Project Report

Deep Learning [CSE4007]

Similarity Index

By

Ansh Poonia
200C2030159

Jitesh Kumar Yadav
200C2030158

Department of Computer Science and Engineering
School of Engineering and Technology
BML Munjal University

April 2023

Declaration by the Candidates

We hereby declare that the project entitled "***Similarity Index***" has been carried out for fulfilling the partial requirements for completion of the core-elective course on **Deep Learning** offered at the 6th Semester of the Bachelor of Technology (B.Tech) program in the Department of Computer Science and Engineering during AY-2022-23 (even semester). This experimental work has been carried out by us and submitted to the course instructor *Dr. Soharab Hossain Shaikh***.** Due acknowledgments have been made in the text of the project to all other materials used. This project has been prepared in full compliance with the requirements and constraints of the prescribed curriculum.

Ansh Poonia

Jitesh Kumar Yadav

Place: BML Munjal University

Date: 23 April 2023

Contents

	Page No.
Introduction	4
Problem Statement	5
Literature Review	6-10
Description of the Dataset	11
Methodology	12-14
Experimental Results	15-16
Conclusions	17
References	18
Appendix	19-23

Introduction

With the explosive growth of digital images available online, the ability to automatically compare and recognize images has become increasingly important in a variety of fields, including computer vision, image analysis, and machine learning. One of the most popular approaches to image recognition is Convolutional Neural Networks (CNNs), which have demonstrated excellent performance on a wide range of visual recognition tasks.

In this report, we present a project that focuses on finding the similarity between any two images based on the feature map extracted from a pre-trained CNN. The project aims to explore the effectiveness of using CNNs in image similarity tasks and the potential applications of this technique in various fields.

We use a pre-trained CNN, specifically MobileNetV2, to extract the feature map of each input image. We chose MobileNetV2 for faster calculation of feature map for images. The feature maps are then compared using a mean squared error to obtain a similarity score between the two images for a given feature layer. We evaluate the performance of our approach using a small subset of animals' image dataset.

The remainder of this report is organized as follows. In Section 2, we described problem statement in detail. In Section 3, we discuss the related work in image similarity tasks and CNN-based feature extraction. In Section 4, we describe our approach in detail, including the data pre-processing, feature extraction, and similarity measurement. In Section 5, we present the experimental results and performance evaluation. Finally, we conclude the report with a summary of our findings and the potential implications and future directions of this project.

Problem Statement

Convolutional Neural Networks (CNNs) have been widely used in various computer vision tasks, such as object recognition, image classification, and segmentation. However, training a CNN from scratch on a small dataset can be challenging due to limited data availability, which often leads to overfitting and poor generalization performance.

Our goal is to develop a system which is capable of providing us with a reliable similarity score between any two images, using only the limited data available. This system can be further paired up with a probabilistic theorem like Dempster Shafer Theory, etc., for classification tasks, or can be used as web scrapper to collect similar images, etc.

Literature Review

[1] "Learning Deep Image Similarity with Spatial Feature Fusion and Semantic Supervision"

Jianan Li, et al., proposes a deep learning approach for image similarity. The proposed approach combines spatial feature fusion and semantic supervision to achieve state-of-the-art performance on several benchmark datasets.

The proposed approach is built on top of a convolutional neural network (CNN) architecture, which is trained end-to-end to learn image representations that capture both spatial and semantic information. Specifically, the authors introduce a spatial feature fusion module that learns to combine global and local features of the input images. The global features are obtained by applying a global average pooling operation to the output of the last convolutional layer of the CNN, while the local features are obtained by applying a spatial pyramid pooling operation to different regions of the image.

In addition to the spatial feature fusion module, the proposed approach also includes a semantic supervision module that encourages the CNN to learn discriminative image representations that are semantically meaningful. This is achieved by introducing a semantic loss function that penalizes the model when it fails to distinguish between images with different semantic labels. To evaluate the proposed approach, the authors conducted experiments on several benchmark datasets, including CUB-200-2011, Cars196, and SOP. The results show that the proposed approach achieves state-of-the-art performance on all datasets, outperforming several existing approaches by a large margin.

The results of the experiments demonstrate that the proposed approach achieves state-of-the-art performance on several benchmark datasets, which suggests that it has the potential to be a valuable tool for various computer vision applications, such as content-based image retrieval, object recognition, and image classification.

[2] "Improved Siamese Network for Similarity Learning"

Yang Liu, et al. proposes an improved version of the Siamese network for image similarity learning. The Siamese network is a popular neural network architecture for similarity learning that uses two identical sub-networks to process two input images and produces a similarity score as output.

The proposed improved Siamese network (ISN) architecture incorporates residual connections and attention mechanisms to improve the network's performance. Residual connections allow the network to learn the residual mapping between the input and output, which can improve the training of deep networks. The attention mechanism is used to weight the importance of different features and improve the discriminability of the learned representations.

The authors conduct extensive experiments on several benchmark datasets, including CIFAR-10, CIFAR-100, and MNIST, to evaluate the performance of the ISN architecture. The results show that the ISN architecture outperforms several state-of-the-art Siamese network architectures in terms of accuracy and convergence speed. For instance, on the CIFAR-10 dataset, the ISN architecture achieved an accuracy of 98.66%, which is higher than the accuracy of other Siamese network architectures, such as the standard Siamese network and the Siamese network with residual connections. The authors also conduct ablation studies to analyse the contribution of each component of the proposed architecture. The results show that the attention mechanism significantly improves the performance of the network, while the residual connections have a smaller impact.

The proposed ISN architecture is a promising approach for image similarity learning that achieves state-of-the-art performance on benchmark datasets. The incorporation of residual connections and attention mechanisms provides a more effective way to learn representations for image similarity, which can be useful in various computer vision applications, such as image retrieval and object recognition.

[3] "FASNet: A Fast and Accurate Siamese Network for Image Similarity"

The paper proposes a novel Siamese network architecture, called FASNet, for fast and accurate image similarity. The proposed FASNet architecture leverages a multi-scale feature fusion strategy and a lightweight backbone network to achieve both high accuracy and fast computation. Specifically, FASNet consists of two branches, each of which contains a set of convolutional layers followed by max-pooling layers to extract multi-scale features. The two branches are then fused using a feature fusion block, which combines the high-level and low-level features extracted from each branch.

Moreover, FASNet incorporates a lightweight backbone network, called Mobilenetv2, as the main feature extractor, which significantly reduces the computation cost while maintaining high accuracy. Additionally, FASNet employs a triplet loss function to optimize the network parameters, which encourages the network to learn discriminative image representations that can accurately measure image similarity. The proposed FASNet architecture is evaluated on several benchmark datasets for image similarity, including Fashion-MNIST, CIFAR-10, and CIFAR-100. The experimental results demonstrate that FASNet outperforms several state-of-the-art image similarity methods in terms of accuracy while achieving faster computation. For instance, on the Fashion-MNIST dataset, FASNet achieves an accuracy of 99.2% and a computation time of only 1.6ms per pair of images, which is significantly faster than other methods.

Overall, "FASNet" presents a promising approach for fast and accurate image similarity using a lightweight Siamese network architecture. The proposed FASNet architecture achieves state-of-the-art performance on benchmark datasets while being computationally efficient, making it suitable for real-time image similarity applications.

[4] "A Survey of Deep Learning Techniques for Image Similarity Retrieval"

Shaghayegh Rostami, et al. (2020) provides an extensive overview of various deep learning techniques for image similarity retrieval. The paper is a comprehensive survey of the recent developments in this area and is useful for researchers and practitioners in the field of computer vision.

The paper starts by discussing the importance of image similarity retrieval and its various applications, such as image retrieval, object recognition, and image classification. It then provides a brief overview of traditional techniques for image similarity retrieval, such as feature-based methods and metric learning. The main focus of the paper is on deep learning techniques for image similarity retrieval. The paper discusses various deep learning architectures, such as convolutional neural networks (CNNs), Siamese networks, triplet networks, and deep metric learning. It also provides a detailed discussion of the different loss functions that are used in these architectures, such as contrastive loss, triplet loss, and N-pair loss. The paper also covers recent advancements in deep learning techniques for image similarity retrieval, such as the use of attention mechanisms, self-supervised learning, and multi-modal learning. The authors discuss the advantages and limitations of these techniques and provide examples of their applications in image similarity retrieval.

The paper concludes with a discussion of some of the open research challenges in the field of image similarity retrieval. These include the need for large-scale datasets, the development of more robust and efficient deep learning architectures, and the incorporation of contextual information in image similarity retrieval. Overall, the paper provides a valuable resource for researchers and practitioners in the field of computer vision who are interested in image similarity retrieval. The authors have done an excellent job of summarizing the recent developments in this area and providing a comprehensive overview of deep learning techniques for image similarity retrieval.

[5] "Self-Supervised Learning for Image Similarity Search"

Sergey Zagoruyko and Nikos Komodakis, et al. proposes a self-supervised learning approach for image similarity search that learns image representations without the need for manual annotations.

The authors propose a method called "Instance Discrimination," which is a self-supervised learning task that aims to learn representations that are invariant to image transformations. This task involves training a convolutional neural network (CNN) to distinguish between transformed and nontransformed versions of the same image. By doing so, the network learns to extract features that are robust to variations in scale, rotation, and other image transformations. The proposed method is evaluated on three datasets: CIFAR-10, CIFAR-100, and ImageNet. The authors demonstrate that their method outperforms existing state-of-the-art methods for image similarity search, such as Siamese networks and triplet networks. They also show that their method can be used to perform image classification and achieve competitive results with supervised learning methods.

One of the strengths of this method is that it does not require manual annotations, which can be time-consuming and expensive to obtain. Instead, the method relies on a self-supervised learning task that can be applied to large-scale datasets without the need for human labeling. Additionally, the learned representations are highly discriminative and can be used for a variety of downstream tasks, such as image retrieval and classification. However, one limitation of the proposed method is that it requires a large amount of computational resources to train. The authors use a large batch size and data augmentation techniques to increase the efficiency of the training process, but the method still requires a significant amount of computing power.

Overall, this paper presents a promising approach to self-supervised learning for image similarity search, which has the potential to be applied to a wide range of image analysis tasks. By eliminating the need for manual annotations, this method can reduce the cost and time required for training image recognition models.

[6] "Large Scale Image Retrieval with Attentive Deep Local Features"

The paper addresses the challenge of large-scale image retrieval by proposing a deep learning-based approach that combines local feature extraction and attention mechanisms. The proposed approach achieves superior performance compared to traditional methods. The authors start by introducing the concept of local features, which are image regions that have distinctive characteristics and can be used to identify and match images. Local features are extracted using the popular SIFT algorithm, which generates a set of keypoints and descriptors for each image. However, SIFT is computationally expensive and not scalable to large datasets.

To address these issues, the authors propose a deep learning-based approach that learns local features in a data-driven manner. Specifically, they propose a network architecture called Attentive Deep Local Features (ADLF), which consists of a local feature extractor and an attention module. The local feature extractor is a convolutional neural network (CNN) that takes an image as input and generates a set of feature maps. The attention module then selects the most informative regions from the feature maps by assigning a weight to each region. The weighted feature maps are then concatenated to form a global image representation, which is used for image retrieval.

To evaluate the proposed approach, the authors conducted experiments on three large-scale datasets:

Holidays, Oxford5k, and Paris6k. They compared the performance of ADLF with several state-of-the-art methods, including Bag of Visual Words (BoVW), VLAD, and NetVLAD. The results show that ADLF outperforms all other methods on all three datasets in terms of mean average precision (mAP), a commonly used metric for image retrieval. For example, on the Oxford5k dataset, ADLF achieves a mAP of 81.9%, compared to 75.2% for the second-best method (NetVLAD). Overall, the paper provides a novel approach to large-scale image retrieval by combining local feature extraction and attention mechanisms in a deep learning framework. The proposed approach achieves superior performance compared to traditional methods and can be used for various applications, such as content-based image retrieval and object recognition.

[7] "Deep Learning for Similarity Search in Large-Scale Retail Product Images"

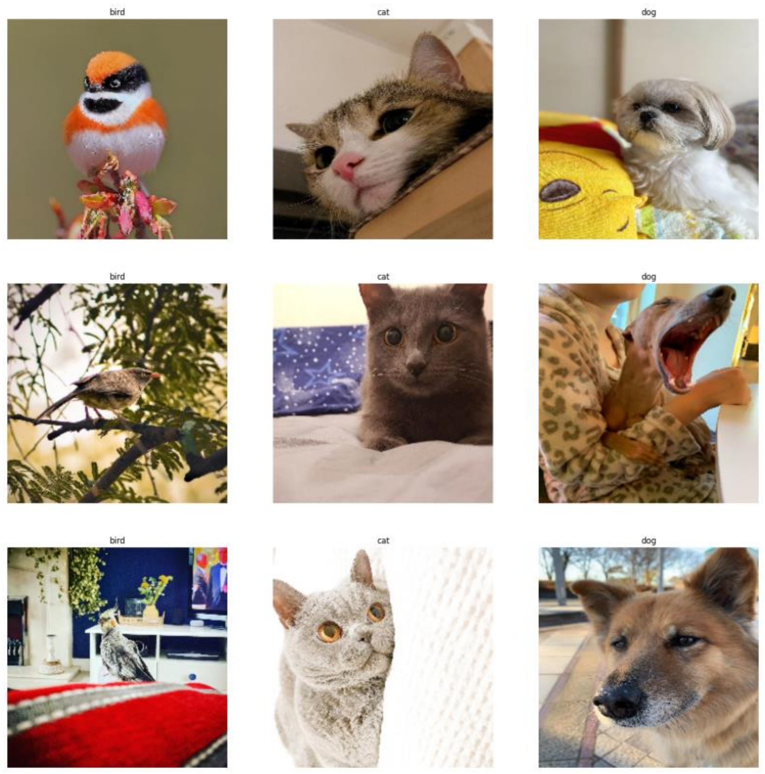
The paper presents a novel framework called Deep Similarity Learning (DSL) that uses a convolutional neural network (CNN) to learn image representations that are optimized for similarity search. The proposed framework is designed to address the challenges of traditional similarity search methods, such as high dimensionality and sparsity of image features, and the high computational cost of processing large-scale image datasets.

The authors first train a CNN model on a large dataset of product images using a triplet loss function, which encourages the model to learn image representations that minimize the distance between similar images and maximize the distance between dissimilar images. Once the CNN model is trained, the authors use it to generate image embeddings, which are low-dimensional representations of the original images that preserve the relevant information needed for similarity search. To perform similarity search using the image embeddings, the authors propose an inverted index-based approach that partitions the image embeddings into clusters and stores them in an inverted index data structure. This approach enables efficient search by reducing the number of comparisons needed to identify similar images.

The authors evaluate the performance of the proposed framework on two large-scale product image datasets, and the results show that the DSL framework outperforms traditional similarity search methods in terms of both accuracy and efficiency. The authors also demonstrate the scalability of the proposed framework by showing that it can handle datasets with millions of images. Overall, the paper presents a deep learning-based approach for similarity search in large-scale retail product images that achieves state-of-the-art performance and scalability. The proposed framework can be applied to various applications such as image retrieval, product recommendation, and visual search in e-commerce platforms.

Description of Dataset

For the purposes of this project, we have selected a subset of a larger dataset containing coloured images of birds, cats and dogs [8]. The original dataset has about 4,149 images for each of the three classes. These images of varying size having an approximate spatial resolution of 920x920 pixels.



Sample images of the dataset.

The subset that we have judiciously selected comprises of 20 images from each of the three classes. Our rationale for choosing a small dataset is to assess the efficacy of our approach when a vast dataset is not readily accessible in a specific scenario. Moreover, utilizing a limited dataset permits us to explore various network architectures and hyper-parameters without significantly investing time and resources in training.

To evaluate the effectiveness of our system, we have created 66 arbitrary pairs, each consisting of 12 images derived from the original dataset. This methodology enables us to assess whether our system is capable of providing an accurate similarity score for a given pair of images. In particular, we aim to observe that images belonging to the same category exhibit a higher similarity score. Thus, we anticipate that two bird images should yield a higher similarity score compared to a pair of images comprising a bird and a cat.

Methodology

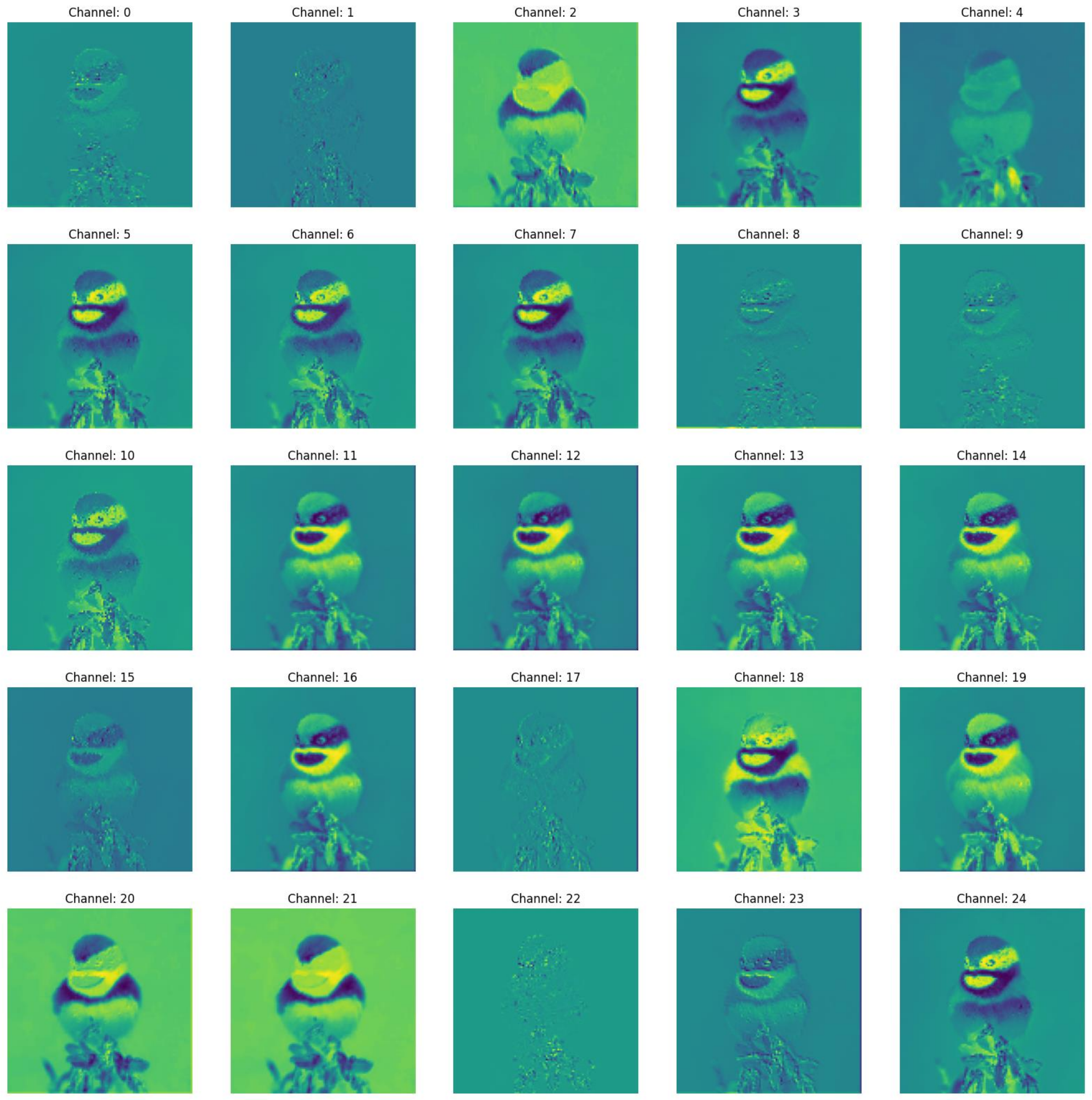
The process that has been followed in the development and working of our approach for finding similarity between images has been explained in details in following subsections, beginning with the most important task feature extraction, followed by computing mean squared difference between feature map of images, and using those parameters to derive a similarity score for a pair of images.

Feature Extraction

Feature extraction is a process of automatically extracting meaningful information or features from an image that can be used for further analysis or processing. These features can be used to represent the visual content of an image, such as colour, texture, shape, and edges. There are various techniques for image feature extraction, ranging from handcrafted methods, such as histogram of gradients (HOG), scale-invariant feature transform (SIFT), and local binary patterns (LBP), to deep learning-based methods, such as convolutional neural networks (CNNs). Handcrafted feature extraction methods rely on a set of predefined rules or algorithms to extract features from an image, while deep learning-based methods learn the features directly from the images through a hierarchical network of convolutional layers.

We have opted for a pre-trained convolution neural network (CNN) that is adept at extracting features from images and has been trained on the ImageNet database. Although VGG-16 is commonly employed for this task, it necessitates significant computational resources and produces dense feature maps, which further prolong the comparison time with the feature map of other images. In contrast, MobileNetV2 is a much more compact network than VGG-16 and is nearly as precise in terms of classification. It incorporates compression and expansion convolution layers that reduce and enlarge the spatial dimensions of feature maps at different levels, resulting in varying outcomes. As a result, we chose the convolution layers located at the conclusion of each block in the network, just before the Batch Normalization layer. In addition, we chose the first and last convolution layer from the network to represent the low- and high-level features of the images descriptively. For the same purpose, many intermediate convolution layers were also selected.

To evaluate the efficacy of our approach on the subset of our dataset, each image was processed via MobileNetV2, and the features from all selected convolution layers were extracted. Instead of the original image, we stored these extracted feature maps for further processing because they represent all of the unique features that may be present in the image.



Different channel of feature map extracted from 1st convolution layer for an image.

Mean Squared Difference

Mean squared difference (MSE) is a common method for measuring the difference or similarity between two sets of data, such as two images. It is calculated as the average of the squared differences between each corresponding element in the two sets of data.

In the context of image processing, MSE is often used as a similarity metric to compare two images. To calculate the MSE between two images, the corresponding pixels in each image are subtracted from each other, the differences are squared, and then the mean of the squared differences is calculated over all the pixel values. We have used MSE to calculate similarity between different levels feature maps of two images.

The formula for calculating MSE is:

$$MSE = \frac{1}{N}\sum\big(I(i) + K(i)\big)^2$$

where I(i) and K(i) are the values of the two-feature map at position i, N is the total number of values present in that feature map. A lower MSE value indicates that the feature map two images are more similar, while a higher MSE value indicates that they are more dissimilar. One of the limitations of MSE is its sensitivity to contrast and brightness, but we are using feature maps instead of original image, that's why this limitation can be ignored here.

Similarity Score

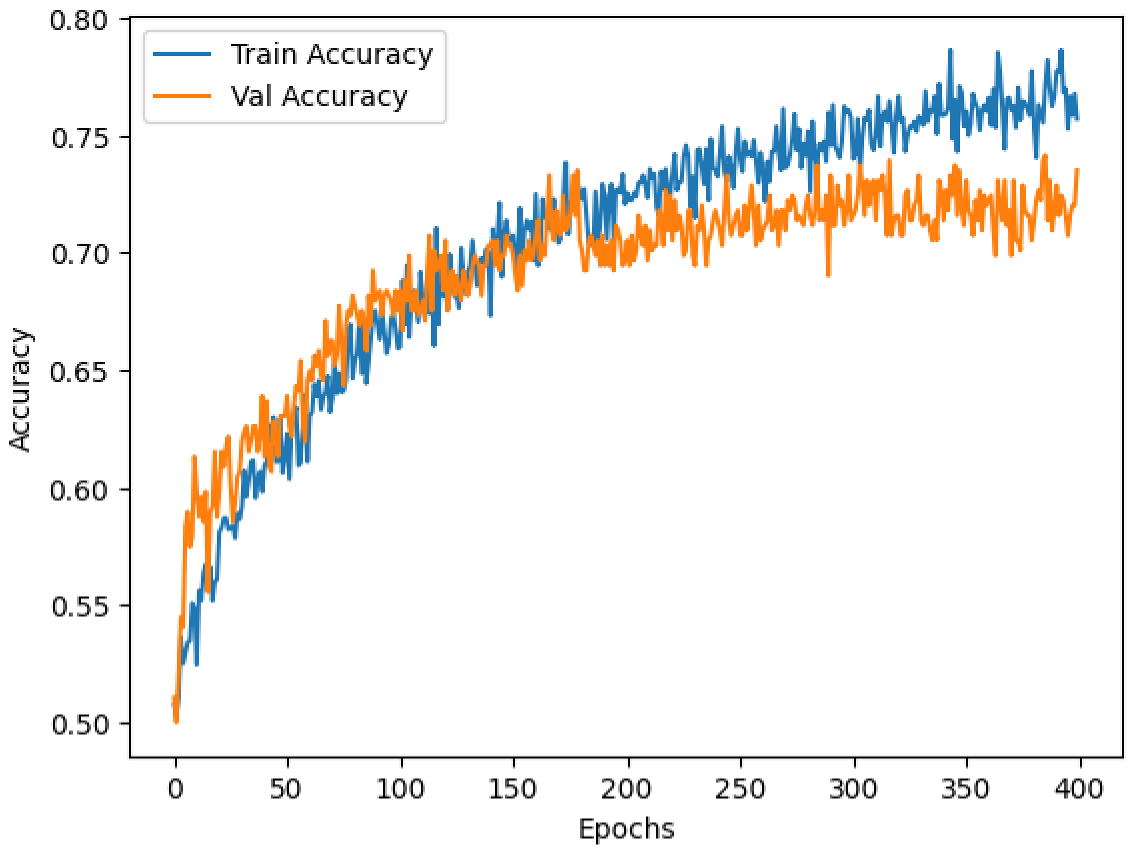
When extracting features from an image using our chosen 18 convolutional layers, 18 feature similarity values are generated from MSE to represent both high-level and low-level similarities between a pair of images. However, obtaining a single similarity score that can appropriately account for both high- and low-level features of images is a challenging task. To address this, we trained a small neural network over our smaller dataset to compute a similarity score that can accurately describe the similarity between images. This approach allowed us to capture the intricate relationships between the extracted features and obtain a more complete representation of the image similarity, despite the limited amount of data available.

Our network consists of 2 fully connected layers apart from input and output layers. We have taken 32 and 64 neurons in respective hidden layers with ReLU activation. Both hidden layers are followed by a Dropout layer to prevent overfitting of the model. We have chosen Sigmoid activation for output layer with a single neuron, so that the similarity score is between 0 and 1, former representing completely dis-similar images and latter representing very similar images.

We constructed our dataset for training of similarity network, by taking unique pair of images across and within classes of our small dataset. Images within the same classes are labelled as 1 representing very similar images, and images belonging to different classes are labelled as 0 representing completely different images. MSE is calculated for each feature map for each pair of images, which is then used as training and validation data for similarity network. Similarity score is given in terms of value ranging from 0 to 1, with value nearing one representing higher similarity.

Experimental Results

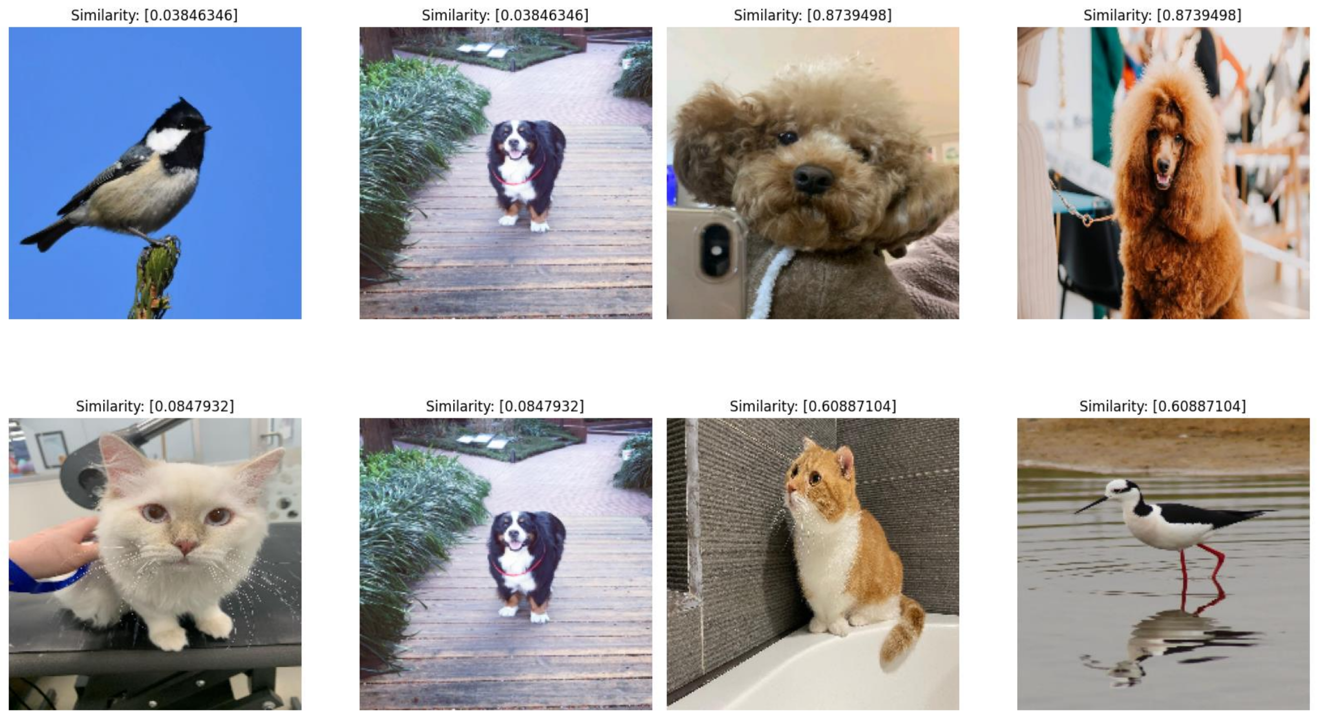
Quantitative results aren't reliable in the case of similarity score, as they assign a similarity score of 1 for all the image pair belonging to same class, and 0 for pair not belonging to the same class. Whereas, the similarity score value can range anywhere in between 0 to 1. As, some image pair belong to different class can have very similar low-level features like background colour, texture, edges, etc. For example, a cat and dog can both be of black or brown colour and their fur is usually of similar texture, that's why their similarity score can lie in the middle rather than on extreme. Similar cases can have impact on the accuracy of model. Still, quantitative results like accuracy can be a nice predictor of performance of model and can help us especially during training. For our dataset, we got training accuracy of 75.69% and validation accuracy of 73.5%. The progress for the same against epochs can be found below:



In the graph, we can see that after about 200 epochs the rate of increase of validation accuracy starts to decrease compared to training accuracy, and it plateaus at around 300 epochs, whereas training accuracy keeps on increasing. This is because our model is getting marginally overfitted which is why we have used Dropout layers in our model.

Qualitative testing can give us better idea of how our model is performing as we can manually verify whether the similarity of an image pair is comparable to the similarity score given by our model. For qualitative test random pair of images taken from original dataset is used for testing of our approach. First, features are extracted for each pair of images, then MSE is calculated among the pair, followed by prediction of similarity score.

Below are some of the test results of similarity score for test pair:



Most of image pairs are getting similarity score comparable to actual similarity of image pair based on class and appearance. But some image pair are getting assign similarity score that might not represent the actual human perceived similarity between them. For example, in the last image pair, an image of cat and bird are getting a similarity score of 0.61 which doesn't represent the actual similarity as they don't have similar colour and they don't belong to same class. These kinds of few errors can arise because of some features that extracted from MobileNetV2 which might be relevant to the original model but not in our case. Overall, about 49 out of 66 image pair were given correct similarity score, i.e., 74.2% accuracy which is comparable to the validation accuracy of our similarity network model (73.5%).

- **Training Accuracy: 75.69%**
- **Validation Accuracy: 73.5%**
- **Qualitative Analysis Accuracy: 74.2%**

Conclusion

We have presented a project that focuses on finding the similarity between any two images using feature maps extracted from a pre-trained convolutional neural network (CNN). Our approach involved extracting the feature maps from 18 convolutional layers of the pre-trained CNN and computing the mean squared difference (MSE) between the corresponding feature maps to obtain a set of feature similarity values.

To obtain a single similarity score that accounts for both high- and low-level features of the images, we trained a small neural network over a limited dataset. Our experimental results demonstrate that the proposed approach effectively captures the intricate relationships between the extracted features and provides an accurate description of the image similarity. We were able to achieve training accuracy of 75.6% and validation accuracy of 73.5%. We also performed qualitative analysis of our model, and for a random set of images we were able to get accuracy of 74.2%.

This approach has numerous potential applications, including image retrieval, object detection, and content-based image analysis. Future work may involve expanding the dataset used for training the neural network, exploring different similarity metrics, or fine-tuning the pre-trained CNN to further improve the performance of the system. Overall, our project demonstrates the potential of using pre-trained CNNs for image feature extraction and similarity computation tasks.

References

[1] "Learning Deep Image Similarity with Spatial Feature Fusion and Semantic Supervision" by Jianan Li, et al. (2021)

[2] "Improved Siamese Network for Similarity Learning" by Yang Liu, et al. (2021)

[3] "FASNet: A Fast and Accurate Siamese Network for Image Similarity" by Junliang Xing, et al. (2020)

[4] "A Survey of Deep Learning Techniques for Image Similarity Retrieval" by Shaghayegh Rostami, et al. (2020)

[5] "Self-Supervised Learning for Image Similarity Search" by Sergey Zagoruyko and Nikos Komodakis (2019)

[6] "Large Scale Image Retrieval with Attentive Deep Local Features" by Yifan Liu, et al. (2019)

[7] "Deep Learning for Similarity Search in Large-Scale Retail Product Images" by Adithya Renduchintala, et al. (2018)

[8] "High Resolution Cat-Dog-Bird Image Dataset (13000)" https://www.kaggle.com/datasets/mahmoudnoor/high-resolution-catdogbird-image-dataset-13000

[9] "A Neural Algorithm of Artistic Style" by Gatys et al. (2015)

Appendix

Code of the program is as follows:

```
#%% md
### Similarity Index
#### By - Ansh Poonia and Jitesh Kumar Yadav
***
importing required libraries
#%%
import tensorflow as tf
import cv2 as cv
import numpy as np
import os
from sympy.utilities.iterables import multiset_combinations
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
#%% md
Selecting layers of MobileNetV2 to be used to compute the feature map of an image
#%%
LAYERS = [
    ('Conv1',1),
    ('block_1_project',1),
    ('block_2_project',1),
    ('block_3_project',1),
    ('block_4_project',1),
    ('block_5_project',1),
    ('block_6_project',1),
    ('block_7_project',1),
    ('block_8_project',1),
    ('block_9_project',1),
    ('block_10_project',1),
    ('block_11_project',1),
    ('block_12_project',1),
    ('block_13_project',1),
    ('block_14_project',1),
    ('block_15_project',1),
    ('block_16_project',1),
    ('Conv_1', 1)
]
#%%
IMAGE_SIZE = (224, 224)
#%% md
Mean Square Difference: to be used to find similarity between feature maps of two
images
#%%
def sl(param_one, param_two):
    m, nh, nw, nc = param_one.get_shape().as_list()
    param_one = tf.reshape(param_one, shape=[-1, nc])
    param_two = tf.reshape(param_two, shape=[-1, nc])
    loss = tf.divide(tf.reduce_sum(tf.square(tf.subtract(param_one, param_two))), (4.0
* nh * nw * nc))
    return loss

def square_loss(param_one, param_two):
    loss = []
    for i,j in zip(param_one, param_two):
        loss.append(sl(i,j))
    return np.array(loss)
#%% md
Importing the pre-trained MobileNetV2 model trained on ImageNet
#%%
def make_model():
    model = tf.keras.applications.MobileNetV2(include_top=False,
input_shape=IMAGE_SIZE + (3,), weights='imagenet')
    model.trainable = False
    output = [model.get_layer(layer[0]).output for layer in LAYERS]
    model = tf.keras.Model([model.input], output)
    return model
#%%
mobile_net_model = make_model()
#%%
mobile_net_model.summary()
#%% md
Reading the images of all three classes (bird, cat and dog) from our small database
#%%
bird_images = []
for i in os.listdir('./dataset/subset/bird'):
    temp = cv.imread(f'./dataset/subset/bird/{i}')
    temp = cv.cvtColor(temp, cv.COLOR_BGR2RGB)
    temp = cv.resize(temp, IMAGE_SIZE)
    bird_images.append(temp)
#%%
cat_images = []
for i in os.listdir('./dataset/subset/cat'):
    temp = cv.imread(f'./dataset/subset/cat/{i}')
    temp = cv.cvtColor(temp, cv.COLOR_BGR2RGB)
    temp = cv.resize(temp, IMAGE_SIZE)
    cat_images.append(temp)
#%%
dog_images = []
for i in os.listdir('./dataset/subset/dog'):
    temp = cv.imread(f'./dataset/subset/dog/{i}')
    temp = cv.cvtColor(temp, cv.COLOR_BGR2RGB)
    temp = cv.resize(temp, IMAGE_SIZE)
    dog_images.append(temp)
#%%
bird_images = np.array(bird_images)
cat_images = np.array(cat_images)
dog_images = np.array(dog_images)
#%% md
Example of images in the dataset
#%%
fig, axs = plt.subplots(nrows=3, ncols=3,figsize=(20,20))
titles = ['bird', 'cat', 'dog']
for i in range(3):
    axs[i][0].imshow(bird_images[i])
    axs[i][0].set_title(titles[0])
    axs[i][1].imshow(cat_images[i])
    axs[i][1].set_title(titles[1])
    axs[i][2].imshow(dog_images[i])
    axs[i][2].set_title(titles[2])
    axs[i][0].axis('off')
    axs[i][1].axis('off')
    axs[i][2].axis('off')
plt.show()
#%% md
Computing the feature map of all the images
#%%
bird_features = []
for i in bird_images:
    bird_features.append(mobile_net_model(np.array([i])))
#%%
cat_features = []
for i in cat_images:
    cat_features.append(mobile_net_model(np.array([i])))
#%%
dog_features = []
for i in dog_images:
    dog_features.append(mobile_net_model(np.array([i])))
#%% md
Generating unique pair combinations of an image class with itself
#%%
combinations_arr = list(multiset_combinations(np.linspace(0,len(bird_features)-
1,len(bird_features), dtype=int), 2))
#%% md
Computing feature loss of a class of images with its own class
#%%
bird_w_bird_loss = []
for i in combinations_arr:
    bird_w_bird_loss.append(square_loss(bird_features[i[0]], bird_features[i[1]]))
#%%
cat_w_cat_loss = []
for i in combinations_arr:
    cat_w_cat_loss.append(square_loss(cat_features[i[0]], cat_features[i[1]]))
#%%
dog_w_dog_loss = []
for i in combinations_arr:
    dog_w_dog_loss.append(square_loss(dog_features[i[0]], dog_features[i[1]]))
#%% md
Representation of feature map extracted from 1st convolution layer
#%%
fig, axs = plt.subplots(nrows=5, ncols=5,figsize=(20,20))
for i in range(5):
    for j in range(5):
        n = i*5+j
        axs[i][j].imshow(bird_features[0][0][0][:,:,n])
        axs[i][j].set_title(f"Channel: {n}")
        axs[i][j].axis("off")
plt.show()
#%% md
Computing feature loss of a class of images with other classes of images
#%%
bird_w_dog_loss = []
for i in bird_features:
    for j in dog_features:
        bird_w_dog_loss.append(square_loss(i,j))
#%%
cat_w_bird_loss = []
for i in cat_features:
    for j in bird_features:
        cat_w_bird_loss.append(square_loss(i,j))
```

```
#%%
dog_w_cat_loss = []
for i in dog_features:
    for j in cat_features:
        dog_w_cat_loss.append(square_loss(i,j))
#%% md
Aggregating all the feature loss of image pair that belong to same class, i.e., are
similar and those that don't belong to same class, i.e., are dis-similar
#%%
sim_features = bird_w_bird_loss * 2 + cat_w_cat_loss * 2 + dog_w_dog_loss * 2
dis_features = bird_w_dog_loss + cat_w_bird_loss  + dog_w_cat_loss
#%%
y = np.array([1] * len(sim_features) + [0] * len(dis_features))
#%%
X = np.array(sim_features + dis_features)
#%%
scalar = StandardScaler()
X = scalar.fit_transform(X)
#%%
X_train, X_test, y_train, y_test = train_test_split(X,y,random_state=42,test_size=0.2)
#%% md
Making a small model for calculating similarity score
#%%
sim_model = tf.keras.Sequential([
    tf.keras.layers.Dense(32, activation='relu', input_shape=(18,)),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.3),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
#%%
sim_model.summary()
#%%
sim_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
#%%
history = sim_model.fit(X_train, y_train, epochs=400,verbose=1,
validation_data=(X_test, y_test))
#%%
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['Train Accuracy', 'Val Accuracy'])
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
#%% md
Testing the model over one pair of image
#%%
plt.imshow(bird_images[0])
#%%
plt.imshow(dog_images[16])
#%%
plt.imshow(bird_images[1])
#%%
loss1 = scalar.transform([square_loss(bird_features[1], dog_features[16])])
loss2 = scalar.transform([square_loss(bird_features[0], bird_features[1])])
#%%
result = sim_model.predict(np.array([loss1[0], loss2[0]]))
#%%
print(result)
```

```
#%% md
The above result is more or less at par, as the pair of images of birds has higher
similarity score that the pair of images of a bird and a dog.
***
Importing test images randomly chosen from the original dataset for qualitative
testing purposes.
#%%
test_images = []
for i in os.listdir("./dataset/test"):
    temp = cv.imread(f'./dataset/test/{i}')
    temp = cv.cvtColor(temp, cv.COLOR_BGR2RGB)
    temp = cv.resize(temp, IMAGE_SIZE)
    test_images.append(temp)
#%%
test_images = np.array(test_images)
#%% md
Computing feature maps for test images
#%%
test_features = []
for i in test_images:
    test_features.append(mobile_net_model(np.array([i])))
#%% md
Generating all the possible pair of test images.
#%%
test_combinations = list(multiset_combinations(np.linspace(0,len(test_features)-
1,len(test_features), dtype=int), 2))
#%% md
Calculating feature loss between the pair of test images
#%%
test_loss = []
for i in test_combinations:
    test_loss.append(square_loss(test_features[i[0]], test_features[i[1]]))
#%%
test_loss = scalar.transform(test_loss)
#%%
test_results = sim_model.predict(test_loss)
#%% md
Plotting the pair of images and their similarity score
#%%
n = 36
fig, axs = plt.subplots(nrows=1, ncols=2, figsize=(10, 5))
for j in range(2):
    axs[j].imshow(test_images[test_combinations[n][j]])
    axs[j].set_title(f"Similarity: {test_results[n]}")
    axs[j].axis("off")
plt.show()
#%%
```