

IMAGE SIMILARITY PREDICTOR USING DEEP LEARNING

Project report submitted in partial fulfillment of the requirement for
the degree of Bachelor of Technology

in

Computer Science and Engineering

By

ANANYA SOOD (191501)

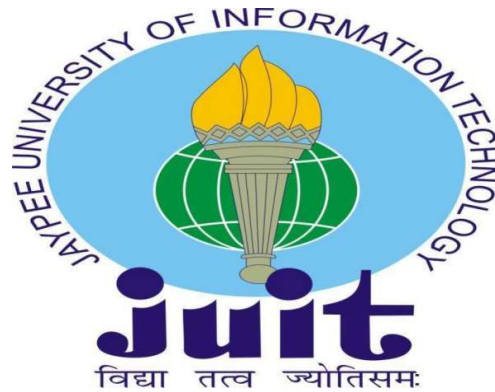
UNDER THE SUPERVISION OF

Dr. Shweta Pandit

&

Prof. Dr. Vivek Kumar Sehgal

to



Department of Computer Science & Engineering and Information
Technology

**Jaypee University of Information Technology Waknaghat,
Solan-173234, Himachal Pradesh**

Candidate's Declaration

I hereby declare that the work presented in this report entitled “**Image Similarity Predictor using Deep Learning**” in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science and Engineering/Information Technology** submitted in the department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology Waknaghat is an authentic record of my own work carried out over a period from August 2022 to December 2022 under the supervision of **Dr. Shweta Pandit**, Assistant Professor(SG), Department of Electronics and Communication Engineering and **Prof. Dr. Vivek Kumar Sehgal**, Professor and Head, Department of Computer Science & Engineering and Information Technology. I also authenticate that I have carried out the above mentioned project work under the proficiency stream **Data Science**.

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature)

Ananya Sood, 191501

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature)

Dr. Shweta Pandit

Assistant Professor(SG)

Department of Electronics and
Communication Engineering

(Co-supervisor Signature)

Prof. Dr. Vivek Kumar Sehgal

Professor and Head

Department of Computer Science
and Engineering and Information
Technology

PLAGIARISM CERTIFICATE

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT			
<u>PLAGIARISM VERIFICATION REPORT</u>			
Date:			
Type of Document (Tick): <input type="checkbox"/> PhD Thesis <input type="checkbox"/> M.Tech Dissertation/ Report <input type="checkbox"/> B.Tech Project Report <input type="checkbox"/> Paper			
Name:		Department:	
Contact No.		Enrolment No.	
E-mail:			
Name of the Supervisor:			
Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters):			
.....			
<u>UNDERTAKING</u>			
I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.			
<u>Complete Thesis/Report Pages Detail:</u>			
- Total No. of Pages =			
- Total No. of Preliminary pages =			
- Total No. of pages accommodate bibliography/references =			
			(Signature of Student)
<u>FOR DEPARTMENT USE</u>			
We have checked the thesis/report as per norms and found Similarity Index at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.			
(Signature of Guide/Supervisor)		Signature of HOD	
<u>FOR LRC USE</u>			
The above document was scanned for plagiarism check. The outcome of the same is reported below:			
Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)
	• All Preliminary Pages		Word Counts
Report Generated on	• Bibliography/Images/Quotes		Character Counts
	• 14 Words String	Submission ID	Total Pages Scanned
			File Size
Checked by Name & Signature		Librarian	
Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com			

ACKNOWLEDGEMENT

I would like to express my profound gratitude to Prof. Dr. Vivek Kumar Sehgal (Professor and Head, Fellow IEI, SM-IEEE, SM-ACM), of the Computer Science and Information Technology department, and Prof. (Dr.) Ashok Kumar Gupta (Dean) of the Jaypee University of Information Technology for their contributions to the completion of my project titled “Image Similarity Predictor Using Deep Learning”.

I would like to express my special thanks to our mentor Dr. Shweta Pandit for the time and efforts she provided throughout the year. Your useful advice and suggestions were helpful to me during the project’s completion. In this aspect, I am eternally grateful to you.

I would like to acknowledge that this project was completed entirely by me and not by someone else.

Ananya Sood

(191501)

TABLE OF CONTENTS

Candidates Declaration	I
Plagiarism Certificate	II
Acknowledgement	III
Table of Contents	IV
List of Abbreviations	V
List of Figures	VI
List of Tables	VII
Abstract	VIII
Chapter 1. Introduction	
1.1 Introduction	1
1.2. Problem Statement	5
1.3. Objectives	5
1.4. Methodology	6
1.5. Organization	7
Chapter 2. Literature Survey	
2.1 Deep Convolutional Neural Networks	11
2.2. Image Similarity Prediction	11
Chapter 3. System Development	
3.1. Resources	16
3.2. Machine Learning	17
3.3. Computer Vision	18
3.4. Image Classification	20
3.5. Deep Learning	20
3.6. Convolutional Neural Networks	21
3.7. CNN Models	27
3.8. Extracting features from a pre-trained network	33

3.9. FastAPI	34
Chapter 4. Performance Analysis	
4.1. Dataset	35
4.2. Frontend	35
4.3. Inference Trials	37
4.4. Threshold Optimization	41
Chapter 5. Conclusion	
5.1. Conclusions	44
5.2. Future Work	45
References	47

LIST OF ABBREVIATIONS

Abbreviations	Meaning
CNN	Convolutional Neural Networks
ANN	Artificial Neural Network
ViT	Vision Transformer
OPMS	Online Pair Mining Strategy
PCA	Principal Component Analysis
SfM	Structure-from-Motion
ResNet	Residual Network
MACs	Multiply-ACcumulates
VGGNet	Visual Geometry Group Network
CSPDarkNet	Cross Stage Partial Dark Network

LIST OF FIGURES

Figure No.	Title
3.1	Convolutional, sub-sampling, and fully connected layers are denoted by the letters C, S, and FC, respectively
3.2	Theory of convolutional neural networks
3.3	Make a feature map matrix using the given data
3.4	ResNet18 Model Architecture
3.5	Efficient Architecture
3.6	MobileNet Architecture
3.7	VGGNet Architecture
3.8	ConvNeXt Model
3.9	CSPDarkNet Architecture
4.1	Frontend Display 1
4.2	Frontend Display 2
4.3	Frontend Response Body Display

LIST OF TABLES

Table No.	Title
4.1	Results with varying input sizes and models with a set of random images.
4.2	Results for E-Commerce Hero Images with Different Models
4.3	Results for transformed images for different models
4.4	Results for transformed images for Deeper CNN Models
4.5	Results for E-commerce images with the mentioned difference between the set of images
4.6	Effect of change in characteristics on Similarity Score

ABSTRACT

This project aims to develop an Image Similarity Predictor using Deep Learning. In recent times, deep learning has played a major role in building real time applications which were earlier based on traditional methods. Our approach makes use of pre-trained CNNs which help extract image embeddings by performing various convolutions and pooling operations. Obtained image embeddings are further compared using cosine similarity and help predict whether the set of images are similar or not.

This Image Similarity Predictor is deployed on the frontend using FastAPI, a modern, fast, web framework used for building APIs with Python. The predictor is deployed on the frontend, wherein the user can upload two images and receive a similarity score between them and whether the two images are similar or not. Its application can be extended to a number of use cases, for example, Image Search Engines, Content-Based Image Retrieval Systems and Recommendation Engines. Our use case is mostly dependent on E Commerce hero images.

Further, the project also involves optimizing the performance of the Image Similarity Predictor to achieve a response time of under 2 seconds. Various experiments were conducted with a set of configurations and settings to improve the response time and obtain better results in terms of prediction accuracy.

Overall, this project demonstrates the potential of using pre-trained CNN models for image embedding extraction and their comparison using cosine similarity score calculation, and the ease with which such models can be deployed on the frontend using FastAPI. The project also provides insights into the challenges of developing and optimizing deep learning models for real-time image similarity prediction.

CHAPTER 1: INTRODUCTION

1.1. Introduction

One of the important areas of research in computer vision are Image Similarity Predictors and play an important role in standalone as well as integrated applications being used across various domains. One of the main goals of an Image Similarity Predictor is to check for the similarity between a set of images based upon their visual features. Hence, with the help of Convolutional Neural Networks being widely used for feature extraction and similarity calculation, Deep Learning is emerging strong in this domain.

Image Similarity Predictors based on Deep Learning have been widely used in various domains such as face recognition, query image retrieval and analysis of medical images. These are just a few to name, but there are a plethora of other domains as well where Image Similarity has made significant contributions for fast and reliable solutions to real-time problems.

There are a number of use cases which have inspired the need for further studies and development in this field. Some of them are as follows:

E-Commerce

E-Commerce has a lot of use cases for image similarity search. From finding similar products, looking up for similar e-commerce hero images, keeping the product catalog well maintained by avoiding the uploading of multiple images of the same product, image similarity has played a significant role. A lot of work of sheer manual intervention has been put to a diminishing end with state-of-the-art image similarity predictors and image similarity search applications.

Digital Document Archives

With the growing amount of data in each and every domain across the world, there's millions of scanned documents, fairly large in size and sparsely structured. Assigning them labels and extracting the ones required instantly requires an approach based on Deep Learning to automate these tasks and eliminate human intervention.

Machine Learning

As they say, data is the new “oil”, but getting and maintaining this high quality training data is a key for the successful implementation of many machine learning projects. Image Similarity using Deep Learning helps us to maintain huge amounts of data by removing the duplicates so as to maintain the quality of data across various domains as per the use case.

A number of approaches have been developed and implemented for finding image similarity. The three major approaches which have played a major and significant role in Image Similarity applications are as follows:

1. Comparing Image Embeddings using Cosine Similarity

This approach to find whether the two images are similar or not involves using embeddings which are extracted from the visual features of the images using Convolutional Neural Networks (CNNs) , which gives us a set of vectorized embeddings. These embeddings can further be compared using various similarity metrics, for example the most common one being the cosine similarity score to determine the percentage of similarity between the set of input images. An image generally has low and high levels of features which help us to distinguish between them depending upon the use case we have at hand.

Here, pre-trained convolutional neural networks(CNNs) are used to extract features from the images. These features are in the form of a vectorized embedding, which represent the visual characteristics of the image in a condensed, numerical form. and are then compared to check for similarity.

For comparison of similarity between the images, cosine similarity is used which gives a score between 0 to 1. Values close to zero indicate that the set of images are not similar whereas a value close to 1 indicates that the set of images are highly similar.

2. Siamese Neural Networks

An Artificial Neural Network(ANN) comprising two or more identical subnetworks generally referred to as the twin networks are known as a siamese neural network. By identical we mean that the two networks have the same set of parameters and weights as well as identical configuration.

In such types of networks, we train either of the sub-networks and the same configuration is also used for other sub networks. Further these networks help to compare two images in terms of similarity by comparing their feature vectors.

These state-of-the-art siamese networks use some sort of either triplet loss or contrastive loss while training as these loss functions are much better suited for such types of networks and helps to improve their accuracy by manifolds.

3. Vision Transformer(ViT)

Vision Transformer is a deep learning architecture for image classification problems that is based on transformers. To describe an image as a series of tokens, each of which is a feature vector taken from a patch of the image, is the primary notion underlying ViT. A transformer-based deep learning model, or ViT, is one of these models. A patch embedding layer and many transformer blocks make up the majority of it.

ViT depicts an image as a sequence of tokens rather than applying convolutional operations to the entire image, which is how it differs from existing image classification models like convolutional neural networks (CNNs). As a result, ViT can effectively handle huge photos and scale to high-resolution images while still being able to capture minute details.

The principal advantages of Vision Transformers (ViT) are as follows:

- Images represented as collections of tokens: Each token in the ViT representation of an image as a sequence of tokens is created using a feature vector extracted from a patch of an image.

- The transformer blocks in ViT use multi-head self-attention to enable the model to pay attention to diverse elements of the image and acquire overall data.

- ViT can handle massive images and scale to high-resolution images since it does not require fixed-sized inputs, unlike other models like CNNs.

One of the key benefits of deep learning-based image similarity predictors is their capacity to automatically learn complex and high-level features from images without the need for manually constructed features. This is especially useful when it may be difficult to identify or quantify the visually important features using standard methods.

Recent years have seen a rise in the use of deep learning-based photo similarity predictions for tasks including object recognition, facial recognition, and content-based image retrieval. Finding images in a large database that are similar to a particular query image is the goal of content-based image retrieval. This is useful in fields like e-commerce where clients might hunt for goods with comparable designs.

Another application where deep learning-based picture similarity predictors have shown considerable promise is facial identification. The objective is to determine whether or not two facial representations of the same individual are the same. This has uses in the security and law enforcement industries, where it's crucial to accurately identify people.

Another area where deep learning-based picture similarity predictors have been successfully used is object recognition. The objective is to determine whether or not the same thing is present in two photographs. This has applications in industries like autonomous vehicles, where precise item detection and identification is essential for safe operation.

Although deep learning-based predictions of image similarity have proved successful, there are still issues that need to be resolved. Deep learning models need a lot of labelled data to be trained, which is one of the difficulties. The need for accurate and effective methods for calculating similarity across large datasets is another obstacle. Additionally, the deep learning models must be comprehensible and explicable, particularly in delicate fields like medical picture analysis.

Finally, it can be said that deep learning-based photo similarity predictions have shown significant promise in a number of domains and have opened up new research avenues in the field of computer vision. Image similarity prediction is anticipated to continue playing a significant role thanks to continued improvements in deep learning methods and hardware.

1.2. Problem Statement

To develop a FastAPI based image similarity predictor using deep learning to compare two images and find a similarity score between them which distinguishes whether the set of images are similar or not.

The problem addressed in this project depicts the need for an accurate and efficient image similarity predictor that can determine whether a set of images are similar or dissimilar. The project aims to develop an Image Similarity Predictor that uses deep learning to extract image embeddings and calculate their cosine similarity score deployed using FastAPI. The predictor allows the users to upload a set of images and receive their similarity score in under 2 seconds, providing a fast, efficient and reliable way of comparing the set of images. Also, this predictor is made configurable to certain parameters to accommodate user choices and requirements.

1.3. Objectives

Our project's main objective is to develop an Image Similarity Predictor using deep learning which is deployed via FastAPI.

Therefore, the conclusive set of objectives that this project aims to address are:

- To propose an efficient Image Similarity Predictor which uses Deep Learning to extract image features.
- To perform an appropriate comparison of the extracted features of the images to check for image similarity based on a certain threshold as set for the similarity score between these sets of images.
- To ensure a response time of under 2 seconds with accurate results and predictions of similarity between the sets of images.
- To make the Image Similarity Predictor configurable to accommodate user choices and requirements.
- To deploy this Image Similarity Predictor via FastAPI to ensure easy access to the user.
- To define an optimal threshold value to differentiate between similar and non similar images.

1.4. Methodology

The approach that we have followed here is to compare image embeddings using Cosine Similarity to check for similarity between them. This approach to find whether the two images are similar or not involves using embeddings which are extracted from the visual features of the images using Convolutional Neural Networks (CNNs) , which gives us a set of vectorized embeddings. These embeddings can further be compared using various similarity metrics, for example the most common one being the cosine similarity score to determine the percentage of similarity between the set of input images. An image generally has low and high levels of features which help us to distinguish between them depending upon the use case we have at hand.

Firstly the set of images are taken as input from the user, which are further pre-processed to ensure consistent dimensionality. This includes resizing the image as required before , making the number of channels consistent in the pair of images, transformed into tensors and are normalized as per the standard values of normalization for the ImageNet dataset. Once the pre-processing is completed, the pair of images are fed to the pre-trained CNN model to extract the embeddings.

An image consists of a number of low and high level features, which can be obtained in a vectorized form using these pre-trained CNNs. These features are in the form of a vectorized embedding, which represent the visual characteristics of the image in a condensed, numerical form. and are then compared to check for similarity.

The images pass through many convolutional layers, pooling layers such as the max pooling layer and the global pooling layer and finally reach the fully connected layer further fed to the softmax function. It is here in the fully connected layer, where we extract the image embeddings from and further flatten them out to find their cosine similarity score.

This cosine similarity score is further normalized, to confine the range between 0 and 1 rather than -1 and 1. Based upon the threshold defined, we further determine whether the pair of images are similar or not.

Here, certain parameters are made configurable to ensure the specification to the use case. These include the list of pre-trained CNNs to choose from, the input size and the threshold value.

1.5. Organization

The project report consists of various chapters as follows: -

Chapter 1:

First chapter gives a succinct overview of the project. It also gives a summary of the fingerprint authentication system and provides an introduction to the project. The project's overall problem statement and its aims are also discussed in this chapter. The chapter also gives a brief overview of the project's methodology and details the stages involved in building an Image Similarity Predictor using deep learning and deployment of the same using FastAPI.

Chapter 2:

The past research on real-time Image Similarity Predictors is discussed in this chapter. Additionally, this offers details about deep learning, machine learning, and neural networks. We have listed a number of journals and associated publications that provide details on the earlier research. The chapter explains how several groups have attempted to employ different models to develop image similarity prediction systems. This chapter includes techniques and their associated results, and by referencing these, we can determine the method to adopt when developing our model or project.

Chapter 3:

This chapter provided details on the procedures we would use to construct the entire project. Both system development and model development are discussed. The chapter contains details regarding the kind of data we'll be working on. Additionally, the chapter has all of the details regarding the libraries we'll be using. Additionally, it provides details on the algorithms that we will be further using for our project.

Chapter 4:

This chapter provides information on how the entire project's work is done and how we have monitored the work at each level. It provides details on the work done at various levels and also gives the outcomes at various levels. It gives details on the model that we built with

the aid of several modules and libraries. It also includes the set of experiments that were conducted, what modifications were made to the configuration and the set of models being added or ruled out to get better results. It offers details about the model's prediction and the response time being taken by these models. The entire chapter gives us details on the results that were observed and the modifications that were made to improve the results in terms of better predictions as well as better response time.

Chapter 5:

The entire conclusion of the work included in this project report is contained in this chapter. It provides information on how to enhance the project and what we can do going forward in line with the research objectives and its enhancement.

CHAPTER 2: LITERATURE SURVEY

As a result of its use in numerous real-world settings, such as image similarity, deep learning has considerably improved during the past ten years. The historical literature has a number of groundbreaking projects utilising deep learning for photo similarity.

The Siamese network, developed by Bromley et al. in 1993, was one of the early and most successful uses of deep learning for image similarity prediction. Two identical subnetworks with common weights make up a Siamese network, which is trained to maximise distances between distinct pairs of pictures and minimise distances between similar pairings. Since then, more deep learning models, such as triplet networks, contrastive loss networks, and ranking loss networks, have been suggested for the prediction of image similarity.

Hadsell et al.'s contrastive loss networks minimise the distance between like photographs and maximise the distance between different pictures in order to learn a similarity metric between pairs of photos. They were first presented in 2006. The objective of contrastive loss networks is to promote the mapping of like images near each other and distinct images far apart. The separation between image pairs serves as the foundation for the loss function.

In order to train a ranking function that can order a collection of photos depending on how similar they are, Wang et al. presented ranking loss networks in 2014. The ranking function has been trained to increase the gap between photos that are similar and those that are dissimilar.

Schroff et al. introduced triplet networks in 2015, which are made up of an anchor network, a positive network, and a negative network. The features are extracted from the anchor image by the anchor network, a related image by the positive network, and a distinct image by the negative network. To achieve the desired outcomes, the anchor and the positive and negative qualities must be as close to one another as possible.

2.1. Deep Convolutional Neural Networks

Recent research has shown that deep convolutional neural network (CNN) produced image descriptors perform better on a number of visual recognition tasks than finely adjusted state-of-the-art systems (Razavian et al., 2014). Girshick (2013), Zeiler and Fergus (2013), and Donahue (2014) have all employed embeddings from cutting-edge CNNs (like Krizhevsky et al. (2012)) to successfully handle a variety of computer vision issues. The approach used in this contribution is the same as that used by Oquab et al. (2014), who trained a CNN on 1512 the ImageNet synsets (Deng et al., 2009), used the trained network's first seven layers as feature extraction algorithms on the Pascal VOC data set, and accomplished outstanding results on the Pascal VOC classification task.

2.2. Image Similarity Prediction

2.2.1. “Image similarity using Deep CNN and Curriculum Learning Image Similarity Prediction” by Srikar Appalaraju, Vineet Chaoji

Image similarity is the discovery of photos that seem to be similar to a reference image. SimNet, a deep Siamese network trained on pairings of positive and negative images utilizing a unique online pair mining technique influenced by Curriculum learning, was the solution that [] presented. In addition, they developed a CNN with multiple scales, where the picture embedding from the final layer is a composite of the embeddings from the top and the bottom levels. They then go on to demonstrate that this multi-scale Siamese network captures fine-grained visual similarity better than conventional CNN's.

Visual search, duplicate product identification, and domain-specific image clustering are a few uses for the ability to find a group of related photos for a given image. Their approach, called SimNet, searches for images that are similar to a new image using a multi-scale Siamese network.

SimNet trains itself to embed images in a 4096-dimensional space using a multi-scale CNN in a Siamese network. The network projects photo pairings into a 4096D subspace after learning a

variety of hierarchical nonlinear transformations, and it makes an effort to minimise the distances between positive image matches and maximise them for negative image matches. Siamese networks need to pair together images, specifically positive image pairs (roughly similar images) and negative image pairs (unsimilar images), in order to learn distance margin. It turns out that attaining a high model performance and speedy model convergence depends on selecting the optimum photo pairings for training. We offer an innovative online pair mining technique (OPMS) to maintain the continuously increasing complexity of photo pairings while the network is being trained.

The primary contributions of the paper are as follows:-

1. CNN is used on several levels by the Siamese network. In the top and bottom levels of this CNN, images are combined. For the goal of determining how similar two images are, our model learns a much better image embedding than a conventional CNN.
2. We use a novel online pair mining technique that was inspired by curricular learning to make sure the model handles local minima better and converges more quickly with a low performance hit.

2.2.2. “Learning to Compare Image Patches via Convolutional Neural Networks” by Sergey Zagoruyko , Nikos Komodakis

This study addresses a fundamental obstacle for many computer vision problems: how to create a general similarity algorithm for comparing picture patches directly from image data (i.e., without utilising manually-designed features).

Such a function is encoded by a CNN-based model that has been trained to take into account a variety of changes in visual appearance. They examine and explore a number of neural network architectures that have been specifically created for this purpose in order to achieve this.

Our decision to represent such a function in terms of a deep convolutional neural network, which is also influenced by recent advancements in neural architectures and deep learning [14, 13], was made in order to achieve that goal.

2.2.3. “Neural Codes for Image Retrieval” by Artem Babenko, Anton Slesarev, Alexandr Chigorin, Victor Lempitsky

It has been demonstrated that a high-level descriptor of the image's visual content may be obtained from the activations that an image causes in the top layers of a substantial convolutional neural network.

In this study, they look into the application of these descriptors (neural codes) in picture retrieval. In our tests, they demonstrate that neural codes outperform conventional retrieval standards even when the convolutional neural network has been trained for a classification purpose unrelated to the retrieval goal (such as Image-Net). We also investigate if retraining the network with a set of images that are comparable to the images seen during testing can improve the retrieval performance of neural codes.

They go on to evaluate the effectiveness of the compressed neural codes and show that on various datasets, a simple PCA compression produces great short codes with cutting-edge accuracy. In general, neural codes show to be much more resistant to such compression than alternative state-of-the-art descriptors.

Finally, they show that when discriminative dimensionality reduction is trained on a dataset of pairs of matched images, PCA-compressed neural codes do even better.

Their quantitative research generally demonstrates that neural codes can function as useful visual descriptors for picture recollection.

2.2.4. “Learning Image Embeddings using Convolutional Neural Networks for Improved Multi-Modal Semantics” by Douwe Kiela, Leon Bottou

By linking a language representation vector generated by the feature extraction layers of a deep convolutional neural network (CNN) trained on a significant labelled object recognition dataset with a visual representation vector, we may create multi-modal idea representations. Features developed using the conventional bag-of-visual-words technique perform noticeably worse than those made using this transfer learning procedure. The WordSim353 and MEN semantic relatedness evaluation tasks experimental findings are presented. We employ visual characteristics identified in either ESP Game or ImageNet photos. Image similarity is the discovery of photos that seem to be similar to a reference image.

2.2.5. “Learning Fine-Grained Image Similarity with Deep Ranking” by Jiang Wang, Yang Song, Thomas Leung, Chuck Rosenberg, Jingbin Wang, James Philbin, Bo Chen, Ying Wu

It's difficult to learn fine-grained visual similarities. It must monitor visual differences both within and between classes. This study introduces a deep ranking system that directly extracts similarity metrics from images using deep learning. It can learn more than models with hand-crafted characteristics. To adequately explain the photos, a unique multiscale network structure has been constructed. It is also advised to master the distributed asynchronous stochastic gradient model utilising a strong triplet sampling technique. The suggested method for visual feature extraction beats deep classification models and models made by hand, according to numerous tests.

2.2.6. “Siamese network features for image matching” by Iaroslav Melekhov, Juho Kannala, Esa Rahtu

For many computer vision applications, including structure-from-motion (SfM), multi-view 3D reconstruction, image retrieval, and image-based localization, finding matching images across enormous datasets is crucial. In order to identify matching and mismatching photo pairings, this study recommends employing feature vectors built on neural networks, whose similarity is determined by Euclidean distance.

Convolutional neural networks that use a contrastive loss function in a Siamese network architecture to learn from labelled examples of matching and non-matching picture pairs are used to create the feature vectors. Siamese architecture has only been used for local image patch matching and face image verification; it has not yet been applied to general image retrieval or whole-image matching.

Their test findings demonstrate that the proposed features outperform baseline features obtained with networks that are trained for image classification tasks in terms of matching performance. The features enhance photo matching for untrained places in novel surroundings and are highly generalizable. This is true even if our training data's labelling of matching and unmatching pairs is imperfect. In terms of image retrieval applications, the results are promising, but there is still room for improvement by using more training photo pairings with more precise ground truth labels.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1. Resources

3.1.1. Hardware Requirements

- A physical memory (RAM) of 4 GB and above are required
- Hard disk capacity: Maximum 5 GB
- Laptop

3.1.2. Operating System

- Windows7/8/8.1/10

3.1.2. Language Requirement

- Python Version 3.9.0

3.1.2. Tools and Framework

- Torch Version 2.0.0
- Torchvision
- Pillow
- FastAPI
- Uvicorn
- Timm

3.2. Machine Learning

Unsupervised techniques, often known as machine learning, are composed of a variety of algorithms that can identify fundamental correlations and patterns from incoming data and form conclusions on their own without specific instructions or direct human involvement. Human behaviour served as the main design motivation for such algorithms. It was necessary to create a system that can "see" like humans and then infer conclusions from what it has seen. When compared to humans, the majority of the algorithms created in the 1990s did not successfully show that they were more accurate and efficient at performing recognition tasks. A number of methods, including artificial neural networks, have been employed to solve the challenges associated with classifying images, making judgements, and making predictions[1].

One subset of artificial intelligence is computer vision. The basic goal is to build an artificially intelligent machine that works similarly to the human brain. When used for a variety of tasks like image categorization, motion detection, 3D modeling, and localisation, a computer may automatically build high-level knowledge from text, picture, and video input in this field [2]. Classifying processes is one of the most crucial tasks for machine learning algorithms. Due to the fact that the majority of system input data is in this format, photos often receive the most classifications. The practice of classifying images based on their traits and aesthetic appeal is known as image classification.

Numerous techniques and algorithms were proposed in order to improve accuracy and address classification problems, which finally led to the development and application of deep learning techniques and artificial neural networks [3].

The creation of the deep learning technique was prompted by issues with picture classification and learning capacity in machine learning. Deep learning, sometimes referred to as a deep neural network, has numerous hidden layers and is capable of autonomously learning and extracting characteristics even from unlabeled input.

It has proven to perform well in a number of applications, including object detection, segmentation, and image classification. The performance of deep learning has been suggested to be improved by many models utilising various architectures. The most well-known sort of

design is the CNN, or Convolutional Neural Network [4]. Convolutional neural networks are a subtype of deep neural networks, which are these networks. A convolutional neural network has several layers, each of which handles a specific processing function. Major companies like Google, Facebook, and AT&T have all hired numerous scientists to develop and introduce the best designs that can be employed in CNNs because this network has proven to be so effective. The best CNN designs and their individual components are discussed in this section [5].

3.3. Computer Vision

The science that gives computers or other machines the ability to recognise objects is the most simple and understandable definition of computer vision. Computer vision is believed to involve more than just photographing light. The process' practical and functional components include memory, recognition, assessment, and estimation, with the main objective being to store and extract details and traits from what has been seen. The goal of computer vision is to give machines the ability to understand reality through the processing of digital data, a process that is frequently referred to as "picture perception." By extracting substantial information and properties from digital signals and performing challenging calculations, this type of machine understanding is accomplished [6].

The ability of computer vision to distinguish and capture motions based on the study of each element of a picture, such as human movements, has significantly improved over the past 20 years. This process, which uses sensors with various qualities, consists of the four sub-processes initialization, detection, estimate, and recognition. Each of them is broken up into an enormous variety of various categories and processes.

It should be mentioned that the usage of various data sources and altering system parameters may have an effect on performance and efficacy [7]. Image segmentation, which has grown to a high level of sophistication, is one of the most practical applications of computer vision. The two types of image segmentation are supervised methods and unsupervised approaches.

For specific class images or image sets with different classes, the supervised method classification can be utilised. It is crucial to perform a mental check before to using this

strategy to make sure that the segmentation method has been correctly implemented and that it is effective. A person must intervene and verify the method's accuracy by contrasting the segmented picture outputs with pre-defined categories.

The supervised technique appears difficult and unsuitable for many computer vision applications as a result of these constraints. The classification of enormous amounts of data or classes is one of these limitations. Unsupervised approach was developed to address these problems, however it has a number of shortcomings.

These segmentation techniques were developed to build complex mathematical functions and heavy algorithms that can adapt to a variety of input formats, including Image, Video, and Text[8], and can automatically carry out the needed job with great accuracy and precision. Using cameras and tracking algorithms, the field of computer vision has made significant advancements in overcoming localization problems during the past 20 years.

In order to monitor and recognise characteristics in this way, a variety of strategies are applied, which fixes an issue with prior computer vision algorithms. These methods include accelerating computing and processing rates as well as refreshing the available memory. In other words, rather than only relying on links between two images, powerful computer vision algorithms can now locate the desired object in real-time.

Context modelling has recently attracted a lot of attention and work from academics due to computer vision's particular purpose of enhancing and improving the process of processing images and extracting information from them. Contextual modeling's key objectives are to give data structure, make it simpler, and help users understand how to maintain it. For people who want to apply context modelling in their jobs today, there are several apps available that supply and clarify information such as specific methodology, applications, and techniques.

In a number of applications, including face recognition, object identification, and image categorization, computer vision has made considerable strides. However, due to the variations in the structure and implementation of each of these applications, a distinct set of underlying

hypotheses and calculations are required [11]. According to studies, systems that can analyse images and extract information such as gender and various body parts have been developed by combining computer vision and graphics. Thanks to the greatly improved development of these processes that have taken them to their current levels, robots are now capable of determining population density, recognising and foreseeing impending occurrences, and monitoring individuals [12].

3.4. Image Classification

Images are classified using a variety of sophisticated computer vision algorithms based on their characteristics, content, and other elements that may affect the classification process. The goal of image classification is to categorise and assign each pixel in a digital image to one of several defined classes. Colourful data is transformed to grayscale or a certain colour level before its attributes can be recognised and retrieved. The data can be used to create feature-maps in a picture after being categorised. The spectral pattern in the data for each pixel serves as a numerical foundation for categorization in multi-spectrum data, which are frequently employed for this.

The image classification process requires the use of an algorithm that differs from one designed for a particular purpose. The five fundamental steps that define and carry out the approach are preprocessing, feature selection and extraction, training sample selection, classification processing, and accuracy evaluation [13].

3.5. Deep Learning

A deep learning system makes use of numerous layers to discern between higher-level data and unprocessed input stages. When a picture is fed into a deep neural network, for instance, the lower layers first find the edges and extract the most evident components before the higher levels recognise the deeper elements. Deep learning, one of the most significant and useful machine learning algorithms now available, has achieved substantial success in a number of applications, including image analysis, speech recognition, and text recognition [14].

By extracting characteristics from input photos using two strategies—supervised and unsupervised with a distinctive architectural characteristic—this approach teaches students to recognise and categorize items. The origins of deep learning may be traced back to 1943, when Walter Pitz and Warren McCulloch created a computer model based on neural networks seen in the human brain. For many reasons, including the unavailability of cutting-edge hardware and software, this approach was not regarded to be helpful until recently. Instead, they employed a mix of algorithms and mathematics known as "threshold logic" to simulate the human cognitive process.

By extracting characteristics from input pictures using two strategies—supervised and unsupervised with a specific architectural trait—this method teaches students to identify and classify items. Walter Pitz and Warren McCulloch, who created a computer model based on the neural networks discovered in the human brain in 1943, are credited with creating deep learning. Until recently, this tactic was not considered to be effective for a number of reasons, including the lack of cutting-edge software and technology. Instead, to simulate how people think, scientists employed a combination of mathematics and computers dubbed "threshold logic". .

3.6. Convolutional Neural Networks

The idea that neural networks have a structure similar to the human brain was established even before computers were created. Propositional reasoning was initially used to assess neural networks. Convolutional and backward propagation artificial intelligence fields has helped deep neural networks, but there has also been a substantial barrier to CNN development. The issue arose because there weren't enough systems available to complete a challenging and time-consuming activity. As a result, neural networks were developed, but they were not profitable. This condition existed up to the advent of GPUs. Today's powerful networks and systems make it possible to use CNN in real-world applications[15].

3.6.1. Overview

Many hidden brain neurons that can each detect and classify specific features make up a convolutional neural network. With this method, the input size is decreased without compromising the image quality. Neurons are able to extract characteristics for a range of uses, such as voice recognition and picture edge extraction. Numerous convolutional cores make up the CNN structure, which draws attention to crucial visual characteristics so the system can recognise them. By reducing the need to process a number of extraneous pixels that are present in each image but add nothing useful, time will be saved.

The number of neural layers in a convolutional neural network can be increased in order to extract more high-level information [16]. In 1989, a researcher by the name of LeCuN utilised CNN for the first time to analyse network-like topological data (images and time series data), demonstrating its effectiveness and piquing academics' attention. As a result, CNN was later inspired by the same essential structure [17].

A convolutional neural network's fundamental structure consists of two convolutional layers—the Conv layer and two sub-sampling layers—which carry out the fully connected layer's sampling function. The first hidden layer, which consists of three filters, is where the process starts. Three weighted maps of the qualities are created after processing. The subsampling layer's nonlinear activation function is then used to create a new feature map. Then, using the following subsampling layer, three trained filters receive these maps from the Conv layer in order to produce three new maps.

The second subsampling layer transfers the output of the fully connected layer to the neural network during training, where it is transformed into a vectorized coordinate and fed into the network.

The process's flow is shown in the below figure:

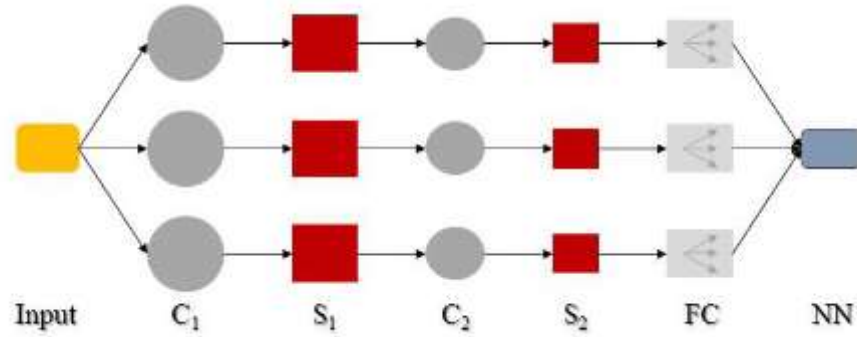


Fig 3.1: Convolutional, sub-sampling, and fully connected layers are denoted by the letters C, S, and FC, respectively.

3.6.2. Architecture

Since its creation, convolutional neural networks have undergone several iterations to enhance performance and address various problems, including feature extraction.

Convolutional layers, Pooling layers, and Fully Connected Layers are the three primary neural layers that now make up the traditional CNN architecture []. These layers carry out a substantial amount of numerical processing-related tasks. In order to enhance the performance of the neural network, regulatory techniques such batch normalisation or dropout are occasionally introduced to the network parameters [18].

Furthermore, there are two stages to the network training: forward steps and backward steps. The next phase seeks to keep the input weight and value for each layer constant. After that, data loss is measured using output prediction. Chain rules computations estimate the gradient of each parameter in the backward step based on the loss anticipated in the first step, updating the parameters for the subsequent forward step. Up until the network is fully trained, these two phases are repeated [19]. The hypothetical CNN building is displayed:

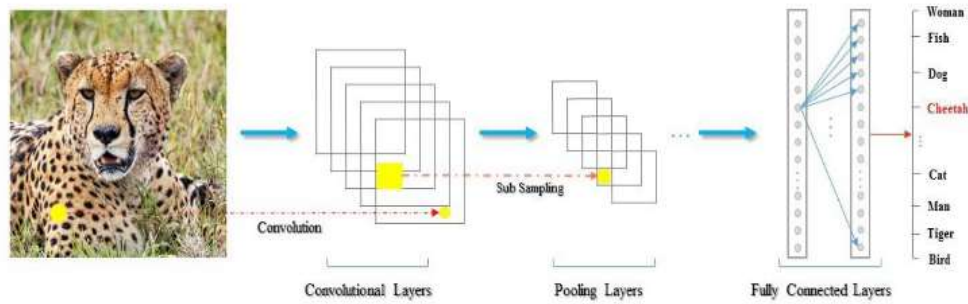


Fig 3.2: Theory of convolutional neural networks

3.6.3. Convolutional Layer

The convolutional layer (Conv), the top layer of this neural network design, regulates how much output is generated in response to the receiver's input. Each of the filters that make up the convolution layer is composed of a kernel of neurons, or individual neurons. Receiver fields are the smaller chunks into which input images are separated, also known as convolution kernels. By dividing and breaking the input into pieces, which accentuates critical information, it is easier to extract important information. The outputs are identified by the cores that manage the length and width of the data. A 2D map of feature activation is also created. By identifying a certain characteristic in the input, the network quickly selects the filters that were activated. Depending on the weight distribution capability of each Conv layer, various groupings of image characteristics may be retrieved from the input without degrading the input weight. Processing techniques can be categorised according to the kind and size of filters.

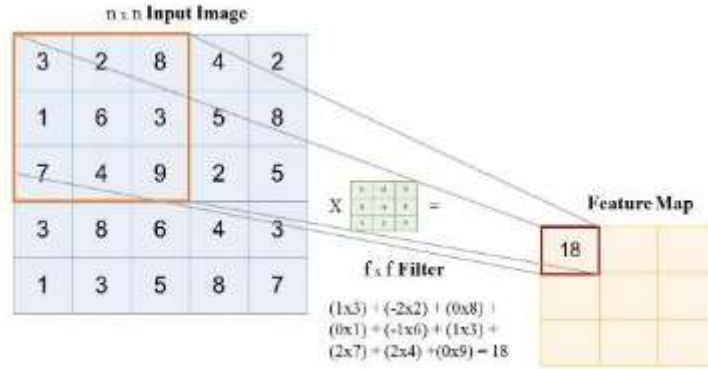


Fig 3.3: Make a feature map matrix using the given data

3.6.4. Pooling Layer

The pooling layer often follows the convolution layer. The main purpose of this layer is to reduce the size of the parameters and the feature map. Utilising pooling techniques facilitates the extraction of a variety of properties. By cutting the size of the feature map into smaller subsets and eliminating unnecessary links, the network's complexity may be managed and performance can be increased.

3.6.5. Fully Connected Layer

In a convolutional neural network design, one or more fully connected layers are built after the last pooling layer. The FC layer converts two-dimensional feature maps created from prior layers into a one-dimensional vector in order to display more features. Fully connected layers make up around 90% of a CNN model and function similarly to traditional neural networks. It offers a preset length output feature vector that may be used for classification or taken into consideration as a feature vector for further processing [20]. The information-processing neurons in each filter are totally interconnected with the neurons in the layer above. The FC layer has a large number of parameters, which makes training difficult. Therefore, one of the current hot challenges for scientists is how to reduce the number of connections and filters while maintaining accuracy [21].

3.6.6. Activation Function

The output of each layer in the neural network is governed by a set of mathematical equations called the activation functions. Each neuron in the network has a connection to these functions. Based on each neuron's role, this technique decides whether or not the functions should be active [22]. To halt processing and move output to the following layer, functions must be invoked. There have been several alternative activation functions proposed for use in different neural networks, including Sigmoid, Tanh, Maxout, Swish, ReLU and its variants, such as leaky ReLU, ELU, and PReLU [23]. Choosing the best activation function is accelerated since the learner is aware of nonlinear properties. One of the recently considered activation functions is MISH, which outperforms ReLU in deep neural networks [24].

3.6.7. Batch Normalization

A method for training very deep neural networks is batch normalisation, often known as batch norm. Artificial neural networks are used to accelerate the process and enhance its performance and stability. Batch norm, which standardises inputs into discrete layers in accordance with their classifications, makes the training process more stable. A deep network might be developed with a lot fewer training cycles [25].

3.6.8. Dropout

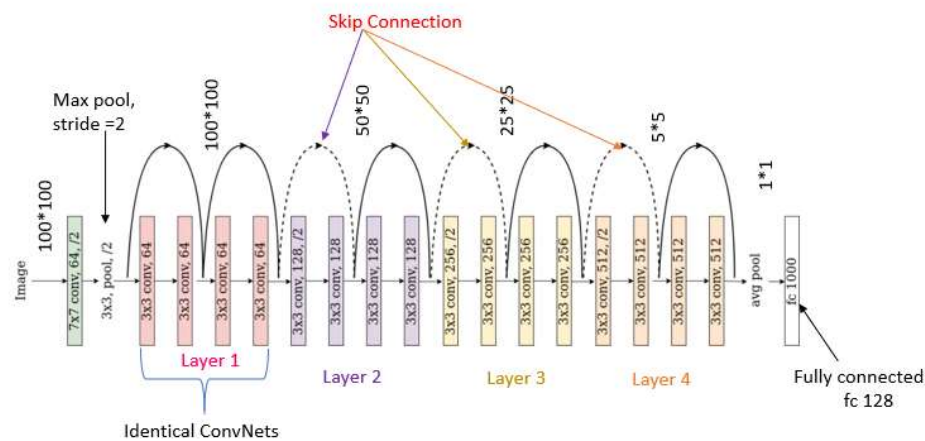
By eliminating complicated coordination in the training data, Google Research's dropout regularisation approach decreases the number of connections that exist between neurons in the network. This method is a really amazing way to increase the effectiveness of neural networks, particularly CNNs. In neural networks, overfitting results from the numerous connections that learn a nonlinear relationship being tightly fitted. Dropout reduces certain connections to produce various thin network designs. At the conclusion, a thorough network with low weights is chosen. Almost all other networks in the model are taken into account while choosing the network architecture [26].

3.7. CNN Models

3.7.1. ResNet

For computer vision applications, the Residual Network (ResNet) deep learning model is used. It is a design for a convolutional neural network (CNN) that can support a large number of convolutional layers—possibly thousands. The limited number of layers that earlier CNN systems could accommodate had an adverse effect on performance. When they added additional layers, the researchers ran into the "vanishing gradient" problem.

The backpropagation approach used to train neural networks uses gradient descent to reduce the loss function and identify the weights that minimize it. If there are too many layers, the gradient will eventually grow so tiny that it "disappears". With each successive layer, performance will likewise get saturated or deteriorate.



Fig

3.4: ResNet18 Model Architecture

The "Skip connections" provides a fix for the vanishing gradient problem. Convolutional layers that are initially inactive (many identity mappings; ResNet) are stacked, skipped, and the activations from the previous layer are recycled. Skipping speeds up the initial training process by reducing the number of layers in the network.

After the network has been restrained with all layers expanded, the leftover portions, also known as the residual parts, are free to explore more of the feature space of the input picture. With batch normalization and nonlinearity in between, the majority of ResNet models skip two or three layers at once. HighwayNets, a form of more complex ResNet architecture, have the potential to learn "skip weights," which determine how many layers to skip on the fly.

3.7.2. EfficientNet

The EfficientNet convolutional neural network design and scaling approach equally scales each depth, breadth, and resolution parameter using a compound coefficient. The EfficientNet scaling approach uses a set of established scaling factors to consistently increase network breadth, depth, and resolution as opposed to conventional practise, which scales these variables arbitrarily. To utilise $2N$ times more computer power, for instance, we might simply raise the network depth by αN , the width by βN , and the image size by γN . The original microscopic model's constant coefficients, alpha, beta, and gamma, were discovered via a tiny grid search. To scale the network's width, depth, and resolution evenly, EfficientNet employs a compound coefficient.

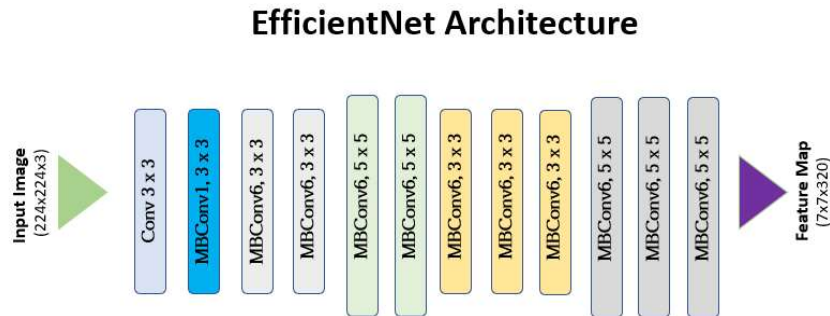


Fig 3.5: Efficient Architecture

The compound scaling technique is justified by the fact that larger input pictures need additional layers and channels in order to enhance the network's receptive area and catch more fine-grained patterns on the larger image. Squeeze-and-excitation blocks and MobileNetV2 inverted bottleneck residual blocks make up the basic EfficientNet-B0

network. Using orders of magnitude fewer parameters on the CIFAR-100 (91.7%), Flowers (98.8%), and 3 more transfer learning datasets, EfficientNets likewise transfer efficiently and attain cutting-edge accuracy.

It is anticipated that EfficientNets might potentially serve as a new basis for next computer vision tasks by significantly enhancing model effectiveness. In order to help the greater machine learning community, we have open-sourced all of the EfficientNet models.

3.7.3. MobileNet

Google has made its open-source MobileNet machine vision model available for classifier training. It creates a lightweight deep neural network by using depthwise convolutions to drastically lower the number of parameters compared to previous networks. The first mobile computer vision model for Tensorflow is called MobileNet.

In comparison to other networks that utilise ordinary convolutions and the same depth in the nets, it greatly reduces the number of parameters by employing depthwise separable convolutions. Compact deep neural networks are produced as a result. Google open-sourced the MobileNet family of convolutional neural networks (CNNs), which may be used to train extremely quick and tiny classifiers.

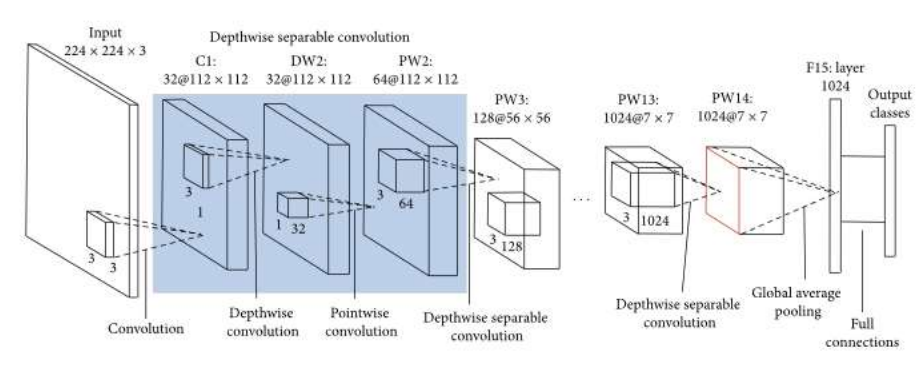


Fig 3.6: MobileNet Architecture

The number of multiply-accumulates (MACs), which is a measurement of the quantity of fused multiplication and addition operations, affects the network's performance and power use.

The word separable, which describes this convolution, comes from the notion that a filter's depth and spatial dimension may be separated. Consider the Sobel filter, which is used in image processing to identify edges.

The filters' height and width measurements can be separated. As a matrix product of $\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$ transpose $\begin{bmatrix} -1 & 0 & 1 \end{bmatrix}$, Gx filter may be thought of.

We can see that the filter has posed as something else. Although it only contains six, it displays that it has nine parameters. The separation of its height and width makes this feasible.

The same concept holds true for a depth dimension that is distinct from horizontal, providing us with a depthwise separable convolution in place of a depthwise convolution. Then we cover another dimension with a filter.

3.7.4. VGGNet

Visual Geometry Group, or VGG, is the abbreviation for a deep convolutional neural network (CNN) architecture that is typical in that it includes several layers. VGG-16 or VGG-19, respectively, contain 16 or 19 convolutional layers. The term "deep" refers to the quantity of layers.

Innovative item identification models are built using the VGG architecture. The VGGNet, a deep neural network created for a variety of applications and datasets outside of ImageNet, outperforms benchmarks. It's still one of the most widely used image recognition frameworks out there.

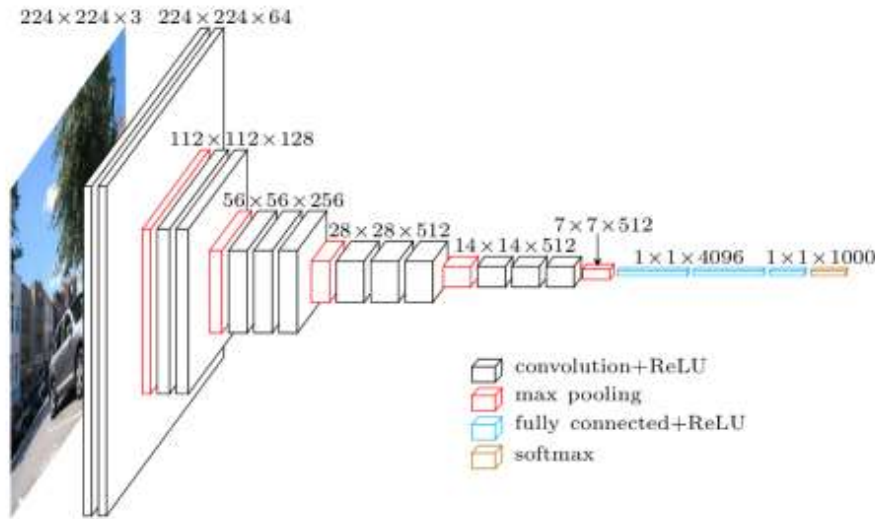


Fig 3.7: VGGNet Architecture

The convolutional neural network model called the VGG model, or VGGNet, that supports 16 layers is also known as VGG16. It was developed by A. Zisserman and K. Simonyan from the University of Oxford. Their model was reported in the study "Very Deep Convolutional Networks for Large-Scale Image Recognition."

In ImageNet, the VGG16 model achieves top-5 test accuracy of about 92.7%. A dataset called ImageNet has over 14 million photos that fall into about 1000 types. It was also among the most well-liked models submitted at ILSVRC-2014. It significantly outperforms AlexNet by substituting a number of 3×3 kernel-sized filters for the huge kernel-sized filters. Nvidia Titan Black GPUs were used to train the VGG16 model over a period of many weeks.

The previously discussed 16-layer VGGNet-16 is capable of classifying photos into 1000 different item categories, including keyboard, animals, pencil, mouse, and many more. The model also supports images with a resolution of 224×224 .

3.7.5. ConvNeXt

ConvNeXt adopts the concept of inception that ResNeXt had earlier developed. Information is divided, changed, and combined during this process.

The focus of this is depthwise convolution. It is a particular illustration of clustered convolutions. In this case, the number of groups and channels is equal.

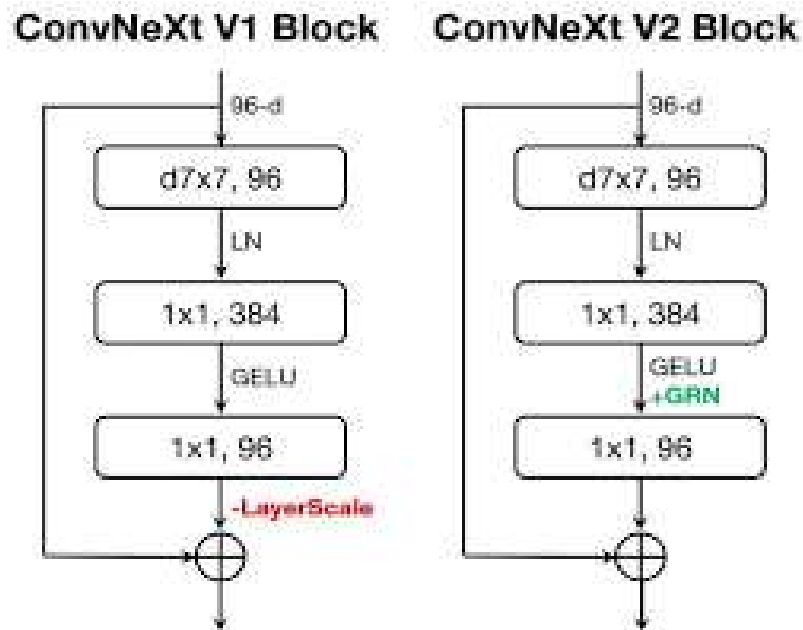


Fig 3.8: ConvNeXt Model

These convolutions mirror the self-attention equivalent of the weighted sum operation. They only integrate information in the spatial dimension and work on a per-channel basis. Using this concept leads to a huge improvement in performance.

To match the strength of vision transformer models with a global receptive field, larger kernel sizes are necessary. Through self-attention that spans the entire image, the vision transformer has a tendency to take in the entire

scenario at once.

3.7.6. CSPDarkNet

A convolutional neural network that makes use of DarkNet-53 as its foundation is known by the term CSPDarknet53. The base layer's feature map is divided in two using a CSPNet method, and these two sections are then combined. When adopting the split-and-merge approach, the gradient flow in the network is greater. For YOLOv4, this CNN serves as the foundation.

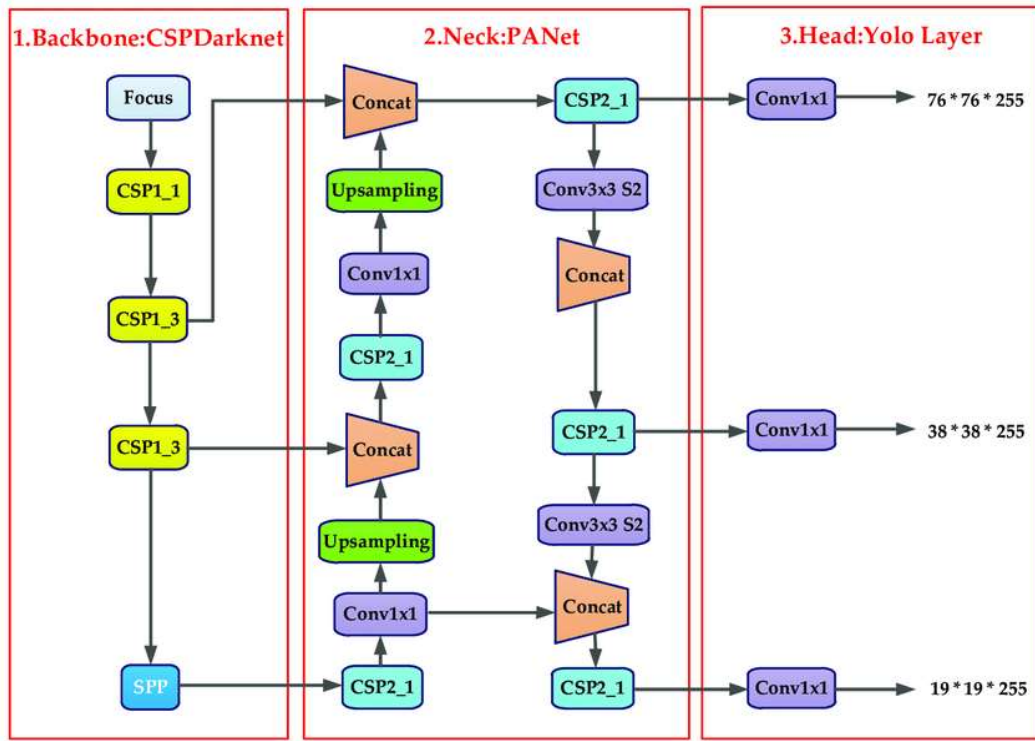


Fig 3.9: CSPDarkNet Architecture

3.8. Extracting features from a pre-trained network

It has been demonstrated that features derived from pre-trained neural networks trained on large-scale tasks may provide outcomes that are superior to many commonly used methodologies and high-accuracy clustering when applied to fresh data sets. Even when

grouping into categories for which the network was not explicitly trained, it has demonstrated great performance on novel tasks.[27] These deep convolutional neural network (CNN) VGG-F network characteristics are made available via the open source collection of pre-trained model implementations known as MatConvNet. Three fully linked layers and five convolutional layers make up the network. 1000 soft-max values are obtained once the characteristics from the penultimate layer's neural activity are retrieved. The network was trained using a sizable data set of 1.2 million images, and it was then used to categorise 1000 distinct sorts of objects. Other data sets will utilise the retrieved features as descriptors. [28]

3.9. FastAPI

FastAPI is a rapid web framework based on common Python type hints for building APIs with Python 3.7+. It makes it possible for developers to create apps efficiently and fast. FastAPI, which is based on the Starlette web server and includes features like automated data validation, error management, and interactive API manuals, facilitates the building of online apps. This is one of the swiftest web frameworks for Python.

1. Because it takes use of the capabilities of the Pydantic and Starlette libraries, its performance is among the finest and comparable to that of NodeJS and Go.
2. FastAPI not only delivers outstanding performance but also significant speed for development, a decrease in human-caused mistakes in the code, ease of learning, and full production readiness.

FastAPI is completely compatible with well-known API standards like the JSON format and OpenAPI.

CHAPTER 4: PERFORMANCE ANALYSIS

4.1. Dataset

Here, we do not use any specific dataset of images to train the model, but rather we make use of pre-trained CNN models. Thus, no training set is required for our model. Since our use case is for E-Commerce hero images, different samples of hero images were collected from E-commerce websites such as Amazon, Flipkart etc.

Customers see a hero image as soon as they click on a product listing or in their search results. This is the image that distinguishes a product from its rivals and entices consumers to learn more about it.

Set of two images of electronic products, cosmetics, grocery items, stationary etc were gathered to form the test dataset. These images were of varying sizes and dimensions, some of them with slight levels of similarity whereas others of a noticeable amount of similarity to ensure the correctness of the predictions.

4.2. Frontend

This project was deployed on the frontend using the FastAPI UI Swagger. FastAPI is a cutting-edge web framework for creating Python RESTful APIs. Due to its simplicity, speed, and resilience, it has quickly been a favorite among developers since its initial release in 2018.

The frontend allows the user to give the following as inputs:

1. First image to be checked for similarity
2. Second image to be checked for similarity
3. Pre-Trained CNN model to be used from a drop down list of models as provided.

As soon as the Execute button is clicked, an output is generated with the calculated values for the following:

1. Similarity score obtained between the set of input images whose value lies between 0 and 1.
2. A string value, either “Yes” or “No” stating whether the set of input images are similar or not based on the threshold value as set in the config file.
3. Time taken for the execution in milliseconds.

Image Similarity Predictor 0.0.1 OAS3
An application to check the similarity between two input images

default

POST /cosine_similarity_score Get Cosine Similarity Score

Parameters

Name Description

model_name * required string (query) resnet18

Request body * required multipart/form-data

image_1 * required string (binary) Choose File No file chosen

image_2 * required string (binary) Choose File No file chosen

Execute

Fig 4.1: Frontend Display 1

Here, we upload two images of a Television. These two images vary in context of the view and are in the .jpg format. We also choose a CNN model, ResNet18 from a set of available models and then we further click on the execute button to get the desired results.

Image Similarity Predictor 0.0.1 OAS3
An application to check the similarity between two input images

default

POST /cosine_similarity_score Get Cosine Similarity Score

Parameters

Name Description

model_name * required string (query) resnet18

Request body * required multipart/form-data

image_1 * required string (binary) Choose File tv_1.jpg

image_2 * required string (binary) Choose File tv_2.jpg

Execute

Fig 4.2: Frontend Display 2

We receive the following as the response body:

```
Response body
{
  "Similarity Score": 0.8528041489492493,
  "Are the images similar?": "Yes",
  "Time taken in milliseconds": 1289.3362845288886
}
```

Fig 4.3: Frontend Response Body Display

4.3. Inference Trials

A number of trials had been conducted to analyze the results for varying sets of images on a set of CNN models. Based on the performance of each of the models being used, a set of three models were chosen as the most optimal one and a threshold was set to distinguish between similar and non-similar images.

The trials with specific details and their respective details have been mentioned below:

TRIAL 1: Set of input images are tested over a set of models like ResNet18, Enception_B0 and MobileNetV2 with varying image sizes. The results obtained are as follows:

Table 4.1: Results with varying input sizes and models with a set of random images

Image 1	Image 2	Model Name	Model Version	Input Size	Similarity Score	Time taken(ms)
		Resnet	resnet18	224	0.3348660932	736.7250919
				416	0.2546172572	897.1583843
				512	0.280176996	1126.07618
		Efficientnet	efficientnet_b0	224	0.05767446011	805.3966032
				416	0.04232284427	936.4049435
				512	0.05250382051	1126.119375
		Mobilenet	mobilenet_v2	224	-0.3502461016	341.1450386
				416	-0.3490512371	319.6718693
				512	-0.3487915695	269.7618008
		Resnet	resnet18	224	0.4116253853	786.3633633
				416	0.4728201032	1202.44956
				512	0.4444875012	1215.913773
		Efficientnet	efficientnet_b0	224	0.1803062558	727.3771763
				416	0.08190860599	1262.457609
				512	0.06841967255	1390.268326
		Mobilenet	mobilenet_v2	224	-0.2479596734	382.0011616
				416	-0.2458454072	383.6834431
				512	-0.2454129905	388.1311417

TRIAL 2: Further, a set of E-commerce hero images are tested over a set of models like ResNet18, Enception_B0 and MobileNetV2 with image size as 224*224. The results are obtained as follows:

Here, the Similarity Scores were normalized to avoid the negative values of the same being observed.

The Normalized values are displayed alongside and would help better define the set of images similar or non-similar based on the threshold score.

Further, from now on the similarity score displayed will be the normalized similarity score.

Table 4.2: Results for E-Commerce Hero Images with Different Models

Product Name	Image 1	Image 2	Model Name	Model Version	Input Size	Similarity Score	Time taken(ms)	Are they similar?	Normalized Similarity Score	Time taken(ms)	Are they similar?
Dettol Soap			Resnet	resnet18	224	0.5582407713	739.4063473	Yes	0.7791203856	3859.804853	Yes
			Efficientnet	efficientnet_b0	224	0.2516269088	580.9600353	No	0.6298134842	1589.175701	Yes
			Mobilenet	mobilenet_v2	224	0.4679504037	266.2346363	No	0.733975172	1000.296116	Yes
Instant Coffee			Resnet	resnet18	224	0.4498453438	593.890667	No	0.724922657	1573.181629	Yes
			Efficientnet	efficientnet_b0	224	0.2103108764	720.3171253	No	0.685155468	1584.196329	Yes
			Mobilenet	mobilenet_v2	224	0.3310152292	316.4970875	No	0.665987046	777.4391174	Yes
Sanitizer			Resnet	resnet18	224	0.7194362283	696.9234943	Yes	0.8597180843	1405.280352	Yes
			Efficientnet	efficientnet_b0	224	0.491863966	580.793807	No	0.745931983	1932.631969	Yes
			Mobilenet	mobilenet_v2	224	0.6435696483	262.6597881	Yes	0.8217848539	718.6286449	Yes
Toothbrush			Resnet	resnet18	224	0.6847165823	617.087841	Yes	0.8423582911	1964.878559	Yes
			Efficientnet	efficientnet_b0	224	0.5843703747	646.7080116	Yes	0.7921851873	1786.388712	Yes
			Mobilenet	mobilenet_v2	224	0.3848917484	256.8950653	No	0.6924458742	643.2051659	Yes
Pendrive			Resnet	resnet18	224	0.5541655421	886.2748146	Yes	0.7770828089	1642.747879	Yes
			Efficientnet	efficientnet_b0	224	0.3348092437	688.0271435	No	0.6674046516	1667.070627	Yes
			Mobilenet	mobilenet_v2	224	0.02623596787	284.9938869	No	0.513117968	760.8580589	Yes
Pen			Resnet	resnet18	224	0.5523452163	813.5721684	Yes	0.7761726379	1423.225403	Yes
			Efficientnet	efficientnet_b0	224	0.3368597925	586.8954659	No	0.6684299111	5011.227008	Yes
			Mobilenet	mobilenet_v2	224	0.08800016344	307.7068329	No	0.5440000892	789.6764278	Yes
Bottle			Resnet	resnet18	224	0.6280888319	707.9238892	Yes	0.814044415	1612.895367	Yes
			Efficientnet	efficientnet_b0	224	0.4023365974	647.7334499	No	0.7811682987	2409.142437	Yes
			Mobilenet	mobilenet_v2	224	0.6934704781	317.4960613	Yes	0.846735239	8.846735239	Yes
Watch			Resnet	resnet18	224	0.5052074194	721.978426	Yes	0.7526037097	2027.361155	Yes
			Efficientnet	efficientnet_b0	224	0.187396124	735.3823185	No	0.5936980844	2217.149973	Yes
			Mobilenet	mobilenet_v2	224	0.5437835455	289.4668579	Yes	0.7718917727	997.4017143	Yes
Diary			Resnet	resnet18	224	0.4545905292	658.9214802	No	0.7272952795	1490.435839	Yes
			Efficientnet	efficientnet_b0	224	0.1003547534	623.1217384	No	0.5501773953	2773.525476	Yes
			Mobilenet	mobilenet_v2	224	-0.3051995039	299.7965813	No	0.3474002481	800.4062176	No

TRIAL 3: The next trial consisted of comparing a similar set of images, hence for the same a set of transformations were applied to a source image to produce a new set of images with different properties.

The following transformations were applied:

The 'transforms_list' variable contains a list of data augmentation transforms that can be applied to images before they are input into a deep learning network. Each transform modifies the input image in some way, and when combined, they can help to increase the variety and robustness of the training data, which may enhance model performance.

The list's transforms are each briefly described below:






1. 'RandomHorizontalFlip' turns the image horizontally at random with a probability of 0.5. This helps to add variance to the training data and decreases the model's sensitivity to the orientation of objects in the image.
2. The 'RandomRotation' command rotates the image at random, between -10 and 10 degrees. The model will be able to manage photos shot at various angles or in various orientations thanks to the model's increased diversity and robustness as a result.
3. The 'RandomResizedCrop' command resizes the image to a random size between 224x224 and the original image size while cropping the image to 224x224. If the model is more diverse and strong, it will be better equipped to handle images with various sizes and aspect ratios.
4. "ColorJitter(brightness=0.5)": Randomly changes the image's brightness by a factor of up to 0.5. The model might gain more diversity and robustness as a result, enabling it to handle images captured under varied lighting scenarios.
5. 'RandomAffine': The image is rotated, scaled, and sheared using a random affine transformation. This can offer the model additional resilience and variety, enabling it to handle photos captured from various angles more deftly.
6. "ColorJitter(hue=0.3)": Changes the image's hue at random by up to 0.3. By adding diversity and robustness, the model may be better able to handle photos with various color casts.

7. 'ColorJitter(saturation=0.5)': Alters the saturation of the image at random by a value of up to 0.5. As a result, the model might receive more diversity and robustness, enabling it to deal with images that have different saturation levels.

8. "RandomPerspective": "RandomPerspective" alters the image to make it appear as though you are viewing it from a different angle. This can help give the model more variety and resilience so that it can handle images taken from different angles more skillfully.

These set of images were checked for similarity using various models with the image size as 512*512. The results obtained are as follows:

















Table 4.3: Results for transformed images for different models

Product Name	Image 1	Image 2	Model Name	Model Version	Input Size	Similarity Score	Time taken(ms)	Are they similar?
TV(Transformed-0)			Resnet	resnet18	512	0.900292933	1247.74456	Yes
			Efficientnet	efficientnet_b0	512	0.8420031071	1363.506317	Yes
			ConvNeXt	convnext_tiny	512	0.8686514497	2571.590185	Yes
			ConvNeXt	convnext_base	512	0.8793603778	8630.71847	Yes
			MobileNet	mobilenet_v2	512	0.974134326	610.7227802	Yes
TV(Transformed-1)			Resnet	resnet18	512	0.9739716649	1115.211964	Yes
			Efficientnet	efficientnet_b0	512	0.9845130444	1383.251429	Yes
			ConvNeXt	convnext_tiny	512	0.9417986274	2593.089581	Yes
			ConvNeXt	convnext_base	512	0.9443613291	8708.068371	Yes
			MobileNet	mobilenet_v2	512	0.9858493209	401.7748833	Yes
TV(Transformed-2)			Resnet	resnet18	512	0.8366554379	1341.895342	Yes
			Efficientnet	efficientnet_b0	512	0.7829236388	1531.070232	Yes
			ConvNeXt	convnext_tiny	512	0.7577900887	2480.923176	Yes
			ConvNeXt	convnext_base	512	0.7901024818	8678.598881	Yes
			MobileNet	mobilenet_v2	512	0.9514205456	333.7273598	Yes
TV(Transformed-3)			Resnet	resnet18	512	0.9235289693	1185.921192	Yes
			Efficientnet	efficientnet_b0	512	0.9365375638	1517.156839	Yes
			ConvNeXt	convnext_tiny	512	0.9104378819	2482.058764	Yes
			ConvNeXt	convnext_base	512	0.93621099	8510.525227	Yes
			MobileNet	mobilenet_v2	512	0.9864225984	388.5285854	Yes
TV(Transformed-4)			Resnet	resnet18	512	0.7538198233	1141.103268	Yes
			Efficientnet	efficientnet_b0	512	0.6285532713	1405.774117	Yes
			ConvNeXt	convnext_tiny	512	0.6455427408	2775.927067	Yes
			ConvNeXt	convnext_base	512	0.676004529	8517.36474	Yes
			MobileNet	mobilenet_v2	512	0.6910771728	367.1507835	Yes
TV(Transformed-5)			Resnet	resnet18	512	0.9457140565	1157.198668	Yes
			Efficientnet	efficientnet_b0	512	0.927149415	1394.244194	Yes
			ConvNeXt	convnext_tiny	512	0.9178698063	2730.755568	Yes
			ConvNeXt	convnext_base	512	0.9454369545	8659.169197	Yes
			MobileNet	mobilenet_v2	512	0.9901431799	357.6667309	Yes
TV(Transformed-6)			Resnet	resnet18	512	0.9907981806	1133.870602	Yes
			Efficientnet	efficientnet_b0	512	0.9933161139	1319.916725	Yes
			ConvNeXt	convnext_tiny	512	0.9873760939	2709.930897	Yes
			ConvNeXt	convnext_base	512	0.9911057949	8004.403067	Yes
			MobileNet	mobilenet_v2	512	0.9991630316	336.7693424	Yes
TV(Transformed-7)			Resnet	resnet18	512	0.9967323542	1168.980837	Yes
			Efficientnet	efficientnet_b0	512	0.997523725	1377.719879	Yes
			ConvNeXt	convnext_tiny	512	0.9930843115	2670.876026	Yes
			ConvNeXt	convnext_base	512	0.99609375	8621.114731	Yes
			MobileNet	mobilenet_v2	512	0.9997081161	348.2527733	Yes

TRIAL 4: Further, the same set of E-Commerce Images were tested for much deeper CNN Models such as ResNet50, ResNet101 and VGG16 with image size as 512*512.

The results obtained are as follows:

Table 4.4: Results for transformed images for Deeper CNN Models

Product Name	Image 1	Image 2	Model Name	Model Version	Input Size	Similarity Score	Time taken(ms)	Are they similar?
TV(Transformed-0)			Resnet	resnet50	512	0.9113080502	3095.2847	Yes
			Resnet	resnet101	512	0.8932785988	4482.766151	Yes
			VGG	vgg_16	512	0.8980676532	5734.280825	Yes
TV(Transformed-1)			Resnet	resnet50	512	0.9585080147	2703.313828	Yes
			Resnet	resnet101	512	0.9687157869	4252.116442	Yes
			VGG	vgg_16	512	0.948841095	5690.315723	Yes
TV(Transformed-2)			Resnet	resnet50	512	0.8522388339	2843.516111	Yes
			Resnet	resnet101	512	0.8600888848	4526.68643	Yes
			VGG	vgg_16	512	0.7466290593	5595.859289	Yes
TV(Transformed-3)			Resnet	resnet50	512	0.9252770543	3034.378529	Yes
			Resnet	resnet101	512	0.9309987426	4736.463785	Yes
			VGG	vgg_16	512	0.909802258	5645.214319	Yes
TV(Transformed-4)			Resnet	resnet50	512	0.7324208021	2832.993269	Yes
			Resnet	resnet101	512	0.7459215522	4810.199022	Yes
			VGG	vgg_16	512	0.6132537127	5580.623388	Yes
TV(Transformed-5)			Resnet	resnet50	512	0.9579864144	2653.297186	Yes
			Resnet	resnet101	512	0.9530726671	4584.93042	Yes
			VGG	vgg_16	512	0.94296664	5430.747509	Yes
TV(Transformed-6)			Resnet	resnet50	512	0.9874249697	3180.438042	Yes
			Resnet	resnet101	512	0.9878621101	4570.614815	Yes
			VGG	vgg_16	512	0.986507833	5586.16066	Yes
TV(Transformed-7)			Resnet	resnet50	512	0.9960561395	2961.393833	Yes
			Resnet	resnet101	512	0.9957258701	4447.139502	Yes
			VGG	vgg_16	512	0.9932473898	7864.495993	Yes

4.4. Threshold Optimization

Initially, the threshold value was set to 0.50 as the range of the Similarity Score was between 0 and 1 post normalization. But, after performing a number of trials it was observed that there needs to be an optimal value to distinguish between similar and non-similar images.

Hence, a number of sets of images were tested for a different set of models and a conclusion was drawn that changes in what specific set of features led to how much of a change in the similarity score.

Table 4.5: Results for E-commerce images with the mentioned difference between the set of images

Product Name	Image 1	Image 2	Model Name	Model Version	Input Size	Similarity Score	Are they similar?	Similarity Score(Point wise matching)	Comments
Huggies			Resnet	resnet18	512	0.9198479652	Yes	0.963	Change in: Size and related info, Child image, price on the top left
			CSPDarkNet	cspdarknet53	512	0.9842757583	Yes		
			Efficientnet	efficientnet_b0	512	0.8987013698	Yes		
			ConvNeXt	convnext_tiny	512	0.9019941092	Yes		
Huggies			Resnet	resnet18	512	0.8465135098	Yes	0.827	Change in: Size and related info, Child image and posture, price on the top left is missing
			CSPDarkNet	cspdarknet53	512	0.9373447895	Yes		
			Efficientnet	efficientnet_b0	512	0.800550282	Yes		
			ConvNeXt	convnext_tiny	512	0.8129379153	Yes		
Pendrive			Resnet	resnet18	224	0.7770828009	Yes	0.212	Change in: Orientation and view, although it is the same product
			CSPDarkNet	cspdarknet53	224	0.9170632958	Yes		
			Efficientnet	efficientnet_b0	224	0.6674046516	Yes		
			ConvNeXt	convnext_tiny	224	0.6471964121	Yes		
Iphone			Resnet	resnet18	224	0.8804056644	Yes	0.739	Change in: Colour, display image
			CSPDarkNet	cspdarknet53	224	0.7409863472	Yes		
			Efficientnet	efficientnet_b0	224	0.7601656914	Yes		
			ConvNeXt	convnext_tiny	224	0.7409863472	No		
Bulb			Resnet	resnet18	224	0.9131631851	Yes	0.956	Change in: Packaging
			CSPDarkNet	cspdarknet53	224	0.912674427	Yes		
			Efficientnet	efficientnet_b0	224	0.8796024919	Yes		
			ConvNeXt	convnext_tiny	224	0.8385127783	Yes		
TV			Resnet	resnet18	512	0.8529041409	Yes	0.94	Change in: Orientation(left)
			CSPDarkNet	cspdarknet53	512	0.9306813478	Yes		
			Efficientnet	efficientnet_b0	512	0.7182552218	Yes		
			ConvNeXt	convnext_tiny	512	0.7377798809	Yes		
TV			Resnet	resnet18	512	0.9060668945	Yes	0.97	Change in: Orientation(right)
			CSPDarkNet	cspdarknet53	512	0.9386340976	Yes		
			Efficientnet	efficientnet_b0	512	0.7993991375	Yes		
			ConvNeXt	convnext_tiny	512	0.8175299168	Yes		
Sanitizer			Resnet	resnet18	512	0.8732848167	Yes	0.8	Change in: product bottle
			CSPDarkNet	cspdarknet53	512	0.9803743958	Yes		
			Efficientnet	efficientnet_b0	512	0.8247905374	Yes		
			ConvNeXt	convnext_tiny	512	0.8160049915	Yes		
Sanitizer			Resnet	resnet18	512	0.8892416358	Yes	0.8	Change in: Packaging color, language
			CSPDarkNet	cspdarknet53	512	0.980820179	Yes		
			Efficientnet	efficientnet_b0	512	0.8745308518	Yes		
			ConvNeXt	convnext_tiny	512	0.8495647907	Yes		
Oil			Resnet	resnet18	512	0.9364672899	Yes	0.776	Change in: Product Description Language
			CSPDarkNet	cspdarknet53	512	0.9776338339	Yes		
			Efficientnet	efficientnet_b0	512	0.9325746298	Yes		
			ConvNeXt	convnext_tiny	512	0.9321652055	Yes		
Monitor			Resnet	resnet18	512	0.9230335951	Yes	0.953	Change: No noticeable change, but are the hero images for the same product with different monitor sizes
			CSPDarkNet	cspdarknet53	512	0.9415080547	Yes		
			Efficientnet	efficientnet_b0	512	0.9007775221	Yes		
			ConvNeXt	convnext_tiny	512	0.8554666042	Yes		
Face Wipes			Resnet	resnet18	512	0.9068748951	Yes	0.957	Change in: Logo on product label
			CSPDarkNet	cspdarknet53	512	0.9624779224	Yes		
			Efficientnet	efficientnet_b0	512	0.8798915148	Yes		
			ConvNeXt	convnext_tiny	512	0.8352175951	Yes		
Moisturizer			Resnet	resnet18	512	0.9937085509	Yes	0.95	Change: Content on label
			CSPDarkNet	cspdarknet53	512	0.9902694225	Yes		
			Efficientnet	efficientnet_b0	512	0.9929708242	Yes		
			ConvNeXt	convnext_tiny	512	0.9887746572	Yes		

Here, results have also been compared with the already integrated application which was built on the approach of point wise matching. Further, an analysis was conducted to study the effect of change in view, orientation, color etc on the Cosine Similarity Score, the same is shown in the table below:

Table 4.6: Effect of change in characteristics on Similarity Score

Change in	Models	Effect on similarity score
Basic details like price, size , quantity etc	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	10%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	8%
1-2 missing details in packaging	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	18%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	15%
Rotation and view	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	30%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	26%
Color	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	20%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	26%
Modification in packaging	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	13%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	13%
Left orientation	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	24%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	20%
Right Orientation	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	16%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	15%
Product bottle, same information	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	17%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	13%
Description Language, label color	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	13%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	10%
Description Language	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	7%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	6%
Same hero image, different product description	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	10%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	10%
Logo on product label	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	13%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	11%
Content on product label	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	1%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	1%
Content on product label as well product packaging	ResNet18, EfficientNetb0, ConvNeXt_tiny ▼	3%
	CSPDarkNet53, EfficientNetb0, ConvNeXt_tiny ▼	2%

CHAPTER 5: CONCLUSION

5.1. Conclusion

In this project, our aim was to build an Image Similarity Predictor to distinguish between similar and non-similar images. The approach used was to extract image embeddings from pre-trained CNNs in a vectorized format and their comparison, which was done using Cosine Similarity.

Trials were conducted for various sets of E-commerce images with varying image sizes such as 224*224, 416*416 and 512*512.

Many pre-trained CNN models were used to compare the results and choose the three most efficient ones which are:

1. ResNet18
2. EfficientNetB0
3. ConvNeXt_tiny

These were the three top models which performed well on the set of images as per our use case for E-commerce hero images.

To determine the optimal value of threshold, the effect of changes in products in terms of color, packaging, label content, view, orientation etc on the Cosine Similarity Score were analyzed. It was concluded that:

1. Change in color and change in the design of the product container majorly impact the Cosine Similarity Score.
2. Product orientation and view also add to the variation in the Cosine Similarity score in some of the cases.

3. Looking at the results of the set of E-commerce hero images that have been tested, a threshold between 0.75-0.80 seems to be optimal to differentiate between similar and non-similar images.

Hence, the optimal value of the threshold was set to 0.80 .

5.2. Future Work

Deep learning and computer vision are vast fields with new advancements each day. As of now, only a limited number of models were chosen, but at the later stages further models can be incorporated and a performance comparison could be done to update the list of models with the best results.

The use case of this Image Similarity Predictor can be extended to various other E-commerce applications in various domains. A few of them could be as follows:

1. Fashion: Based on a hero image that included a clothing or accessory, the image similarity predictor may make suggestions for relevant products based on colour, pattern, or style.
2. Home furnishings: The picture similarity predictor may recommend furniture or accessories that might seem similar to the arrangement in the hero image by using the colour, texture, or design of the environment as a model.
3. Food: If a dish is shown in a hero image, the image similarity predictor may provide recipes or related products based on the dish's flavour, ingredients, or cuisine.
4. Beauty: Based on colour, finish, or formula, products similar to a cosmetics look that is exhibited in a hero image may be suggested.

The client would utilize the hero image in each of these scenarios as a starting point to research related products, and the image similarity predictor would help expedite the buying process by making helpful suggestions.

REFERENCES

- [1] W.-L. Chao, “Machine learning tutorial,” *Digit. Image Signal Process.*, 2011.
- [2] E. Murphy-Chutorian and M. M. Trivedi, “Head pose estimation in computer vision: A survey,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 4, pp. 607–626, 2008
- [3] M. E. Cintra, M. C. Monard, H. A. Camargo, and T. P. Martin, “A comparative study on classic machine learning and fuzzy approaches for classification problems,” 2005.
- [4] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, “Understanding deep learning requires rethinking generalization,” *arXiv Prepr. arXiv1611.03530*, 2016.
- [5] F. Altenberger and C. Lenz, “A non-technical survey on deep convolutional neural network architectures,” *arXiv Prepr. arXiv1803.02129*, 2018.
- [6] J. A. E. Anderson et al., “Task-linked diurnal brain network reorganization in older adults: A graph theoretical approach,” *J. Cogn. Neurosci.*, vol. 29, no. 3, pp. 560–572, 2017
- [7] T. B. Moeslund and E. Granum, “A survey of computer vision-based human motion capture,” *Comput. Vis. image Underst.*, vol. 81, no. 3, pp. 231–268, 2001.
- [8] H. Zhang, J. E. Fritts, and S. A. Goldman, “Image segmentation evaluation: A survey of unsupervised methods,” *Comput. Vis. image Underst.*, vol. 110, no. 2, pp. 260–280, 2008. S. A. Nazeer, N. Omar, and M. Khalid, “Face recognition system using artificial neural networks approach,” in *2007 International Conference on Signal Processing, Communications and Networking. IEEE*, 2007, pp. 420–425.
- [9] R. J. Radke, “A survey of distributed computer vision algorithms,” in *Handbook of Ambient Intelligence and Smart Environments*, Springer, 2010, pp. 35–55.

- [10] O. Marques, E. Barenholtz, and V. Charvillat, "Context modeling in computer vision: techniques, implications, and applications," *Multimed. Tools Appl.*, vol. 51, no. 1, pp. 303–339, 2011
- [11] X. Yang, S. Sarraf, and N. Zhang, "Deep learning-based framework for Autism functional MRI image classification," *J. Ark. Acad. Sci.*, vol. 72, no. 1, pp. 47–52, 2018.
- [12] J. C. S. J. Junior, S. R. Musse, and C. R. Jung, "Crowd analysis using computer vision techniques," *IEEE Signal Process. Mag.*, vol. 27, no. 5, pp. 66–77, 2010.
- [13] D. Lu and Q. Weng, "A survey of image classification methods and techniques for improving classification performance," *Int. J. Remote Sens.*, vol. 28, no. 5, pp. 823–870, 2007.
- [14] M. Hammad, Y. Liu, and K. Wang, "Multimodal biometric authentication systems using convolution neural network based on different level fusion of ecg and fingerprint," *IEEE Access*, vol. 7, pp. 26 527–26 542, 2018.
- [15] A. Sarraf, "Binary Image Classification Through an Optimal Topology for Convolutional Neural Networks," *Am. Sci. Res. J. Eng. Technol. Sci.*, vol. 68, no. 1, pp. 181–192, 2020
- [16] S. Sarraf, "EEG-based movement imagery classification using machine learning techniques and Welch's power spectral density estimation," *Am. Sci. Res. J. Eng. Technol. Sci.*, vol. 33, no. 1, pp. 124–145, 2017.
- [17] D. T. Mane and U. V Kulkarni, "A survey on supervised convolutional neural network and its major applications," in *Deep Learning and Neural Networks: Concepts, Methodologies, Tools, and Applications*, IGI Global, 2020, pp. 1058–1071
- [18] S. Sarraf and G. Tofighi, "Classification of alzheimer's disease using fmri data and deep learning convolutional neural networks," *arXiv Prepr. arXiv1603.08631*, 2016.

- [19] S. Yang, P. Luo, C.-C. Loy, and X. Tang, "From facial parts responses to face detection: A deep learning approach," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 3676–3684.
- [20] S. Gidaris and N. Komodakis, "Object detection via a multi-region and semantic segmentation-aware cnn model," in Proceedings of the IEEE international conference on computer vision, 2015, pp. 1134–1142.
- [21] X. Wang, L. Gao, J. Song, and H. Shen, "Beyond frame-level CNN: saliency-aware 3-D CNN with LSTM for video action recognition," IEEE Signal Process. Lett., vol. 24, no. 4, pp. 510–514, 2016.
- [22] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in European conference on computer vision, 2014, pp. 818–833.
- [23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.
- [24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv Prepr. arXiv1409.1556, 2014.
- [25] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.
- [26] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 1251–1258.

[27] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. CoRR, abs/1310.1531, 2013

[28] K. Chatfield, K. Simonyan, A. Vedaldi, and A. Zisserman. Return of the devil in the details: Delving deep into convolutional nets. In British Machine Vision Conference, 2014.

condense sem

ORIGINALITY REPORT

8%

SIMILARITY INDEX

7%

INTERNET SOURCES

7%

PUBLICATIONS

2%

STUDENT PAPERS

PRIMARY SOURCES

1

www.researchgate.net

Internet Source

2%

2

research.aalto.fi

Internet Source

1%

3

leon.bottou.org

Internet Source

1%

4

www.semanticscholar.org

Internet Source

1%

5

docplayer.net

Internet Source

1%

6

Submitted to Hong Kong Baptist University

Student Paper

<1%

7

S. Zagoruyko, N. Komodakis. "Deep compare: A study on using convolutional neural networks to compare image patches", Computer Vision and Image Understanding, 2017

Publication

<1%

8

Mohammed Abdulameer Aljanabi, Zahir M. Hussain, Noor Abd Alrazak Shnain, Song Feng Lu. "Design of a hybrid measure for image similarity: a statistical, algebraic, and information-theoretic approach", European Journal of Remote Sensing, 2019

Publication

<1 %

9

arxiv.org

Internet Source

<1 %

10

www.yumpu.com

Internet Source

<1 %

11

Lecture Notes in Computer Science, 2003.

Publication

<1 %

Exclude quotes

Off

Exclude matches

< 14 words

Exclude bibliography

On