HTSM Masterwork Report

LostPaw: Finding Lost Pets using Contrastive Learning with Visual Input

Robin Kock (r.kock@student.rug.nl)
Andrei Voinea (c.a.voinea@student.rug.nl)

Department of Artificial Intelligence, University of Groningen
9747 AG Groningen, The Netherlands

Supervisor: M.A. (Maruf) Dhali

Abstract

In this report, we present a contrastive neural network model that is able to differentiate between pictures of pets with high accuracy. The model was trained on a large dataset of images of various breeds of cats and dogs, and was evaluated using 3-fold cross validation. After being trained for 350 epochs, the model achieved an accuracy of 73.2% on the test data. The model did not suffer from overfitting, as the test accuracy closely followed the train accuracy. These results suggest that contrastive neural network models may be a promising tool for helping owners find lost pets. As such, the findings in this report can be used as the basis for a web application to help find lost pets. By using this model, users could upload images of their lost pets and receive alerts when a matching pet is detected in the application's database of images. This would allow pet owners to quickly and accurately locate their lost pets and reunite them with their families.
Keywords: contrastive learning, neural networks, object detection, transformers

1 Introduction

Losing a beloved pet can be a traumatic experience for their owners, who often go to great lengths to find them. These efforts can include posting flyers, searching online, and even hiring private investigators. Unfortunately, these methods are often unsuccessful, as it can be difficult to exhaustively search an area, especially when the pets take longer to be found. One of the challenges of finding lost pets is that they can travel long distances from their homes, especially if they become disoriented or are afraid. In many cases, pets that go missing are found a short distance from their homes, often within a few blocks or even just a few houses away. However, it is not uncommon for pets to travel much further, especially if they are chased by humans or other animals, or if they are attracted by the smells and sights of new environments. In some cases, pets have been known to travel dozens of miles from their homes, making it difficult for their owners to find them using traditional search methods.

As such, owners often rely on information from other people, which might have not seen the original request for help. Attempts to help in this scenario are often limited, as there are no unified channels that could be accessed both by volunteers and worried owners. In such cases, the use of artificial intelligence can be especially helpful, as it can analyse images of pets from any location and help to identify and reunite them with their owners.

Contrastive learning is a concept that has gained increasing attention in the field of artificial intelligence in recent years [1]. It refers to the process by which a machine learning model learns to differentiate between two or more different classes of input data by comparing them with each other. This approach has been shown to be particularly effective in tasks such as image classification, where the model is trained to distinguish between different objects or scenes based on their visual characteristics.

There are several key advantages to using contrastive learning methods. One of the primary benefits is that it allows for the efficient learning of high-dimensional representations of data, as the model is able to learn the relevant features for each class by comparing them to other classes. Additionally, contrastive learning can be used to learn representations that are invariant to certain types of transformations, such as rotation or scaling, which can be useful in tasks such as object recognition.

Using contrastive learning, a model could be trained on a large dataset of images of various pets. By comparing the characteristics of images, the model could learn to accurately distinguish between them and identify which ones are most likely to be the missing pet. This could potentially save pet owners time and effort in their search for their lost animal, and could also be a useful tool for animal shelters and other organizations that work with lost pets.

A contrastive neural network model uses advanced machine learning techniques to analyse and compare images of pets, making it possible to accurately distinguish between different breeds and individuals. This technology has the potential to revolutionize the way that lost pets are found, making it easier for owners to locate their missing companions and reunite with them. In this report, we will delve deeper into the technical details of such a contrastive neural network model and discuss its potential applications for solving the problem of searching a lost pet.

2 Theoretical Background

To create a contrastive learning model that is capable of differentiating between images of pets, several key technologies are required. At the heart of such a model is a neural network architecture that can learn a robust and effective representation of the data. In this study, we employed the Vision Transformer model as the foundation of our contrastive learning model. In addition, we used the Detection Transformer model to extract the pets from the images and the AutoAugment feature to augment the images. To optimize the model, we utilized a contrastive loss function, which enabled the model to learn the underlying structure of the data by contrasting similar and dissimilar examples. In the following sections, we provide a more in-depth examination of these technologies and their implementation in our contrastive learning model.

2.1 Transformer models

Transformer models are a type of neural network architecture that have been widely successful in a variety of natural language processing tasks, and have achieved state-of-the-art results on many benchmarks [2]. One key feature of transformers is the use of self-attention mechanisms, which allow the model to attend to different parts of the input data at different times while processing it. This allows the model to effectively capture long-range dependencies in the data, which is particularly useful for tasks such as language translation, where the meaning of a word can depend on the context in which it is used.

In addition to self-attention mechanisms, transformers also use multi-headed attention, which allows the model to attend to multiple parts of the input data simultaneously. This allows the model to more efficiently process the data and can improve its performance on tasks such as language translation. Overall, transformers have proven to be a powerful and effective neural network architecture for a wide range of natural language processing tasks, and have been applied to a variety of other tasks as well, including image classification and object detection.

2.2 Detection Transformer

The Detection Transformer (DETR) is a set-based object detector that utilizes a Transformer encoder-decoder architecture [3]. This architecture is designed to be an end-to-end object detector,

meaning that it is capable of performing both object classification and bounding box regression.

The model consists of a convolutional neural network backbone, which is used to extract features from the input image. These features are then flattened and supplemented with a positional encoding before being passed through a Transformer encoder. The Transformer encoder processes the features and generates a set of feature maps that represent the objects in the image.

The output of the Transformer encoder is then passed to a Transformer decoder, which takes as input a small number of fixed, learned positional embeddings called *object queries*. The Transformer decoder attends to the encoder output and generates a set of embeddings, one for each object query. Each embedding is passed through a shared feedforward network that predicts either a detection or a "no object" class. In the case of a detection, the model returns the class of the object (e.g., a cat, or a dog), and a bounding box that represents where the object is in the image.

2.3 Vision Transformer

Vision Transformer (ViT) is a neural network architecture designed to perform image classification tasks by processing raw pixel values as input, rather than using convolutional layers as in traditional image classification models [4]. ViT consists of a series of transformer blocks. Each transformer block contains a self-attention mechanism, which allows the network to process input tokens in a contextualized manner by weighting the importance of different tokens based on their relationships to other tokens.

In the context of ViT, the input tokens are 16 by 16 pixel patches of pixels from an image, and the transformer blocks are used to process these patches and extract features that are relevant for image classification. The patches are first embedded into a high-dimensional space using a learned linear transformation, and then passed through twelve transformer blocks. Each transformer block takes in a sequence of patches as input and produces a new sequence of patches as output.

As described in the study of [4], the output of the transformer blocks is then passed through a linear layer and a softmax function to produce the final class probabilities. ViT is trained using supervised learning, where the ground truth class labels are used to compute the cross-entropy loss and backpropagate the gradients through the model to update the weights. Therefore, it is possible to use a model pre-trained on a classification task to fine-tune ViT to a variety of tasks.

2.4 AutoAugment

AutoAugment [5] is a method for automating the process of data augmentation, which is the practice of applying various transformations to images in a dataset in order to artificially increase the size of the dataset and improve the robustness of machine learning models trained on the data. AutoAugment formulates the problem of finding the optimal data augmentation policy as a discrete search problem, and uses a search algorithm (implemented as a controller recurrent neural network) to sample a policy that specifies which image processing operations to use, the probability of using each operation in each batch, and the magnitude of the operations.

The search algorithm is trained using policy gradient methods, which allow the algorithm to update the controller based on the validation accuracy of a neural network trained with the fixed architecture and the sampled policy. The operations used by AutoAugment include functions that take an image as input and return an image that was altered. These operations include shear, translate, rotate, and various adjustments to contrast, colour, brightness, and sharpness.

Overall, AutoAugment is a powerful tool for finding effective data augmentation policies that can improve the performance of machine learning models, and has been shown to be particularly effective for image classification tasks.

2.5 Contrastive loss

Contrastive loss is a type of loss function that is commonly used in machine learning models designed for unsupervised learning [6]. It is based on the idea of contrastive learning, whose goal is to learn a representation of the data that captures its underlying structure, such as the relationships between different classes or the differences between similar and dissimilar examples.

To train a model using contrastive loss, the model is presented with a set of pairs of data points, along with a label indicating whether the two points in each pair are similar or dissimilar. The model is then trained to predict the correct label for each pair. During training, the model is optimized to minimize the contrastive loss, which is computed based on the predicted labels and the true labels.

Contrastive loss is often used in conjunction with a neural network architecture known as a Siamese network, which is a type of network that consists of two or more identical subnetworks that are trained to process different input data. The subnetworks are typically trained using the same weights, which allows them to learn a shared representation of the data. This shared representation is then used to compute the contrastive loss, which is used to update the weights of the network. The loss function is given in Equation 1, where d is the Euclidean distance between the two vectors representing each pet and m is the contrastive margin (a hyperparameter). The loss function in this approach serves to minimize the distance between feature vectors of the same class (i.e. the same pet) while simultaneously maximizing the distance between feature vectors of different classes (i.e. different pets), such that the distance between the feature vectors exceeds a certain margin m.

$$\mathcal{L} = \frac{1}{2}\,\mathrm{E}\left[\begin{cases}\max(m-d,0)^2 & \text{different pet}\\ d^2 & \text{same pet}\end{cases}\right] \tag{1}$$

3 Methods

In this study, we developed a contrastive neural network model to differentiate between pictures of pets. The model was trained on a large dataset of images of various breeds of dogs, and was evaluated on a held-out test set of images. The model was implemented using the PyTorch framework [7], and we used a combination of supervised learning techniques to train the model. In the following section, we will describe the dataset used for training and evaluation, the model architecture and training procedure, and the evaluation metrics used to assess the performance of the model.

3.1 Dataset

To create the dataset used in this study, we obtained images of pets from an adoption website named AdoptAPet [8]. Each image was fed through the DETR model, and the resulting bounding boxes of the pets were used to crop the pet from the image. For this study, we focused only on images of dogs, resulting in 31860 pets being stored with an average of 2.47 images per pet (78702 total images). The cropped images were then resized to fit a square of 384×384 pixels, and if the image was not wide or tall enough, the missing area was filled with black. Each image was then augmented twice using a pre-trained AutoAugment model, which followed the policies CIFAR10, IMAGENET and SVHN. A test set was created by putting aside images extracted from 3595 pets, with a total of 8854 images. The augmented dataset contained 236106 train images and 26562 test images, which were used to train and evaluate the contrastive neural network model developed in this study. For a schematic view of the data pipeline, see Figure 1.

To create the training dataset for the contrastive ViT model, we used a deterministic pseudo number generator that used the index of the pair as a seed value. This allowed us to generate a consistent set of training pairs for each epoch of training. We chose whether each pair was the same or different based on a probability of 50%. This value of 50% was chosen for the similarity probability in order to create an equal number of same and different pairs in the dataset used to train the contrastive ViT model. By selecting pairs to be the same or different with equal probability, we ensured that the model was exposed to a balanced set of training examples that were representative of the overall distribution of the data. This can help to prevent the model from becoming biased towards one type of example or the other, and can improve the model's generalization performance on the held-out test set.

The dataset returned a pair of images and a label, which was either "same" or "different". If the label was "same," the pair of images was made up of two different images of the same pet.

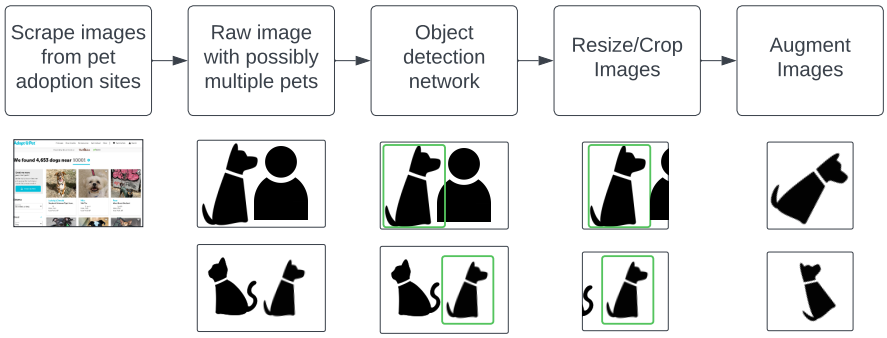


Figure 1: Data collection process. Top: individual steps. Below: example images.

These images were selected from the training set described earlier, where the chosen sample was either the original image, or an augmented version of it. If the label was "different," we randomly selected two different pets from the dataset (using the same seed described earlier), and an image was selected for each pet. These images were then included in the pair, and this process was repeated to create many training pairs for the contrastive ViT model. An example of pet pairs, along with their labels, is displayed in Figure 2.

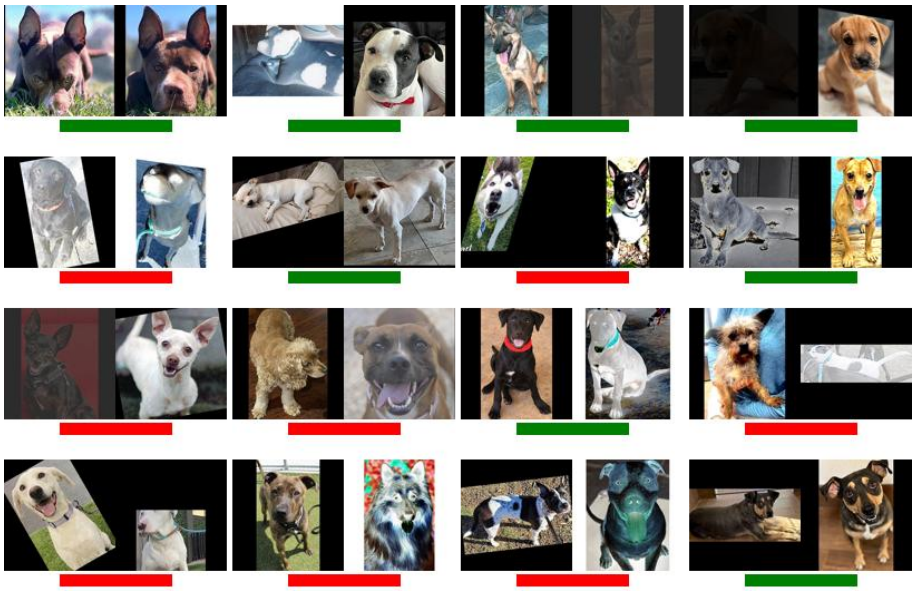


Figure 2: Example data pairs (green: same pet, red: different pet).

3.2 Model architecture

To develop the contrastive ViT model, we used ViT as the backbone of the model [9]. The output of ViT was flattened, and we appended three fully connected layers to the end of the model, which reduced the size of the output to a desired latent vector size. The last three layers of the contrastive ViT model consisted of two hidden layers and a final output layer. The hidden layers contained twice the number of neurons as the size of the latent space, while the final output layer contained the same number of neurons as the size of the latent space. These layers were used to transform the output of the ViT backbone into a compact representation of the data that captured the underlying structure of the image data. The ELU activation function was used between the final layers to allow the propagation of negative values, which might become available during the use of the contrastive loss function. For a graphical overview of the model architecture, see Figure 3.

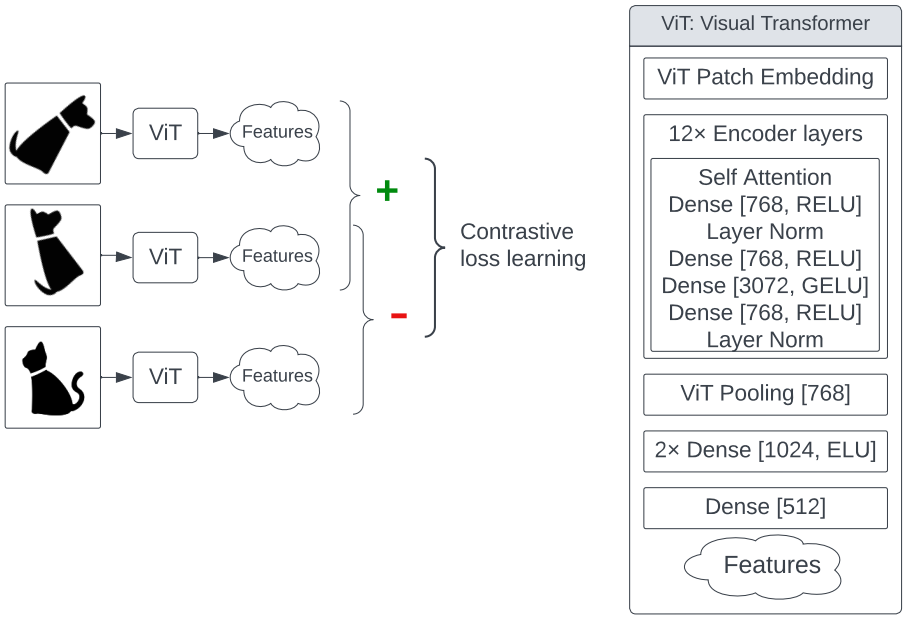


Figure 3: Architecture of the Contrastive Vision Transformer model.

The hyperparameters for the contrastive ViT model were chosen through an empirical process of parameter sweeps. This involved training the model with a range of different values for each hyperparameter, and then evaluating the performance of the model on a validation set. By comparing the results of the different runs, we were able to identify the values of the hyperparameters that resulted in the best performance. This process allowed us to fine-tune the model and ensure that it was able to achieve the best possible performance on the task of differentiating between pictures of pets. A list of these hyperparameters can be found in Table 1.

Hyperparameter	Value
Epochs	350
Latent Space Size	512
Batch Size	8
Batch Count per Epoch	128
Test Batch Size	8
Test Batch Count	128
Optimizer	AdamW
Learning Rate	5.0e-5
Weight Decay	2.0e-4
Contrastive Margin	1.66

Table 1: Hyperparameters chosen for the contrastive ViT model.

3.3 Evaluation metrics

To evaluate the performance of the contrastive learning model, we used k-fold cross validation. This is a standard method for evaluating machine learning models that helps to mitigate the risk of overfitting. In k-fold cross validation, the training data is divided into k subsets, and the model is trained k times, each time using k-1 subsets for training and the leftover subset for testing. This allows the model to be evaluated on a variety of different test sets, which helps to provide a more robust estimate of its generalization performance.

In our study, we used k=3 for our cross validation, which resulted in 3 different models being trained and evaluated. For each fold, we trained the model for a fixed number of epochs and used

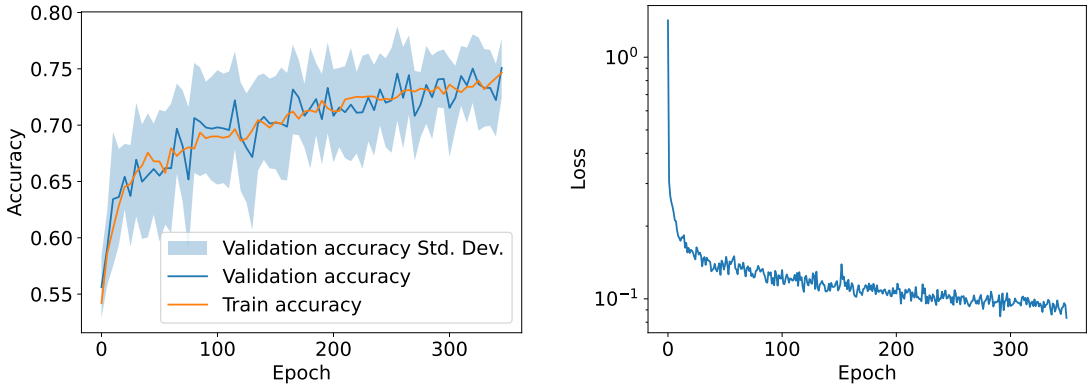
the validation set to tune the model's hyperparameters. Once the model was trained, we evaluated it on the test set and recorded its performance in terms of accuracy, as well as the type I and type II error rates. The type I error rate is the fraction of times that the model predicts that two pets are the same, when in fact they are different. The type II error rate is the fraction of times that the model predicts that two pets are different, when in fact they are the same. By examining these error rates, we were able to get a more detailed understanding of the model's performance.

After evaluating the model on each of the three folds, we averaged the test metrics to obtain a final score that indicated whether our method is overfitting to the provided data or not. This allowed us to get a more reliable estimate of the model's generalization performance.

4 Results

As described earlier, we used 3-fold cross validation to evaluate the performance of our contrastive learning model. Similarly, a held-out test set was used to assess the generalizability of the model. Over the course of 350 epochs, the model achieved on average an accuracy of approximately 74.5% on the validation set. The model was trained on a large dataset and did not appear to be overfitting, as the validation accuracy closely followed the train accuracy (see Figure 4a).

When examining the error rates of the model (see Figure 5), we observed that the model initially classified every pair of pet images as the same pet. However, over the course of training, the model learned to differentiate between different pets, and the type 1 error rate decreased. The type 2 error rate was very close to zero for most of the training. These results suggest that the model was able to learn a robust and relatively effective representation of the data, which was capable of distinguishing between different pets.



(a) Accuracy

(b) Loss

Figure 4: Mean train accuracy and loss of the contrastive ViT model, averaged over three model runs. The data for accuracy was smoothed by averaging the values every 5 epochs.

In addition to the accuracy results, we further examined the loss values of the model during training (see Figure 4b). We observed that the loss value steadily dropped from a starting value of approximately 1.6 to a final value of approximately 0.09. This trend is generally indicative of the model learning a better and better representation of the data over the course of training. The low loss value suggests that the model was able to learn an effective representation of the data, which could potentially allow the model to make accurate decisions on unseen samples.

When investigating the results of the models on the held-out test set, we observed that the mean accuracy was 73.19% (SD=0.16%). Similarly, the mean type 1 error rate was 26.2%, and the type 2 error rate was 0.44%. These results follow the values of the metrics on the train and validation sets closely, which indicates that the model can generalize to external data.

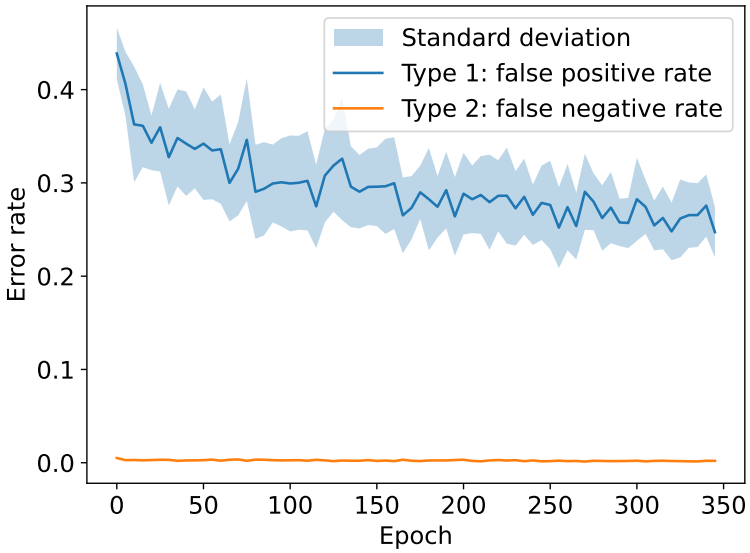


Figure 5: Type I and II error rates of the model on the test set at every epoch. The data for the error rates was smoothed as described previously.

5 Discussion

In this study, we trained a contrastive neural network model to differentiate between pictures of dogs and evaluated its performance on a held-out test set. The results of the 3-fold cross validation have shown that the contrastive learning model is able to differentiate between pictures of pets with an acceptable accuracy, and that it is able to generalize to unseen data. In the following section, we will discuss the results of the evaluation in more detail, as well as the potential implications of these results for the use of artificial intelligence in the search for lost pets.

One issue with the results is the high false positive rate. While this might initially seem like a negative aspect of the model's performance, it could actually be beneficial in the context of finding lost pets. If there are only a relatively small number of missing pets in a given area at any given time, then a high rate of false positives may not be a significant issue, as these can easily be dismissed.

Another issue to consider is the use of the AutoAugment feature, which sometimes inverts the colour of the pet images. While this could potentially skew the training and testing accuracy, it could also lead to better generalization. By introducing variations in the colour of the images, the model may be able to learn more robust features that are less sensitive to changes in the appearance of the images. This could help the model to perform better on real-world data that may contain variations in colour and lighting conditions.

One potential direction for future research would be to expand the network to include other types of pets. This could be done by first using DETR to identify which pet is present in the image (e.g., a cat or a dog). Once the image has been identified as containing a certain pet, it could be passed through a separate fine-tuned model that is specialized in comparing pets within each class. This approach would allow the network to take advantage of the strengths of both the DETR and ViT models, and could lead to a more robust model due to having more contrastive data available.

Model demonstration

To make the contrastive learning model available to a wider audience, we developed a web application that allows users to upload pictures of dogs and discover whether there are any similar dogs found in the system. The web application is built using a combination of HTML, CSS, and JavaScript, and is designed to be easy to use and accessible from any device with a web browser.

To use the web application, users simply need to visit the website and follow the prompts to upload an image of a pet. Once the image has been uploaded, the web application processes the image using the contrastive learning model and returns a list of pets along with their similarity score.

In addition to providing predictions about the similarity of pets, future implementations of such an application can restrict the area of search to a certain area. Similarly, users could opt to search pets based on their breed, size or other criteria. Such features can be useful both for users who are looking to adopt a pet or who are trying to find a lost pet. Overall, the web application is a convenient and easy-to-use tool that leverages the power of contrastive learning to help users identify and find pets.

References

[1] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, "A Simple Framework for Contrastive Learning of Visual Representations," June 2020.

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Dec. 2017.

[3] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-End Object Detection with Transformers," May 2020.

[4] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," June 2021.

[5] E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le, "AutoAugment: Learning Augmentation Policies from Data," Apr. 2019.

[6] 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)*, vol. 2, pp. 1735–1742, June 2006.

[7] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in PyTorch," 2017.

[8] "AdoptAPet: Search for local pets in need of a home." https://www.adoptapet.com/.

[9] B. Wu, C. Xu, X. Dai, A. Wan, P. Zhang, Z. Yan, M. Tomizuka, J. Gonzalez, K. Keutzer, and P. Vajda, "Visual Transformers: Token-based Image Representation and Processing for Computer Vision," 2020.