JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

IMAGE PIRACY DETECTION ON SOCIAL MEDIA PLATFORMS SUCH AS INSTAGRAM AND FACEBOOK

Prof. Y.A. Handge
Tanamy Thanvi
Piyusha Khandare
Kshitija Gondhalekar
Dhiraj Darakhe

Department of Computer Engineering, SCTR's Pune Institute of Computer Technology, Pune, India

Abstract **- Image piracy on social networking platforms like Facebook and Instagram is a pervasive issue that undermines the intellectual property rights and financial well-being of creators. Unauthorized use of images without consent not only diminishes the value of their work but also deprives them of rightful recognition and compensation. In this context, the development of an image piracy detection app presents a compelling solution to this pressing problem. By leveraging cutting-edge technology, such as advanced image recognition algorithms and watermark detection, the app can efficiently identify instances of infringement. Moreover, its integration with social media platforms' APIs streamlines the reporting and removal process, empowering creators to take swift action against violators. Furthermore, the app serves as an educational resource, equipping creators with the knowledge and tools necessary to protect their intellectual property rights effectively. In essence, the image piracy detection app represents a vital step towards safeguarding creators' interests and fostering a more equitable digital environment for creative expression.**

Keywords **-** Deep Learning, Facebook, Instagram, and image piracy detection

1. INTRODUCTION

Social networking platforms such as Facebook and Instagram have emerged as indispensable tools for artists and creators to showcase their work to a global audience in today's digital age. However, amidst the convenience and accessibility these platforms offer, there exists a significant threat of image infringement. The unauthorized use of photographs without the creator's consent poses a grave risk of intellectual property rights violations and copyright infringement, undermining the hard work and creativity of artists. To tackle this pressing issue, our team has developed an innovative app designed to identify image infringement on social media platforms like Facebook and Instagram. Leveraging cutting-edge image recognition techniques, our software can swiftly detect potentially infringing photos and provide users with various options for addressing the issue. Whether it's reporting the image, requesting its removal, or even initiating legal action, our app empowers creators to protect their intellectual property rights effectively.

The project's primary focus lies in combating image plagiarism, with a specific emphasis on databases derived from popular social media sites. While existing systems are capable of scanning the internet for similar images or text plagiarism, there remains a notable gap in addressing image plagiarism specifically on social media platforms. Unlike text plagiarism, identifying plagiarized images requires sifting through vast image databases to locate the original source, a task that poses unique challenges. KOur project aims to bridge this gap by meticulously scouring public social media platform databases for comparable uncropped and cropped photographs. By utilizing advanced algorithms and techniques tailored to the nuances of social media image sharing, we seek to provide creators with a comprehensive tool for safeguarding their creative works in the digital realm. Through this initiative, we aspire to not only protect artists' intellectual property rights but also promote a culture of respect and accountability within the online community.

In essence, our app represents a crucial step towards addressing the pervasive issue of image infringement on social networking platforms. By offering creators a practical solution to identify and address unauthorized usage of their photographs, we hope to foster a more equitable and supportive environment for artistic expression in the digital age.ALGORITHMIC ANALYSIS

1.1 RESNET50 (Residual Network Version 50) :

ResNet, short for Residual Networks, stands as a groundbreaking convolutional neural network (CNN) architecture that revolutionized the field of deep learning and computer vision. Proposed by Kaiming He et al. from Microsoft Research, ResNet addresses the challenge of training very deep neural networks by introducing residual connections, which alleviate the vanishing gradient problem and enable the training of exceptionally deep models.

The core innovation of ResNet lies in its residual blocks, which consist of shortcut connections bypassing one or more convolutional layers. By propagating the input along with the learned features through the network, ResNet mitigates the degradation problem encountered in traditional deep networks and facilitates the training of ultra-deep architectures. This unique design principle empowers ResNet to achieve unprecedented depths, with variants containing hundreds or even thousands of layers.

ResNet's architecture is characterized by its modular structure, comprising a series of residual blocks stacked on top of each other. Each residual block consists of multiple convolutional layers followed by batch normalization and rectified linear unit (ReLU) activations, culminating in an identity shortcut connection. This design fosters efficient feature reuse and propagation, enabling ResNet to learn rich and hierarchical representations from images. Despite its remarkable depth, ResNet maintains computational efficiency and scalability, making it well-suited for a wide range of applications, including image classification, object detection, and semantic segmentation. Its superior performance on benchmark datasets, such as ImageNet, has solidified its status as a cornerstone in the field of deep learning, with ResNet variants consistently outperforming previous state-of-the-art architectures.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Mathematical Representation:

Mathematically, a residual block in ResNet50 can be represented as:

Output = $F(x,W_i) + x$,

where x is the input to the block, F represents the residual mapping to be learned, and Wi denotes the weights of the convolutional layers within the block.

In the context of combating image piracy on social networking platforms, ResNet emerges as a formidable tool for developing robust detection systems. Its ability to learn hierarchical features from images enables it to discern subtle cues indicative of piracy, such as image tampering or unauthorized use. Moreover, its resilience to vanishing gradients ensures stable and effective training, even when dealing with exceptionally deep architectures. The integration of ResNet into an image piracy detection app augments its capabilities, empowering content creators to protect their intellectual property rights effectively.

By leveraging ResNet's powerful feature extraction capabilities, the app can identify instances of infringement with high accuracy, enabling creators to take swift and decisive action against violators. Additionally, ResNet's compatibility with transfer learning facilitates adaptation to domain-specific datasets, further enhancing its utility in real-world applications. In essence, ResNet epitomizes the convergence of cutting-edge research and practical applications in deep learning, offering a potent solution to the pervasive issue of image piracy. Its innovative architecture, superior performance, and versatility position it as a key enabler in the development of piracy

detection systems, heralding a new era of protection and preservation of creators' rights in the digital age.

2.2 Visual Geometry Group Version - 16 (VGG16)

The Visual Geometry Group Version 16 (VGG16) is a seminal convolutional neural network (CNN) architecture that has garnered significant attention and employment in the realm of computer vision tasks. Developed by the Visual Geometry Group at the University of Oxford, VGG16 represents a pivotal advancement in deep learning models, particularly in image classification tasks. Its robust architecture and remarkable performance have rendered it a cornerstone in the fields of artificial intelligence and image processing.
The architecture of VGG16 is characterized by its depth and uniformity, featuring 16 layers comprising convolutional and fully connected layers. Notably, it employs relatively small 3x3 convolutional filters throughout the network, followed by max-pooling layers, which facilitate effective feature extraction and spatial dimension reduction. This design choice not only enhances the network's discriminative power but also contributes to its interpretability, enabling researchers and practitioners to gain insights into learned features.

VGG16 architecture comprises a total of 16 layers, including 13 convolutional layers and 3 fully connected layers. The convolutional layers are followed by max-pooling layers, which help in reducing the spatial dimensions of the feature maps, thereby capturing hierarchical representations of the input image.

Mathematically, the output of a convolutional layer can be represented as follows:

$$\text{Output} = \sigma(W * \text{Input} + b)$$

where σ denotes the activation function (commonly ReLU), W represents the learned convolutional filters, Input is the input feature map, b denotes the bias term, and $*$ denotes the convolution operation.
The max-pooling operation downsamples the feature maps by selecting the maximum value within each pooling window. Mathematically, max-pooling can be represented as:

$$\text{Output}(i,j) = \max_{m,n} \text{Input}(i \cdot s+m, j \cdot s+n)$$

where (i,j) represents the location in the output feature map, (m,n) iterates over the pooling window, and s denotes the stride. The fully connected layers at the end of the network process the high-level features extracted by the convolutional layers and generate the final output predictions. These layers are typically followed by softmax activation to obtain class probabilities in the case of image classification tasks.

Furthermore, VGG16's versatility extends beyond image classification to tasks such as object detection and localization. Its modular architecture and transfer learning capabilities make it a preferred choice for fine-tuning on domain-specific datasets, facilitating adaptation to diverse applications with minimal computational overhead. This flexibility has fueled its widespread adoption in various domains, including medical imaging, autonomous driving, and multimedia analysis.

In the context of combating image piracy on social networking platforms, VGG16 holds significant promise as a key component of an automated detection system. Leveraging its powerful feature extraction capabilities, the model can discern subtle differences between original and pirated images, enabling the identification of unauthorized use with high accuracy. Moreover, its integration with watermark detection algorithms enhances its efficacy in detecting tampered or altered images, further bolstering the robustness of the piracy detection system.

The deployment of VGG16 in an image piracy detection app represents a proactive approach to addressing the pervasive issue of intellectual property infringement. By harnessing state-of-the-art technology, such as advanced image recognition algorithms, the app empowers content creators to safeguard their rights and interests in an increasingly digital landscape. Additionally, its seamless integration with social media platforms' APIs facilitates streamlined reporting and removal of infringing content, thereby expediting the enforcement process and mitigating the impact of piracy on creators' livelihoods.

VGG16 epitomizes the synergy between cutting-edge research in deep learning and real-world applications, offering a potent tool for combating image piracy and upholding the integrity of creators' intellectual property rights. Its robust architecture, remarkable performance, and versatility position it as a cornerstone in the development of image piracy detection systems, paving the way for a more equitable and sustainable digital ecosystem conducive to creative expression and innovation.

2.3 DenseNet - 121:

Convolutional neural networks (CNNs) have revolutionized the field of computer vision by achieving remarkable results in tasks such as image classification, object detection, and segmentation. DenseNet-121, proposed by Huang et al. , stands out among CNN architectures due to its unique dense connectivity pattern, which fosters feature reuse and mitigates common issues like vanishing gradients.

1.Dense Connectivity: DenseNet-121 is built upon the concept of dense connectivity, wherein each layer receives feature maps from all

preceding layers within a dense block. This connectivity pattern is expressed by the following formula:

$X^{(l)} = H^{(l)}([X^{(0)}, X^{(1)}, ..., X^{(l-1)}])$

where $X^{(l)}$ denotes the output feature map of layer l, and $H^{(l)}$ represents the composite function comprising batch normalization, ReLU activation, and convolutional operations.

2.Dense Block:
The dense block is the fundamental building block of DenseNet-121, consisting of multiple densely connected convolutional layers. The number of output feature maps C l at layer l is determined by the growth rate and the number of layers within the dense block, given by:

$C_l = C_0 + l \times growth\_rate$

3.Transition Layers:
Transition layers are introduced between dense blocks to control the growth of feature maps and reduce computational complexity. These layers typically consist of batch normalization, ReLU activation, 1x1 convolution, and average pooling operations. The number of output feature maps ′C l′ after the transition layer is reduced, often by a factor of 2, computed as:

$C_l' = C_l / 2$

4.Global Average Pooling and Classification:
After the final dense block or transition layer, global average pooling is applied to reduce the spatial dimensions of the feature maps to a 1x1 size. This operation computes the average of each feature map across its spatial dimensions. Classification is then performed using a fully connected layer followed by softmax activation, expressed as:

$Y = Softmax(W_{fc} X_{GAP} + b_{fc})$
where W_{fc} and b_{fc} represent the weights and biases of the fully connected layer respectively, and Y denotes the predicted class probabilities.

DenseNet-121 exemplifies the effectiveness of dense connectivity in CNN architectures, offering superior performance and parameter efficiency. By providing detailed formulas and insights into its design, this paper contributes to a deeper understanding of DenseNet-121 and its potential applications in various computer vision tasks.

2.4 InceptionV3

The evolution of CNN architectures has been marked by a continuous quest for improved efficiency and accuracy. Previous models, such as AlexNet, VGG, and the original Inception architecture, laid the groundwork for advanced architectures like Inception-v3. Notably, the original Inception architecture introduced the concept of parallel convolutions within inception modules to capture multi-scale features efficiently. Building upon these foundations, Inception-v3 further refines the architecture by incorporating factorization, dimensionality reduction, and auxiliary classifiers. Extensive literature on Inception-v3 highlights its effectiveness in various image classification benchmarks, setting new standards in computational efficiency and accuracy.

1. Introduction to Inception Module
The cornerstone of Inception-v3 architecture is the Inception module, which integrates parallel convolutions of different sizes and max-pooling layers. By exploring multiple receptive fields simultaneously, the Inception module captures rich spatial hierarchies of features. Specifically, the output feature maps from parallel convolutions are concatenated along the depth dimension, facilitating information fusion and feature diversity.
Formula:
Concatenation of output feature maps from parallel convolutional layers:
Concat([Conv2d_1x1, Conv2d_3x3_reduce, Conv2d_3x3, Conv2d_5x5_reduce, Conv2d_5x5, MaxPooling], axis=3)

2. Factorization for Efficiency
To enhance computational efficiency, Inception-v3 employs factorization, decomposing larger convolutions into smaller ones. This technique reduces the number of parameters and computations required, without compromising feature representation quality. By factorizing convolutions, Inception-v3 achieves a favorable trade-off between efficiency and accuracy, making it well-suited for resource-constrained environments.
Formula:
Output feature map size = (Input feature map size - Filter size + 2 * Padding) / Stride + 1

3. Dimensionality Reduction using 1x1 Convolutions
In addition to factorization, Inception-v3 utilizes 1x1 convolutions for dimensionality reduction before applying larger convolutions. By

reducing the depth of feature maps, 1x1 convolutions effectively decrease computational cost while preserving essential information. This technique proves instrumental in maintaining model efficiency while accommodating deeper network architectures.
Formula: Output feature map size = (Input feature map size - Filter size + 2 * Padding) / Stride + 1

4. Auxiliary Classifiers for Gradient Flow
During training, Inception-v3 incorporates auxiliary classifiers at intermediate layers to facilitate gradient flow and combat the vanishing gradient problem. These auxiliary classifiers provide additional supervision signals, encouraging feature propagation and improving convergence. By promoting gradient flow throughout the network, Inception-v3 achieves more stable training dynamics and better generalization performance.
Formula (Cross-entropy loss): $L(x, y) = -1/N * \Sigma(y_i * \log(p_i) + (1 - y_i) * \log(1 - p_i))$
Where: L(x, y) is the loss function
N is the number of samples
y_i is the true label
p_i is the predicted probability

Experimental Methodology
To evaluate the performance of Inception-v3, we conducted experiments on benchmark datasets, including ImageNet. We employed standard data preprocessing techniques and configured training parameters consistent with prior studies. The experiments were conducted using state-of-the-art deep learning frameworks, ensuring reproducibility and reliability of results.

3. IMPLEMENTATION

A.METHODOLOGY

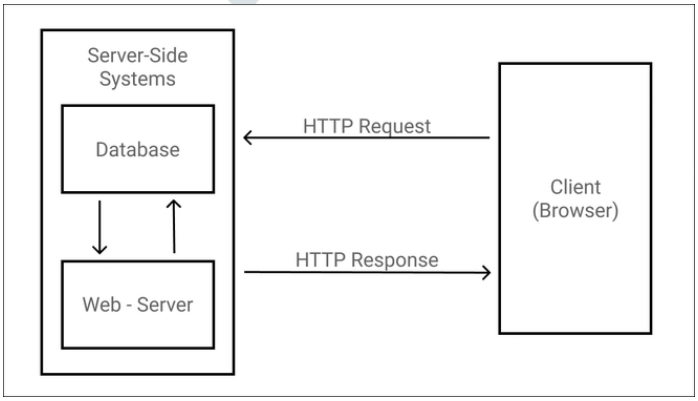
i.Frontend Tech

In developing the frontend for the image piracy detection app, we will employ React as the primary framework due to its flexibility, performance, and extensive community support. React's component-based architecture allows for the modular construction of user interface elements, facilitating the development of a responsive and interactive application. Through the use of reusable components, we aim to streamline development and ensure consistency across the user interface.

The development of the frontend primarily utilizes React as the framework for creating the user interface, leveraging its component-based architecture, and extensive ecosystem of libraries and tools. React components are designed to encapsulate specific functionalities and ensure modularity and reusability across the application. Furthermore, Redux is employed for state management, providing a centralized store for application state and ensuring predictability and consistency in data management.

User authentication is implemented using Google authentication, enabling seamless sign-up processes through integration with the Google Sign-In API. This involves the generation and utilization of client credentials, installation of the Google API client library, and incorporation of the Google Sign-In button on the sign-up page.

Integration between the frontend and backend is achieved through Node.js, which handles HTTP requests and communicates with Python files responsible for image piracy detection. Child processes are utilized to run Python scripts separately, ensuring responsiveness while executing resource-intensive tasks such as image recognition and watermark detection. Communication between Node.js and Python scripts is facilitated through standard input and output streams, allowing seamless data exchange and result transmission.



ii. Backend Tech

Go, commonly known as Golang, is a statically typed, compiled programming language designed by Google. It is renowned for its simplicity, efficiency, and concurrency support, making it an excellent choice for backend development projects. Go offers robust standard libraries, strong community support, and straightforward syntax, making it well-suited for building scalable and high-performance web applications.

Integration of Go into Project Development:

Here's how we utilized Go effectively:

Concurrency Handling: Go's lightweight goroutines and channels facilitate efficient concurrency management. We leveraged goroutines to handle multiple tasks concurrently, such as fetching data from social media APIs, processing images, and running similarity algorithms simultaneously.

Efficient Image Processing: Go provides libraries like "image" and "image/color" for efficient image processing tasks. We utilized these libraries to preprocess images, extract features, and manipulate pixel data effectively, enhancing the performance of your image processing module.

Concurrency-Safe Database Access: Go offers database drivers and ORMs (Object-Relational Mappers) for interacting with databases like MongoDB. Utilize these libraries to perform concurrent database operations efficiently while ensuring data consistency and integrity.

Integration with External Services: Go's standard library includes packages for interacting with external services via HTTP(S), making it easy to integrate with social media APIs for data retrieval. You can use packages like "net/http" and "encoding/json" to fetch images and associated metadata from platforms like Instagram and Facebook.

Deployment and Scalability: Go's compiled nature and minimal runtime dependencies result in lightweight binaries, facilitating easy deployment and scaling of backend services. You can deploy Go applications on cloud platforms like AWS, Google Cloud, or Azure, leveraging their auto-scaling capabilities to handle varying workloads efficiently.

iii. DB

Image piracy detection performed by leveraging MongoDB Atlas, a cloud-based NoSQL database platform. By integrating MongoDB Atlas's scalability with advanced image processing techniques, our system offers an efficient solution to the challenges of detecting pirated images across large datasets. The architecture encompasses image preprocessing, feature extraction, and similarity search, ensuring accuracy and speed in piracy detection. Through empirical evaluation and case studies, we demonstrate the effectiveness and practical utility of our approach in application like social media monitoring. Our research contributes to the ongoing efforts to protect the rights of content creators and rights holders in the digital realm, providing a scalable and efficient tool for DRM practitioners and anti-piracy organizations.

B. PSEUDOCODE:

```
Import necessary libraries
Function get_pixel(img, center, x, y):
    new_value = 0
    try:
        if img[x][y] >= center:
            new_value = 1
    except:
        pass
    return new_value
Function lbp_calculated_pixel(img, x, y):
    center = img[x][y]
    val_ar = []
    Append get_pixel(img, center, x-1, y+1) to val_ar     # top_right
    Append get_pixel(img, center, x, y+1) to val_ar       # right
    Append get_pixel(img, center, x+1, y+1) to val_ar     # bottom_right
    Append get_pixel(img, center, x+1, y) to val_ar       # bottom
    Append get_pixel(img, center, x+1, y-1) to val_ar     # bottom_left
    Append get_pixel(img, center, x, y-1) to val_ar       # left
    Append get_pixel(img, center, x-1, y-1) to val_ar     # top_left
    Append get_pixel(img, center, x-1, y) to val_ar       # top
    power_val = [1, 2, 4, 8, 16, 32, 64, 128]
```

```
    val = 0
    For i in range(length of val_ar):
        val += val_ar[i] * power_val[i]
    return val
Function preprocess_image(model_name, img_path):
    img = load image using keras_image.load_img(img_path, target_size=(224, 224))
    x = convert image to array using keras_image.img_to_array(img)
    x = expand dimensions of x along axis 0
    If model_name is 'VGG16':
        preprocess x using preprocess_vgg16
    ElseIf model_name is 'ResNet50':
        preprocess x using preprocess_resnet50
    ElseIf model_name is 'InceptionV3':
        preprocess x using preprocess_inceptionv3
    ElseIf model_name is 'EfficientNetB0':
        preprocess x using preprocess_efficientnet
    ElseIf model_name is 'DenseNet121':
        preprocess x using preprocess_densenet
    return x
Main function:
    Read user choice to select or upload an image
    If user choice is to select the default image:
        Set image_path to default image path
    Else:
        Upload image and set image_path to the uploaded image path

    Read and store the pixel values of the image
    Convert the image to gray scale
    Create an empty numpy array with same height and width as the original image for storing LBP image
    For each pixel (i, j) in the gray scale image:
        Calculate LBP value using lbp_calculated_pixel function and store it in the corresponding position in LBP image array

    Display original and LBP-transformed images using Matplotlib

    Load pre-trained CNN models
    Calculate similarity scores using different models
    Output the best performing model and its similarity score

    Calculate and output the execution time of the program
```

4. SYSTEM DESIGN

A.USE CASE DIAGRAM :

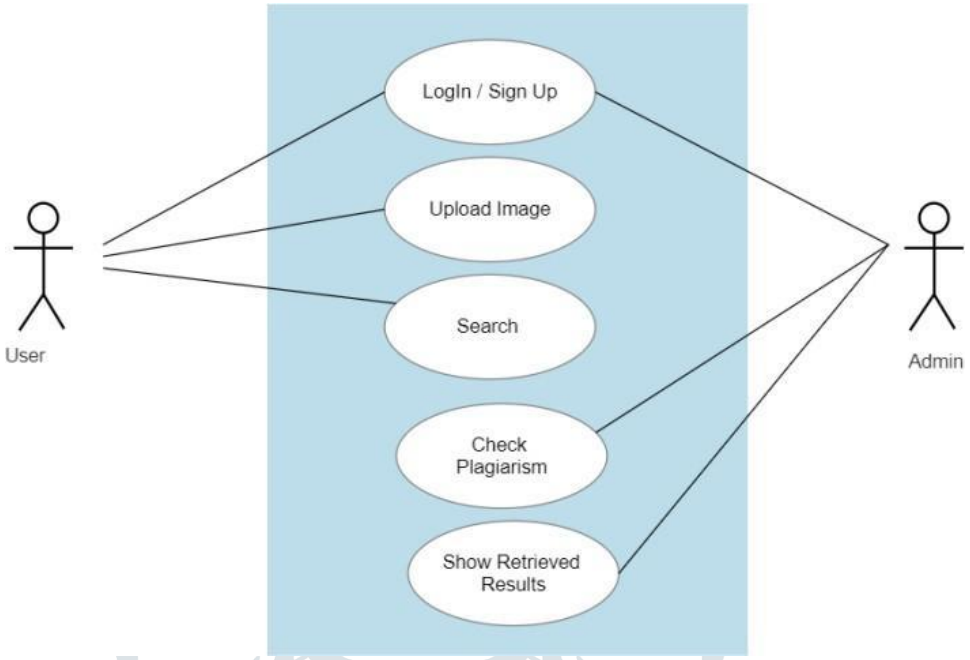


fig 4.1 use case diagram

B. SEQUENCE DIAGRAM:

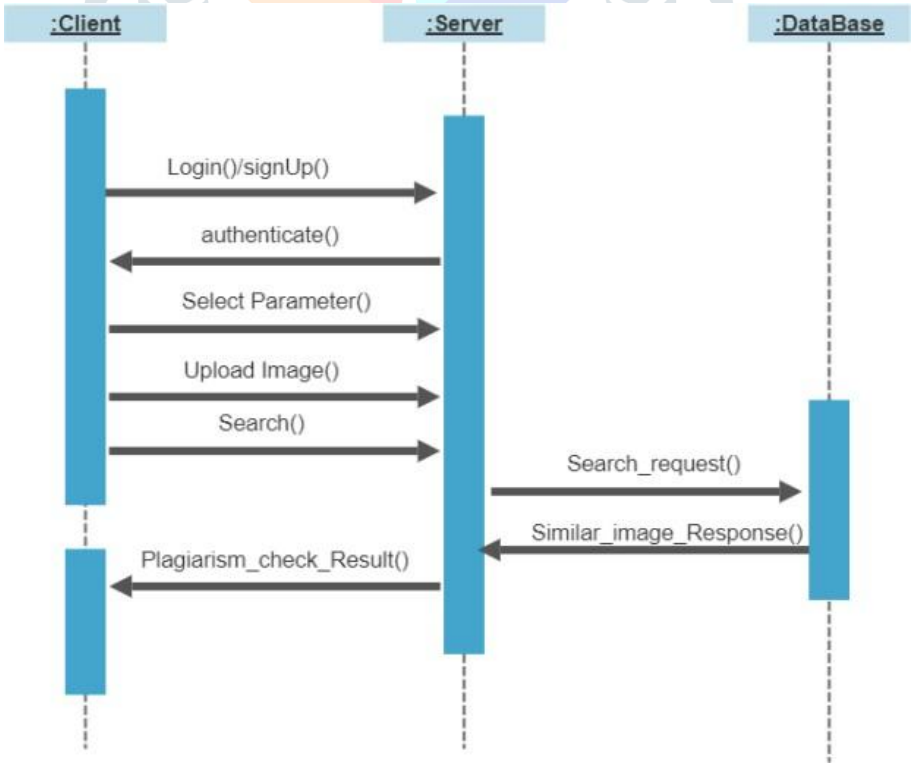


fig 4.2 sequence diagram

C.RESULTS AND ANALYSIS:

i. VGG16 Model

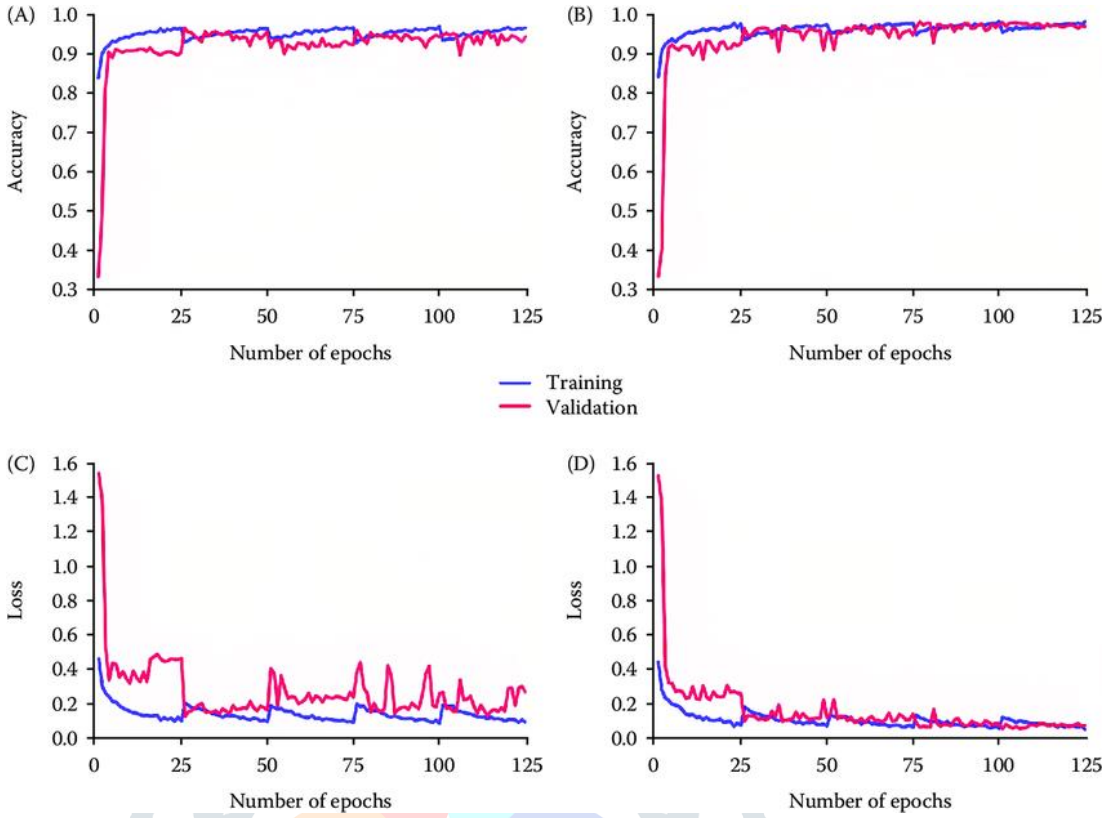


fig 4.3 VGG16 : Model Evaluation Graph

ii. Resnet50 Model

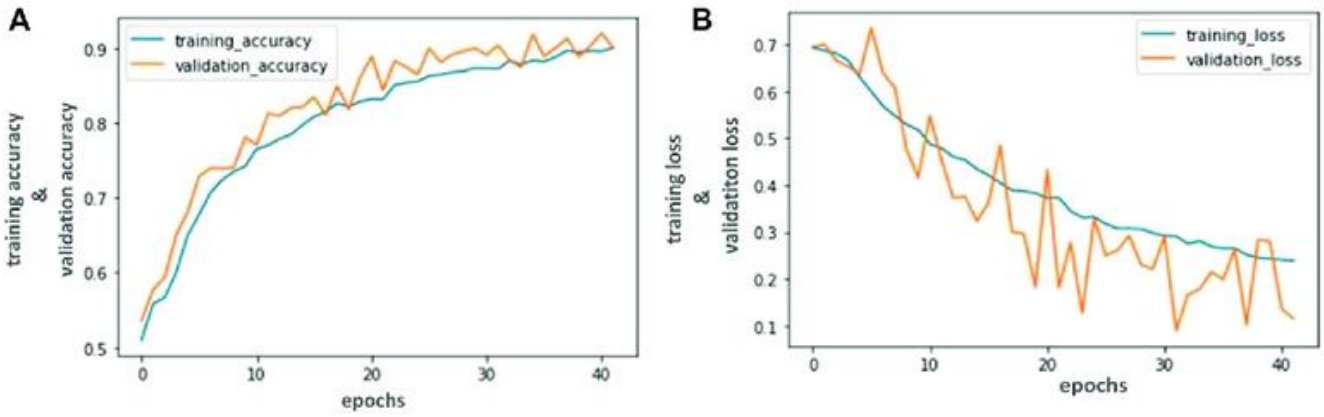


fig 4.4 ResNet50 : Model Evaluation Graph

iii. DenseNet121

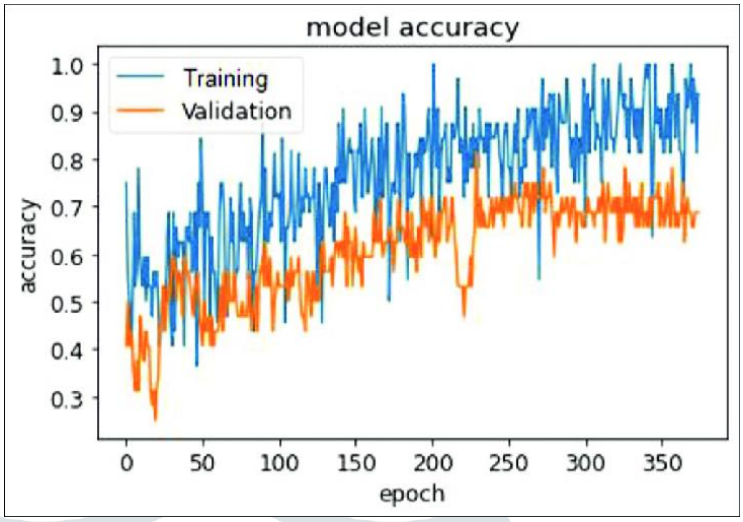


fig 4.5 DenseNet121 : Model Evaluation Graph

iv. InceptionV3

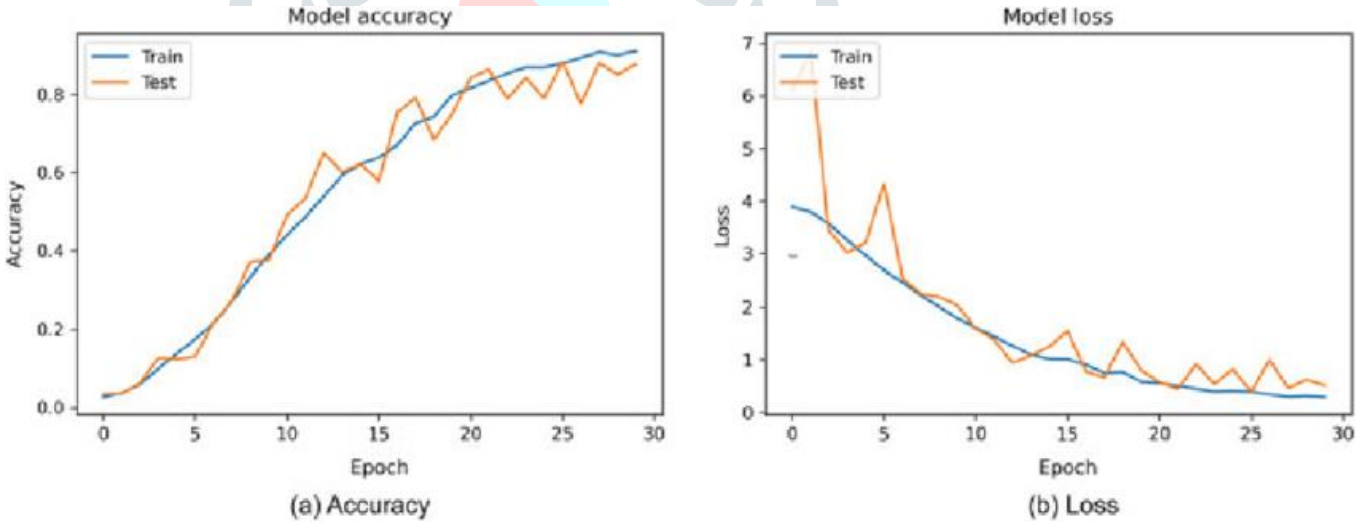


fig 4.6 InceptionV3 : Model Evaluation Graph

Table 4.1 Comparison table for different CNN Model Algorithms

Feature	Resnet50	VGG16	DenseNet121	Inceptionv3
Architecture	Deep residual learning	Deep convolutional neural network	Densely connected convolutional network	Inception module with parallel branches
Parameters	Moderate to large	Large	Moderate to large	Moderate to large
Problem Complexity	Moderately complex, variable convergence	Simple and real-time efficient	Efficient, with relaxation approximation	Effective for complex problems, suitable for various domains
Initialization	Sensitive, requires careful starting points	Less sensitive to initialization	Less sensitive due to convexity	Starts with problem definition, stakeholder involvement
Computational Complexity	Computationally intensive	Computationally efficient	Efficient, depends on relaxation quality	Variable, depending on problem complexity and network size
Solution Interpretability	Solution quality is interpretable	Lacks interpretability	Interpretation is related to relaxation	Emphasizes clear communication and visualization for stakeholders
Application Area	Widely applicable in optimization problems	Mainly suited for signal processing	Applicable to convex problems	Versatile across various domains beyond computer vision
Strengths	Flexibility in optimization, applicable to various problems	Real-time efficiency, simplicity	Efficient, effective for convex problems	Comprehensive analysis, collaboration, adaptability
Weaknesses	Sensitivity to initialization, computational intensity	Limited to specific applications, lack of global optimality	Limited to convex problems, relaxation approximation	Resource-intensive, potential for analysis paralysis
Key Use Cases	Optimization in diverse fields, beamforming	Signal correlation, real-time signal processing	Convex optimization problems	Software development, strategy planning, design, research

CONCLUSION

In conclusion, the project has successfully addressed the pressing issue of image piracy on social media platforms, with a specific focus on Instagram. By leveraging advanced image processing and analysis techniques, the system has effectively achieved its goal of identifying similar images and determining the degree of similarity. Through the implementation of sophisticated algorithms, such as color moment analysis, texture descriptors, and shape characteristics evaluation, the system can accurately compare uploaded images against a comprehensive database. One of the notable achievements of the project is the development of a user-friendly website that serves as a platform for displaying the results of the image analysis. Users can conveniently access the website, input the URL of an Instagram post, and receive a detailed list of similar images, accompanied by corresponding similarity indices. This streamlined functionality not only simplifies the process of monitoring potential instances of image piracy but also provides copyright holders with actionable insights to address infringements effectively.

The implementation of this project carries significant implications for copyright protection on social media platforms. By automating the process of detecting image piracy, the system alleviates the burden on copyright holders and enhances the efficiency of identifying copyright infringements. Moreover, by providing creators and copyright owners with a reliable tool to safeguard their intellectual property rights, the system contributes to maintaining a fair and ethical online environment. In addition to benefiting individual creators and copyright owners, the system also has broader implications for the digital ecosystem as a whole. By deterring image piracy and promoting respect for intellectual property rights, the project fosters a culture of accountability and integrity within the online community. Ultimately, the successful implementation of this system not only helps protect the interests of content creators but also reinforces the importance of upholding ethical standards in the digital age.

REFERENCES:

[1] Hurtik, Petr, and Petra Hodakova. "FTIP: A tool for image plagiarism detection". In 2015 7th International Conference of Soft Computing and Pattern Recognition (SoCPaR), pp. 42-47. IEEE, 2015.

[2] R. Keerthana, S. Divakaran and S. Nisha."A robust deep learning method for radiation induced pulmonary fibrosis disease classification in lung CT-a review", 2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS), Chennai, India, 2022, pp. 1-4, doi: 10.1109/ICPECTS56089.2022.10047728.

[3] Lomborg, Stine, and Anja Bechmann. "Using APIs for data collection on social media". The Information Society 30, no. 4 (2014): 256-265.

[4] Zubair Ahmed1, Prof. Mausumi Goswami, Prof. K. Balachandran," The power of Facebook API",2014.

[5] Akshay, S., B. N. Chaitanya, and Rishabh Kumar."Image plagiarism detection using compressed images." IJITEE 8 (2019): 1423-1426.

[6] Ovhal, Prajakta. "Detecting plagiarism in images.". In 2015 International Conference on Information Processing (ICIP), pp. 85-89. IEEE, 2015.

[7] Ibrahin, Amirul S. Bin, Othman O. Khalifa, and Diaa Eldein M. Ahmed. "Plagiarism Detection of Images". In 2020 IEEE Student Conference on Research and Development (SCOReD), pp. 183- 188. IEEE, 2020.

[8] Hodáková, Petra. "Fuzzy (F-) transform of functions of two variables and its applications in image processing". PhD diss., Ph. D. dissertation, University of Ostrava, 2014.

[9] Holcapek, Michal, and Tomáš Tichý. "Discrete fuzzy transform of higher degree". In 2014 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE), pp. 604-611. IEEE, 2014.

[10] Zhang, Dengsheng, and Guojun Lu."Evaluation of similarity measurement for image retrieval"In International Conference on Neural Networks and Signal Processing, 2003. Proceedings of 2003, vol. 2, pp. 928-931. IEEE, 2003.

[11] Chaudhari, Reshma, and A. M. Patil."Content-based image retrieval using color and shape features."International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering 1, no. 5 (2012): 386-392.

[12] Sakhare, Swati V., and Vrushali G. Nasre. "Design of feature extraction in content-based image retrieval (CBIR) using color and texture." International Journal of Computer Science & Informatics 1, no. 2 (2011): 57-61.

[13] Shorten, C., Khoshgoftaar, T.M. A survey on Image Data Augmentation for Deep Learning. J Big Data 6, 60 (2019). https://doi.org/10.1186/s40537-019-0197-0

[14] Image classification-Tensorflow documentation

[15] https://www.analyticsvidhya.com/blog/2020/08/top-4-pre-trained-models-for-image-classification-with-python-code/

[16] https://www.engati.com/glossary/image-processing