that body color and texture may also contribute to successful retrievals, as embeddings of the body and face may be close enough for matching.

The cats that are more difficult to retrieve have less distinctive facial markings. The Siamese cat closely resembles other Siamese cats, and the Sphynx cat lacks a distinguishing coat pattern. Despite these challenges, the incorrectly matched subjects still exhibit visual similarity to the query subject. Moreover, the retrieval matches were successful even across variations in head positions, backgrounds, and lighting conditions.

2) Retrieval of New Subjects: In this retrieval task, a small set of query images are drawn from a set of images excluded from the train, validation, and test sets. As a result, the model has not seen any of the subjects before. Before retrieval, three to four images of each new subject are added to the session database.

As in the previous retrieval task, the metrics evaluated are top-1 accuracy, top-3 accuracy, and precision@3. The results can be seen in Figure 6 and Table II. For all three query images, the face is not detected and has to be estimated. For these new faces, one out of three subjects are correctly identified in the top result, and the same one subject is correctly identified in the top three matches. The precision@3 is 0.22.

As expected, retrieving new subjects not seen during training is more challenging. However, although largely incorrect, the retrieval results are still visually similar and reasonable. For the first query, the tabby, the only incorrect match is another tabby of similar color. For the second query, the subject in the second and third matches looks almost identical, the only difference being orange markings instead of gray ones. Finally, for the third query, the top three matches are all gray cats with a similar fur texture.

V. DISCUSSION

The learned embedding in this project demonstrates good performance on retrieval tasks. Although the top-1 accuracy of the model needs improvement to enhance its practical usefulness as a cat identification tool, the model provides reasonable predictions even when faces were not found. Additionally, the model showed robustness to pose and lighting variations, indicating its potential for real-world cat identification applications.

The model's success with a limited trained set size can be attributed in part to the use of transfer learning from the pre-trained GoogLeNet model. GoogLeNet's sophisticated architecture and extensive training on the large ImageNet dataset provided the embedding model with a strong foundation of features.

Further research could explore variations in the base model. For instance, unfreezing more layers of GoogLeNet or training from scratch could be explored to determine their impact on performance. Additionally, other pre-trained models such as AlexNet [9] could be compared against GoogLeNet to evaluate their effectiveness for the cat identification task.

Another potential avenue for future investigation is to vary the training data and observe how different factors, such as the number of images per cat and the distribution of cat types in the training set, impact the learned embedding. Additionally, more sophisticated image pre-processing and data augmentation techniques could be employed to further enhance the model's performance. For example, [3] and [4] use face landmark points to normalize the facial features.

Overall, the learned embedding showed promising performance on cat retrieval tasks, demonstrating robustness to challenges such as cut-off faces, pose, and lighting variations. The use of transfer learning from the pre-trained GoogLeNet model was beneficial, and further research can be conducted to optimize the training data and explore variations in base models for continued improvement of the cat identification system. Once the embedding model is improved, it can be applied to many real-world tasks, such as monitoring cats in a shelter or sanctuary, managing cat-restricted locations, or running electronic pet toys.

REFERENCES

[1] S. Bell and K. Bala, "Learning visual similarity for product design with convolutional neural networks," *ACM Trans. Graph.*, vol. 34, jul 2015.
[2] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[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 (CVPR)*, June 2015.

[4] W. Zhang, J. Sun, and X. Tang, "Cat head detection-how to effectively exploit shape and texture features," in *Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part IV 10*, pp. 802–816, Springer, 2008.

[5] J. Howse, "Training detectors and recognizers in python and opencv," in *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, (Los Alamitos, CA, USA), pp. 1–2, IEEE Computer Society, sep 2014.

[6] O. M. Parkhi, A. Vedaldi, A. Zisserman, and C. V. Jawahar, "Cats and dogs," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3498–3505, 2012.

[7] V. Balntas, E. Riba, D. Ponsa, and K. Mikolajczyk, "Learning local feature descriptors with triplets and shallow convolutional neural networks.," in *Bmvc*, vol. 1, p. 3, 2016.

[8] L. Van der Maaten and G. Hinton, "Visualizing data using t-sne.," *Journal of machine learning research*, vol. 9, no. 11, 2008.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, (Red Hook, NY, USA), p. 1097–1105, Curran Associates Inc., 2012.