**CHECKING FOR SIMILARITY OF TWO ANIMAL SPECIES USING A SIAMESE NEURAL NETWORK**

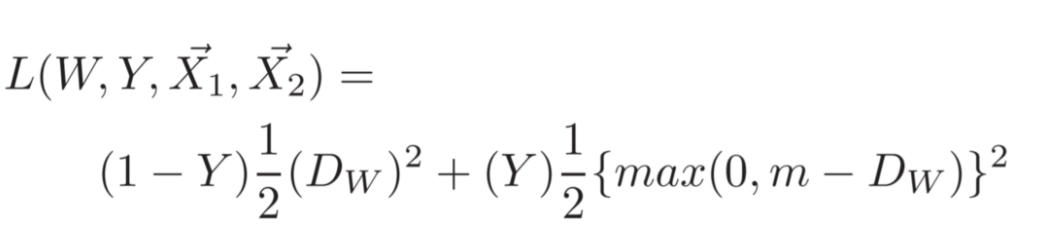
**ABSTRACT**

The aim of this project is to make a system that can identify whether two breeds of animals belong to the same species or not. This idea can be extended to further identify the species of a new unseen animal by comparing it against the already existing animals in the database. This is achieved using a Siamese Neural Network. A Siamese neural network is defined to be a neural network that uses the same weights to compute the output for 2 input vectors. The output is a similarity score that indicates the relationship between the two inputs. In this project I use a Siamese neural network to identify whether two breeds of animals belong to the same species. A Siamese neural network is different from a standard classifier as it is trained to learn a similarity function. The advantage is that training needs to be done only once. After training the network learns the necessary features to distinguish between 2 or more animals. So, to predict the species of a new animal, the model needs to just compare the input image with all the existing images in the database. Moreover, if the image does not match with any of the images in the training database, then it is possible that a new species of animal has been discovered. This is often referred to as one shot learning. The advantage of one shot learning is that the requirement of a large training set is drastically reduced and there is no need for training again and again. This technique has widespread applications in various other domains such as facial recognition, signature verification and domains of natural language processing such as duplicate question detection.

**INTRODUCTION**

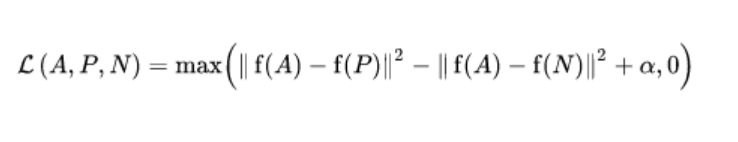
A Siamese network contains 2 or more identical sub-networks sharing the same set of weights to produce comparable outputs. The main idea behind Siamese networks is that they can learn useful features that can be used to distinguish between two or more inputs easily. They have widespread applications in various domains. For instance in facial recognition for attendance in a company, it is not possible to train a neural network every time a new employee joins the company. This will be really cumbersome and time consuming. So instead we train a neural network that can compare 2 images (here faces) and produce a similarity (or dissimilarity in our project) and judge whether the two are identical or not. This neural network consists of 2 identical neural networks and needs to be trained only once. After it has been trained to learn the similarity function, then whenever a new employee joins the company, only 1 picture of that employee will be sufficient and will serve as a baseline for the attendance of that particular employee. This idea has been extended in our project to compare whether two animals belong to the same species or not. The Siamese neural network is trained to learn a similarity function that can be used to distinguish between two images of animals. The Siamese network can be trained using different losses. Some of them are as follows: -

1. Contrastive loss: - This is the loss function that has been used in this project for training.



Here Y represents the label (0 for similar class and 1 for dissimilar class), Dw represents the Euclidean distance between the two outputs corresponding to the two inputs X1 and X2. m is the margin term. It comes into consideration when Y = 1, i.e dissimilar pairs are being considered. If Y = 1 then the pairs will not contribute to the loss only if the model predicts them to be dissimilar by a margin greater than m. This helps the model to become much more robust and is an important hyperparameter. If Y = 0, then the dissimilarity (distance) between the outputs is directly used for loss computation.

1. Triplet loss: - This type of loss involves 3 images. One of them is the baseline and is referred to as the anchor image. It is compared with a similar image (positive) and dissimilar image (negative). The distance between the anchor and positive example is minimized and the distance between the anchor and negative example is maximized using the loss function shown below: -



Here α represents the margin and is a hyperparameter that needs to be tuned. The importance of the margin term is that it encourages the images from different categories to be as far apart from each other as possible i.e it helps to distinguish between the images in a much better way. However the images should be chosen carefully while training. If the images are chosen randomly then the above equation is satisfied really easily and as a result the performance may not be up to the mark. So we should select triplets that are hard to train on so that the model ends up learning the decision boundaries pretty well.

Siamese networks are a very powerful tool for dimensionality reduction as they map inputs to a lower dimensional space. It serves as an important feature extractor as it learns really important features that can help distinguish between different classes of images really easily. The neural network can be a deep artificial neural network or a convolutional neural network (like in this case) or even recurrent neural networks. We use convolutional neural networks because of their extremely good performance on image data. Another advantage associated with Siamese Neural networks is that they are more robust to class imbalance. That is, if there are a few training examples of one particular species, then due to the power of one shot learning any new image of the same species will still be categorised correctly. However one downside associated with these networks is that they don’t output class probabilities, rather they output the distance from each class and the training is slightly cumbersome due to pairs being involved at each step rather than individual instances.

**CONTRIBUTION**

There were several steps involved in the training process. First of all there was a need to find the right dataset. There were several datasets that consisted of either 1 or 2 species of animals such as cats and dogs. Finally I chose the ‘Animals 10’ dataset available on Kaggle that consists of over 28000 images belonging to 10 different classes. They are as follows: - Dog (Cane), Horse (Cavallo), Elephant (Elefante), Butterfly (Farfalla), Chicken (Gallina), Cat (Gatto), Cow (Mucca), Sheep (Pecora), Squirrel (Scoiattolo), Spider (Ragno). Out of these 10 classes I made use of only 6 classes as my project concentrated exclusively on animals. Thus the species used were:- Dog, Horse, Elephant, Cat, Cow and Sheep. The dataset for training was prepared using the following steps:-

1. First the dataset was downloaded onto the Colab notebook by using the unique API token (Kaggle.json file) associated with each Kaggle account.
2. After the dataset was downloaded only the 6 selected species were chosen for further use.
3. An image was selected at random. Then a number was generated from the set [0,1]. Depending on the number chosen a new image was selected from either the same class or a different class. Thus a dataset of 20000 pairs with approximately 50% similar and 50% dissimilar images was built.

Before feeding each image to the model, the images were re-sized to (224,224,3). This is because we are using a pre-trained model that was defined on RGB images of height and width equal to 224 each.

The next step is to define the model. A ResNet-152 model has been pre-trained on the ImageNet dataset. The last layer was removed and instead 2 linear (fully connected) layers were added. The output of the neural network is a 5 dimensional vector. Pre-trained models are specifically chosen to reduce the training time and achieve significantly better results. This was observed when I tried to train the full neural network and the performance was not up to the mark even after several iterations.

After the model is defined for each iteration B pairs of images were taken where B is the batch size (128 in my case) and the output was generated for each of them. The 5 dimensional vectors were then compared using the Euclidean Distance metric. The dimension of the output can be varied depending on the number of features desired for each animal. This distance was later utilised while calculating the contrastive loss. The contrastive loss was defined as described above. After the loss was calculated, the errors were backpropagated and the gradients were computed. After the gradient computation, the trainable weights were updated according to the defined learning rate (0.0001 in this case). Other values of learning rate were tried but this gave the best results. Significantly higher values even lead to divergence of cost function. Another hyperparameter that required significant tuning was the margin of the contrastive loss. As discussed before a small value of the margin gave really bad results. After testing several values of the margin, a value of 2 was chosen. Adam’s optimizer was chosen thought the results didn’t vary much with RMSProp or SGD with momentum.

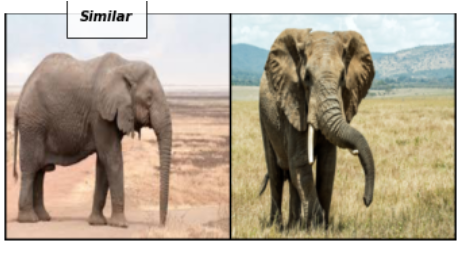
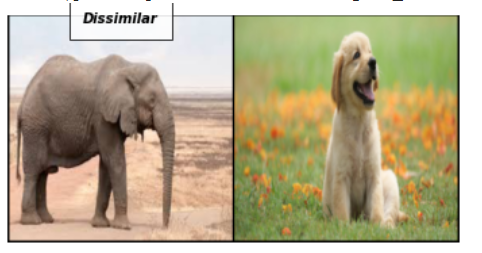
Then the model was trained for 5 epochs. 5 epochs were sufficient as a pre-trained model was being used. The final training loss obtained was 1.1 units.

After the training was completed it was time for testing. To prepare the testing dataset I downloaded several images from Google Images of animals belonging to both the training dataset as well as images of those species which were not included in the training dataset. The images were then uploaded as a separate folder in the same directory where training images are stored. The list of species in the test dataset that were also present in the training dataset were: - Dog and Elephant. The list of species that were not present in the training dataset were: - Tiger, Leopard, Giraffe, Monkey and Camel. To gauge the model’s performance I even included non-animals such as birds, cockroaches and lizards. After preparing my test dataset the images went through an identical transformation like the training images, i.e they were convert to RGB images of size 224\*224.

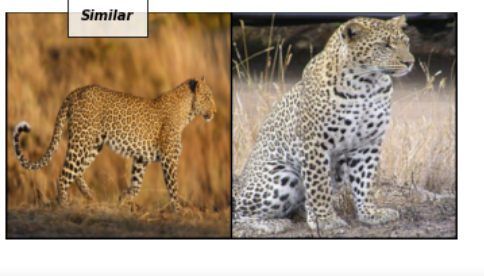
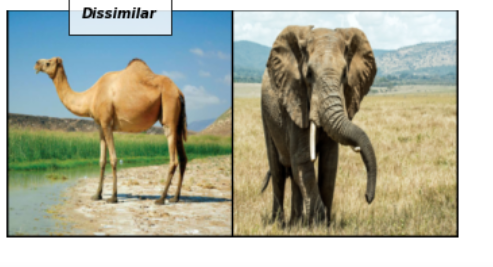
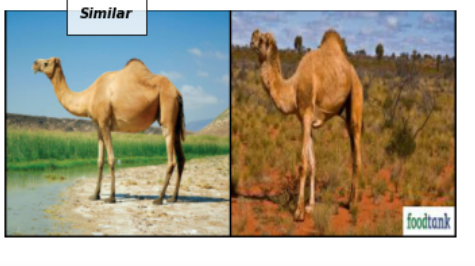
After this pairs of images were used each time and were passed through the Siamese network. The resultant pair of 5 dimensional vectors were then compared using their Euclidean distance. This distance was then used to decide whether the 2 images are Similar or Dissimilar based on a certain threshold. I chose a threshold of 0.035 as it was giving really good results. A value ranging from 0.03-0.05 can be chosen for decent results. Thus for images of the same species this distance was coming out to be less than 0.035 most of the time and vice versa for images of different species. Note that the appropriate threshold was observed to be varying with every training process. Hence the performance may vary slightly each time depending on the convergence of the loss.

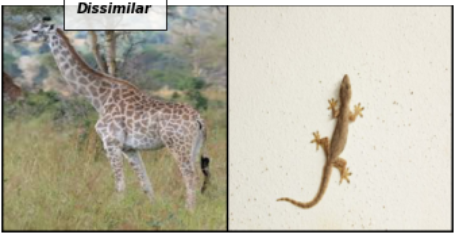
**RESULTS**

First the images belonging to the training set were tested. Two different species of dogs were compared and then two different images of elephants were compared. Later on an image of a dog was compared with that of an elephant. All the 3 outputs have been shown below: -



Then we tested the performance of the model for unseen animals and also checked if the model was able to distinguish an animal from species like cockroaches, lizards etc. More screenshots of the results have been added in the ‘results\_screenshots’ folder present along with the report.

**CONCLUSION AND FUTURE WORK**

The model achieves significantly good results in distinguishing animals of different families but faces some difficulty for animals belonging to the same family. Like here leopard and tiger both belong to the Felidae(Cat) family and are classified as Similar even though they belong to different species.



The work here can be extended to animals belonging to the same family exclusively. For instance, a Siamese network can be trained for animals belonging to the Felidae family. It can also be extended for different breeds of the same species like dogs. Not only animals but this work can be carried out for other living species such as plants, insects, birds and even non-living things.

Also, there were a few cases of misclassification in case of images where the animal occupied a very small part of the image (background more dominant) or the size of the animal is very small (animal not clear in general).

Here my work solely focussed on classifying whether two images are similar. But this concept can be extended to identify the species of a new animal. Such a work would require a very large dataset and training for longer durations. But once training is done, then this model can achieve excellent results. If we want to identify the species of a new animal, it can be compared with the output generated for all images in the training dataset. Moreover if a new species of animal comes up and the results show that it’s dissimilar with all the existing images in the database, then there is a possibility that a new species of animal has been discovered. Also if more research needs to be carried out on this newly discovered species, then outputs corresponding to new images of this species can be compared with the old output directly without any requirement of re-training as Siamese networks require just one instance to correctly classify any new input.

The 5 dimensional output vectors can be interpreted as embeddings for that particular image. So if we group all the images belonging to the same species we can calculate the average of the outputs to get a resultant 5 dimensional embedding that is representative of that particular species. So we can also check for the similarity between two particular species by measuring the distance between these resultant embeddings. If the model is properly trained then these embeddings might be really close for species belonging to the same genus, than for species belonging to the same family but different genus and really distant for species belonging to different kingdoms (if images across different kingdoms are included in the training dataset).

**REFERENCES**

1. Siamese Neural Networks for One-shot Image Recognition – Gregory Koch, Richard Zomel and Ruslan Salakhutdinov - https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf
2. Dimensionality reduction by learning an invariant mapping- Raia Hadsell, Sumit Chopra and Yann LeCun- http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf