Reverse Image Search

Rishabh Monga and Ajinkya Chavan
School of Informatics and Computing, Indiana University, Bloomington

I. Introduction

The objective of this project is to showcase the how various predictive algorithms perform the task of Image Similarity. Our inspiration of the project comes from Googles Image Search (https://images.google.com/), where a user can provide a picture to the website and it in-turn provides similar images to the said image. The similarity in this case can be described to be various things such as objects (object detection), person (face detection), scene (location recognition) etc. In our project, we only attempt to find images based on standard pixel or structural similarity without attempting to recognize the objects in the image.

II. Dataset Description

To begin let us understand the datasets used for training and testing the prediction models of this project. We began with a very well-known CIFAR-10 dataset [1]. This dataset consists of 60,000 images, that are of 32 x 32 in dimension. The images belong to 10 different classes (hence the 10 in the name) which are namely; airplane, automobile, bird, cat, deer, dog, frog, horse, ship and truck. It can be observed here that the classes are completely mutually exclusive. This ensures a fair amount of reduction in miss-classification and confusion for the predictive algorithm when making predictions. This dataset was intended to be used to set a baseline accuracy which we could then use to judge and compare our deep learning models to.
The second dataset that we used was the IMDB-WIKI 500k dataset [2]. This dataset consists of about 500,000 images, that are of 300 x 300 in dimension. The images are of celebrities and are fully labelled, with a meta information file describing the name, gender, age and face location in the image. This dataset was primarily used for training our deep learning algorithms for estimating the similarity between the images and for extracting embedding vectors used for making prediction using various other models.

III. Prediction Models

A. pHash

Perceptual hashing is the use of an algorithm that produces a snippet or fingerprint of various forms of multimedia.[3] If you give a file as an input to a pHash, the output will be 64 bit fingerprint of the file which will be unique to that file. We took a difference of files' pHash and lower the difference, similar the data. We have used 1000 samples for quick results. Change line 65 and 70 to test for different subset.

Run the program as: "**python3.5 CV_Project_pHash.py**"

Accuracy = 10.29%

B. Support Vector Machine

SVM is an optimization algorithm which uses a hyperplane to seperate data in linear or non-linear space. Training SVM on the data took around 3 hours for full dataset. Change the range from line 95 to 103 to test for different subsets of dataset

Run the program[5] as: "**python3.5 CV_Project_SVM.py**"

Accuracy = 3%

C. SIFT

SIFT is a Computer Vision algorithms used to detect and describe local features in images. In our case, we took the key points from two images and applied distance formula to it. The closer the distance, similar the images. Change line 83 and 88 for different subsets.

Requires opencv version OpenCV-3.0.0 and above [4]
Run the program as: "**python3.5 CV_Project_SIFT.py**"

Accuracy = 33.64%

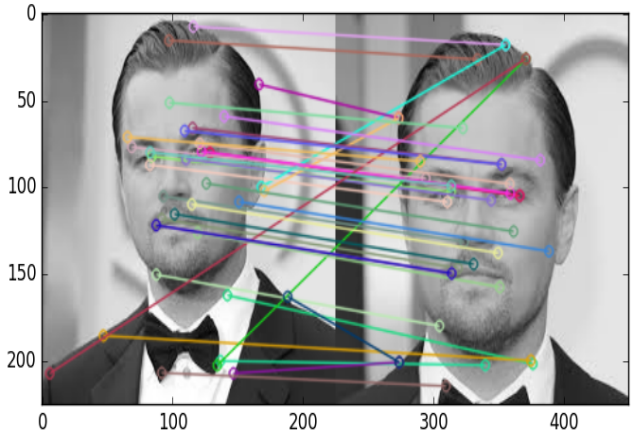

Fig.1 - SIFT

IV. Convolutional Neural Networks

Convolutional Neural Network Our CNN uses a standard architecture, with 4 convolutional and activation layers, two maxpooling layers. The two fully connected layers provide us with the embedding vector used to compare images and provide a similarity score. The final SoftMax layer is used to provide a prediction of the class to which the image might belong. The architecture is depicted below:

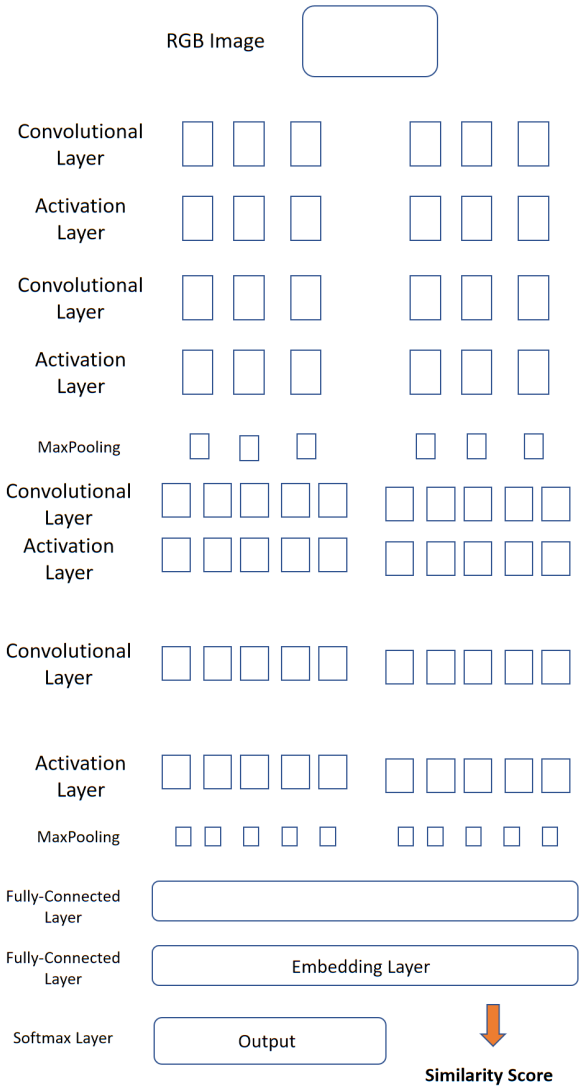

Fig.1 - Convolutional Neural Network

We started with this architecture that was pretrained on CIFAR-10 dataset for image classification. The CNNs are then finetune on our IMDB-WIKI dataset.
The IMDB 500k dataset used here was preprocessed and cropped so as to only contain the portion of the image that was most significant in the image. The training and test datasets were created by randomly choosing 200,000 images in the training dataset and 50,000 images in the training dataset. Each epoch, containing 2000 steps took 1.5 hrs to run per epoch.

V. Image Similarity Prediction

The prediction for similarity score is done by two methods:

A. Embedding vectors

The input image is first preprocessed in order to ensure that the CNN is only provided with the most relevant part of the image so as to maximize feature recognition and reduce confusion for the neural network. A SIFT feature extractor is employed to identify the most interesting parts of the image. Once this is done we crop close to the most relevant features and then use the cropped image as our input image for prediction. The CNN receives this image and then generates an embedding vector at the penultimate layer (final fully connected layer). We extract this vector and compare it to the embedding vectors of our repository images. A cosine similarity is then calculated giving us a score of how close this image is from the repository images. Sorting the repository images in the ascending order of the dot product of their embedding vector with the input image provides us with the order in which these images are like the input image.

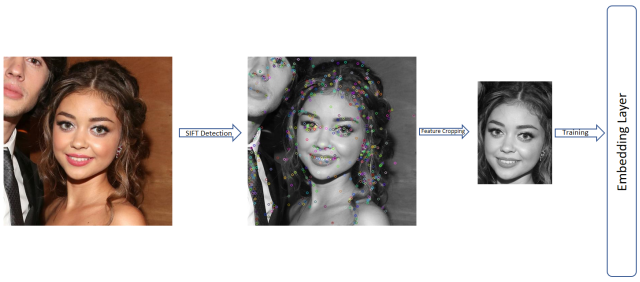

Fig.2 - Embedding

The embedding extraction is where we see the most appreciable increase in the prediction accuracy. It was observed that the accuracy ranged between 67-70% when predicting if the two images were similar.

VI. Neural Network Classification

Like the embedding extraction, another way to classify these images is to train the output SoftMax layer to classify based on the ordinal classes created in the test and training datasets using the celebrity nomenclature. The fine-tuned CNN, trained for 200 epochs, would be able to classify the image and make prediction on which of the 10 celebrities does the cropped input image belong to.

The images are accompanied by the confidence score with which the model could predict the similarity between the images. As evident the CNN was able to boost the accuracy observed using embedding extraction to almost 74%.

Fig.3 - Neural Network Classification

VII. Experiments

Further expanding our experiments, we used a pre-trained Inception-V3[6] architecture neural network which is provided, with pre-trained weights on the ImageNet dataset, in the Keras[7] package and compared it to an Inception-V3 architecture neural network that we trained for 100 epochs. Since these are the most cutting-edge models for image similarity detection a higher accuracy was expected and observed. However, since our neural network was trained on the dataset being tested itself. The numbers favored our model. Below are the tensor board results for our model showing approximately 82% accuracy after 120 epochs.

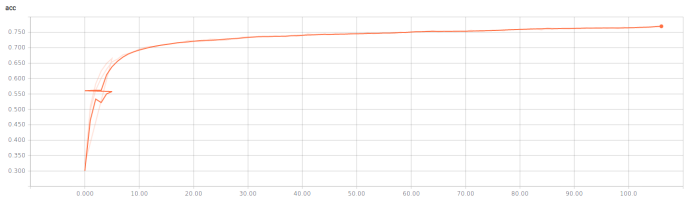

Fig.4 - Accuracy

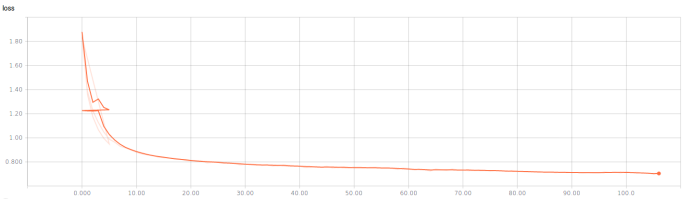

Fig.5 - Loss

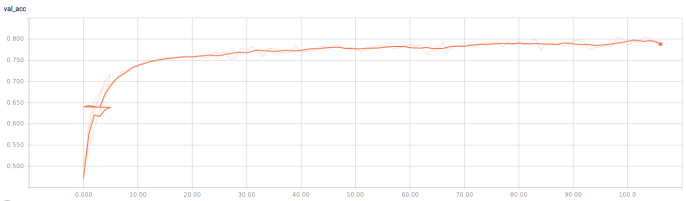

Fig.6 - Validation accuracy

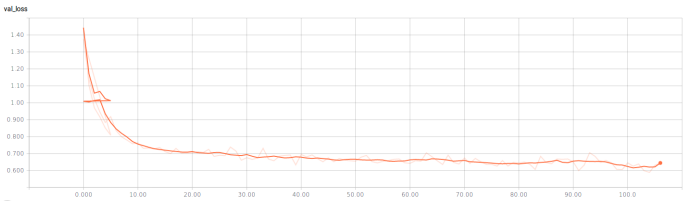

Fig.7 - Validation Loss

Please note that these networks were trained for almost 100 classes differentiated on celebrity names.

VIII. Discussion

Our journey of through various prediction models starting from supervised to deep neural networks conducted on the two datasets were a real eye-opener. We observed that although image similarity is thought to be a solved problem ever since the GoogleNet architecture was introduced, simple models such as clustering and a well-trained CNN can come close to the accuracy numbers produced by even the most advanced models if the dataset is known and the problem is well defined. For our future experiments, we would like to expose our models to out of sample images to see how they fair with inputs that the models have never seen before. Also, we would like to include a capability of the project to allow users to decide which part of the image to be considered relevant rather than using SIFT as the deciding factor. I would also like to investigate how well our model would have performed if we have allowed it to train for another 200 or 250 epochs. I believe toying around with the dropout rate and belief propagation would have provided us with far better numbers had we have the time and resources for this.

References

[1] https://www.cs.toronto.edu/ kriz/cifar.html
[2] https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/
[3] http://www.fmwconcepts.com/misc_tests/perceptual_hash_test_results_510/
[4] http://docs.opencv.org/trunk/d5/d3c/classcv_1_1xfeatures 2d_1_1SIFT.html
[5] http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html
[6] https://github.com/tensorflow/models/tree/master/inception
[7] https://keras.io/