

A Decision Support System to Identify Spam Images Using Ensemble Convolutional Neural Network

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

By

VINAYANA GINJUPALLI (Reg.No: 40110379)

AJAY KUMAR GORLI (Reg.No: 40110400)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Category-1 University by UGC

**Accredited with Grade “A++” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHI SALAI,
CHENNAI – 600119**

APRIL - 2024



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Category - I University by UGC

Accredited "A++" by NAAC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **AJAY KUMAR GORLI (40110400)**, **VINAYANA GINJUALI (40110379)** who have done the Project work as a team entitled **A DECISION SUPPORT SYSTEM TO IDENTIFY SPAM IMAGES USING CONVOLUTIONAL NEURAL NETWORK** under my supervision from November 2023 to April 2024.


Internal Guide 12/4/2024

Ms. R. Anandha Sree, M.E.


Head of the Department
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SATHYABAMA
Dr. L. LAKSHMANAN, M.E., Ph.D.
(DEEMED TO BE UNIVERSITY)
Jeppiaar Nagar, Rajiv Gandhi Salai
Chennai - 600 119.

Submitted for Viva voce Examination held on

13/04/24


Internal Examiner 13/04/24


External Examiner 13/4/24

DECLARATION

I, AJAY KUMAR GORLI (40110400) hereby declare that the Project Report entitled A DECISION SUPPORT SYSTEM TO IDENTIFY IMAGE SPAM DETECTION USING CONVOLUTIONAL NEURAL NETWORK done by me under the guidance of **Ms. R. Anandha Sree, M.E.**, is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in **Computer Science and Engineering**.

DATE: 13/4/2024

PLACE: Chennai

SIGNATURE OF CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA Institute of Science and Technology** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala M.E., Ph. D., Dean**, School of Computing, **Dr. L. Lakshmanan M.E., Ph.D.**, Head of the Department of **Computer Science and Engineering** for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Ms. R. Anandha Sree M.E.** for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

This abstract discusses that people have been hiding harmful content in pictures and sending them via emails, causing a lot of trouble. While we have had some success in spotting spammy text using technology that recognizes characters, dealing with spam in images is tougher. Many smart people have tried different computer programs and techniques to create a good system to recognize image spam, but so far, none have been perfect.

In this project, we are looking at three main ways to deal with unwanted pictures. We are using smart computer programs and fancy math to look at features like colors and shapes. Specifically, we are trying out two types of smart computer programs: Deep Convolutional Neural Networks (DCNN) and pre-trained models from the Internet like Exception and ResNet50. We trained these programs using three different sets of spam pictures to make them better at catching bad images.

By doing this, we hope to learn which method works best for stopping unwanted pictures. Our goal is to figure out if these methods can be combined to make a good system for stopping bad images. At the end of the project, we found that using a mix of these methods, including DCNN, Exception, and ResNet50, gave us the best results in catching image spam.

TABLE OF CONTENTS

Chapter No	Title	Page No
	ABSTRACT	v
	LIST OF FIGURES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
	1.1 Background and Motivation	2
	1.2 Problem Statement	3
	1.3 Objectives and Scope	3
	1.4 Applications of Image Spam Detection	3
	1.5 Organization of Thesis	4
2	LITERATURE SURVEY	5
	2.1 Overview of Existing Research and Literature	5
	2.2 Inference from Literature Survey	6
	2.3 Open Problems in Existing System	7
3	REQUIREMENTS ANALYSIS	8
	3.1 Aim of the Project	8
	3.2 Feasibility Studies of the Project	8
	3.2.1 Technical Feasibility	8
	3.3 Requirements Specification	9
	3.3.1 Hardware Requirements	9
	3.3.2 Software Requirements	9
	3.3.3 Python	10

	3.3.4	Deep Learning	10
	3.3.5	Convolutional Neural Network	11
	3.3.6	Tensor Flow	12
	3.3.7	Keras	12
4		DESCRIPTION OF PROPOSED SYSTEM	13
	4.1	Description of Research Approach and Methods	14
	4.1.1	Data Collection and Pre-processing	14
	4.1.2	Neural Networks	14
	4.1.3	Deep Convolutional Network Model	15
	4.1.4	Cost Sensitive and Transfer Training	15
	4.1.5	Statistical Metrics	16
	4.2	Description of Software for Implementation and Testing Plan of the Proposed System	17
	4.2.1	Visual Studio Code	17
	4.2.2	Backend	18
	4.2.3	Solidity	19
	4.2.4	Declarative	19
	4.2.5	Components	20
	4.3	Project Management Plans	21
5		IMPLEMENTATION DETAILS	23
	5.1	Development and Deployment Setup	23
	5.1.1	Development Setup	23
	5.1.2	Development Workflow	23
	5.1.3	Documentation	24

	5.2	Algorithms	25
	5.2.1	Convolutional Neural Networks	25
	5.2.2	Backpropagation	25
	5.2.3	SoftMax Regression	26
	5.3	Testing	26
6		OUTCOMES AND DISCUSSIONS	29
	6.1	Results	29
	6.2	Discussion	30
7		RESULTS AND DISCUSSIONS	32
8		CONCLUSION	33
	8.1	Conclusion	33
	8.2	Recommendations for Future Research	34
	8.3	Research Issues	35
	8.4	Future Work	35
		REFERENCES	37
		APPENDIX	39
	A	SOURCE CODE	39
	B	SCREENSHOTS	45
	C	RESEARCH PAPER	50
	D	CERTIFICATES	57

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
3.3.4.1	Structure of Artificial Neural Network	11
3.3.5.1	Structure of Convolutional Neural Network	11
4.1	Spam and ham Image Pictures	13
4.1.2	Architecture of Neural Network	14
4.1.3.1	Work flow of CNN	15
4.1.4.1	Trainable and Non-Trainable Parameters	15
4.1.5.1	Statistical Methods	16
4.1.5.2	Flow Diagram	17
B.1	Importing Required Libraries	45
B.2	Pre-processing the data	45
B.3	Extracting the Frames	46
B.4	Creating dataset with required frames	46
B.5	Calculating parameters	47
B.6	Successfully Maxpooled and falltend	47
B.7	Plotting the Model	48
B.8	Evaluating the Model	48
B.9	Accuracy of concatenated models	49
B.10	Evaluating the Model	49

List of Abbreviations

Abbreviations	Expansion
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Networks
VGG	Visual Geometry Group
FNR	False Negative Role
FPR	False Positive Rate
TP	True Positive
FN	False Negative
FP	False Positive
TN	True Negative

CHAPTER 1

INTRODUCTION

In today's digital age, the Internet has become an essential part of our lives, connecting us with the world and enabling various activities, from online shopping to communication. But along with its convenience, the internet also poses risks, especially from cybercriminals. These malicious individuals use tactics like phishing, spamming, and impersonation to exploit internet users. Email spam is a significant issue, constituting a staggering 60% of emails in sectors like mining, finance, insurance, and real estate. To tackle this problem, spam filters are crucial to ensuring safe email communication.

The stakes are high in the battle against spam. Not only does it flood our inboxes with irrelevant and dangerous content, but it also serves as a tool for cybercriminals to spread Trojans, rats, viruses, and other harmful software. These attackers craft spam emails, images, and messages laden with malicious attachments and URLs, aiming to steal identities, commit financial fraud, and breach sensitive data.

Traditionally, email spam was mostly in text form. However, cyber attackers have evolved their methods, introducing image spam to bypass conventional text-based filters. Image spam involves hiding text within images, making it difficult for regular filters to detect. These deceptive images trick users into clicking, leading to unsafe websites or malware infections. To combat this, machine learning (ML) models like Support Vector Machines (SVM), K- Nearest Neighbors (KNN), and Naïve Bayes (NB) have been developed, boasting an impressive 95% accuracy in filtering out spam. But now we want to make some more advance models and want to know if they can perform more better than other models.

In this context, this study dives into the challenges faced by existing spam filters, exploring their impact on efficiency. It conducts a thorough comparison of machine learning techniques, shedding light on the role of machine learning in spam detection. By understanding these challenges and exploring advanced techniques, we aim to enhance our defense against the ever- evolving tactics of cyber attackers, making the

internet a safer place for all users.

This system aims to the fusion or concatenation of VGG16 and ResNet50 models, leveraging their individual characteristics to enhance performance in image classification tasks. By combining the feature extraction capabilities of VGG16 with the residual learning mechanism of ResNet 50 we intend to create a hybrid model that captures the both fine grained cells and high-level features in images.

With the creation of potent convolutional neural network (CNN) designs like VGG16 and ResNet50, deep learning in computer vision has made significant strides. The ability of these models to extract complex characteristics and develop hierarchical representations from visual data has revolutionized picture classification jobs. In this work, we set out on a path similar to that of a conscientious student with the goal of concatenating the strengths of VGG16 and ResNet50.

This is an attempt to investigate the possibility of developing a more reliable and accurate picture classification model by fusing the complex residual learning process of ResNet50 with the extensive feature extraction capabilities of VGG16. By means of thorough testing and documentation.

The proposed work aims to utilize blockchain technology to track food products from their source to consumption, with the ultimate goal of enhancing transparency and food safety. Blockchain, a decentralized and immutable digital ledger, offers a unique solution to the challenges faced in the food industry, such as inconsistent or incomplete information sharing, difficulty in verifying the authenticity of products, and inefficient traceability systems. By using blockchain, every step in the supply chain process can be recorded, verified, and shared among authorized participants in a secure and transparent manner.

1.1 BACKGROUND AND MOTIVATION

To address cybersecurity concerns by utilizing state-of-the-art deep learning techniques, the project involving the concatenation of VGG16 and ResNet50 architectures for visual spam categorization has a human background and motivation. The need of creating strong spam detection systems to safeguard users, improve user

experience, and provide a more secure and reliable digital environment is highlighted by the growing frequency of spam and dangerous information on the internet. Participating in such projects also fosters technological developments, highlights ethical considerations in technology development and deployment.

1.2 PROBLEM STATEMENT

"In the digital landscape, the proliferation of spam content, including spam images, poses significant cybersecurity threats and degrades user experience. Traditional spam detection methods often struggle to effectively distinguish between legitimate and malicious content, leading to potential risks such as phishing attacks and malware dissemination. Therefore, there is a pressing need for advanced spam detection solutions that leverage state-of-the-art deep learning architectures like VGG16 and ResNet50 to accurately classify and filter out spam images, thereby enhancing digital security, user trust, and overall online ecosystem integrity."

1.3 OBJECTIVE AND SCOPE

Objective:

The primary objective of this project is to develop a deep learning system capable of detecting image spam with high accuracy.

Scope:

The scope of this project encompasses the data collection, preprocessing, model design, training, evaluation, and potentially deployment of the trained model for real-time spam detection. Key aspects within the scope include fine-tuning hyperparameters, optimizing model performance metrics, addressing ethical considerations, documenting methodologies, and providing insights into the effectiveness of model concatenation techniques in improving spam classification accuracy and robustness.

1.4 APPLICATIONS OF IMAGE SPAM DETECTION

Image spam detection has several important applications across various domains:

- **Email Filtering:** Image spam detection can be applied to filter out spam emails

containing image-based advertisements, phishing attempts, or malicious content.

- **Social Media Content Moderation:** Social media platforms can use image spam detection techniques to automatically detect and remove spammy or inappropriate images from user-generated content.
- **Online Advertising Fraud Prevention:** In the context of online advertising, image spam detection can help prevent ad fraud by identifying and blocking ads that use misleading or deceptive images to attract clicks or engagement.
- **E-commerce Security:** E-commerce platforms can utilize image spam detection to detect and block spammy product images or advertisements that violate platform policies.
- **Content Filtering in Forums and Communities:** Image spam detection techniques can be integrated into online forums, communities, and discussion platforms to automatically filter out spammy or irrelevant images posted by users.
- **Mobile App Security:** Mobile applications can leverage image spam detection to detect and block spammy or malicious images uploaded by users.

1.5 ORGANIZATION OF THESIS

The organization includes an introduction that provides an overview of the research problem, objectives, and significance. The literature review section discusses the existing research on Image spam detection using convolutional neural network and the Limitations of the existing system. The methodology section describes the data collection process, pre-processing, and the deep learning models used. The results section presents the findings of the study, including the accuracy of the models and the comparison between VGG16 and ResNet50. The discussion interprets the results, discusses the limitations of the study, and suggests research directions.

CHAPTER 2

LITERATURE SURVEY

Several studies on image spam detection systems have been conducted in the recent years. This section summarizes all the previous work on image spam detection systems.

2.1 OVERVIEW OF EXISTING RESEARCH AND LITERATURE

This problem statement has been extensively studied over the past 5 years by researchers and automotive companies in a bid to create a solution.

- Gao et al. (Yan Gao et al., 2008) propose an image spam detection scheme that relies on a probabilistic boosting tree algorithm. These authors obtain an accuracy of 0.8944.
- Kumaresan et al. (Kumaresan, Sanjushree, & Palanisamy, 2014) propose a technique for detecting image spam based on colour features, and using the nearest neighbour (KNN) algorithm. Specifically, the authors rely on the RGB and HSV histograms as features. In this research, a straightforward KNN classifier yields an accuracy of 0.945.
- In the research by Annadatha et al. (Annadatha & Stamp, 2018), support vector machines (SVM) have been applied to a set of 21 image features. Using feature selection based on linear SVM weights, the authors can achieve an accuracy rate of 0.97 with a relatively small set of features.
- Chavda et al. (Chavda et al., 2018) conduct two sets of experiments with SVM and image processing. The authors use an extensive set of 41 image features and they achieve 0.97 and 0.98 accuracy on two publicly available datasets.
- Aiwan et al. (Aiwan & Zhaofeng, 2018) propose an image spam filtering method based on convolutional neural network (CNN). The proposed system uses data augmentation and achieves an accuracy improvement, as compared to selected examples of previous work. Kumar et al. (Kumar,R, & KP, 2018) apply deep learning techniques image spam problem. They obtain an accuracy about 91.

- Chao Wang proposed a method that collected the low-level features of images making the classification process faster. The SVM model proposed by him also proved to get 95% spam detection from the images.
- In another instance Ajay Pal Singh had proposed four CNNs alongside the pretrained VGG19 model to detect spam on the current dataset and an improved. He achieved an accuracy of 95-98.78 % for the various datasets like ISH, Improved Dataset and the Dredze dataset taken in respective permutations.
- Shikar Seth and Sagar Biswas proposed multi-modal architectures which were a combination of image and text classifiers. The proposed models had a significant difference in classification accuracy than the basic CNN which were 96.87% on feature fusion and 98.11% on learned rule.
- In another scenario Ahmad Mahdi Salih proposed a CNN based image spam classification using color models evaluated on a public image dataset. From the proposed models the XYZ color model proved to achieve an accuracy of 98.4%.
- Basically, spams are categorized into Text-based and Image- based. Spams are further divided into content-based spam and non- content-based spam. Content-based spam is the first-generation image spam. This includes the spam in emails in text based form.

2.2 INFERENCES FROM LITERATURE SURVEY

The interfaces from the existing system for image spam detection using concatenated VGG16 and ResNet50 architectures include a data input interface for uploading images, a preprocessing interface for standardizing image data, a model integration interface for integrating the concatenated model, a classification output interface for displaying detection results, a feedback interface for user input and corrections, a reporting interface for performance metrics.

2.3 OPEN PROBLEMS IN EXISTING SYSTEM

While image spam detection has shown great promise in recent years, there are still

several open problems in the existing system that need to be addressed.

I. Choosing the Right Features:

Picking the correct email features is vital. While headers and subjects' matter, manual selection of features is crucial for accuracy in spam detection.

II. Importance of Smart Preparation:

Properly preparing data and using smart features is key. It makes spam filters work better and respond faster, ensuring they can adapt and learn on their own.

III. Considering Time Efficiency:

Besides accuracy, we should care about how quickly the system works. Fast and efficient systems are important, especially in real-time situations.

IV. Smart Updates for Deep Learning:

Deep learning systems need to be smart. They should be able to change and update how they recognize spam as new methods emerge. Keeping up-to-date ensures better spam detection.

V. Prioritizing System Security:

Keeping the spam filter safe is crucial. A secure system not only catches more spam but also ensures reliable and trustworthy results.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 AIM

The purpose of this study is to examine the benefits of combining the ResNet50 and VGG16 architectures for picture categorization. The main goal is to develop a concatenation model that can extract a wide range of characteristics from photos, including high-level semantics and minute details. In comparison to the separate VGG16 and ResNet50 models, we hope to improve the model's performance metrics—accuracy, precision, recall, and F1 score—through this fusion. Furthermore, we want to investigate the idea of knowledge transfer between different structures, making use of previously learned information to accelerate learning and adaption to new datasets. In addition, we want to learn more about the concatenated model's interpretability and how the arrangement of its features affects its decision-making.

3.2 FEASIBILITY STUDY

The feasibility of the project is server performance increase in this phase study evaluates several factors to ascertain the project's viability and potential obstacles for the concatenation of VGG16 and ResNet50 architectures in image classification. For the feasibility study, keep the following points in mind. For feasibility analysis, some understanding of the major requirements for the system is essential.

3.2.1 Technical Feasibility:

Evaluate the technical feasibility of integrating VGG16 and ResNet50 architectures through concatenation. This includes ensuring compatibility between the layers, addressing any architectural differences, and implementing the fusion mechanism effectively using deep learning frameworks like TensorFlow or PyTorch.

- **Resource Requirements:**

Determine the amount of computing power needed to train and optimize the combined model

- **Data Availability:**

Make that the datasets include enough labeled data, span a variety of image categories, and support the project's goals.

- **Model Performance:**

To determine whether the concatenated VGG16-ResNet50. Assess the model's efficacy using metrics including accuracy, precision, recall, and F1 score.

Training Time and Convergence:

Compute the concatenated model's training time and convergence behaviour. In order to maximize training stability and efficiency, take into account variables including batch size, learning rate scheduling, and optimization strategies.

Implementation Complexity:

Consider code development, debugging, and model optimization when assessing how difficult it is to apply the concatenation strategy.

Ethical and Legal Considerations:

Make sure that all data consumption, model deployment, and potential biases in picture classification jobs are in accordance with ethical standards and applicable laws.

3.3 REQUIREMENTS SPECIFICATION

Recommended Practice for Software Requirements, The Hardware and Software Requirements are described

3.3.1 Hardware Requirements

- **RAM:** 4GB or above
- **Processor of Frequency:** 1.5GHz or above
- **Processor:** Intel Pentium 4 or higher

3.3.2 Software Requirements

- **Operating System:** Windows 8 and above
- **Languages:** Python
- **Tools used:** Visual Studio Code, Jupyter Notebook

3.3.3 Python

Python is a scripting language that is high-level, interpreted, interactive, and object-oriented. Python is intended to be extremely readable. It commonly employs English terms rather than punctuation, and it has fewer syntactical structures than other languages.

- **Easy-to-learn** - Python has a small number of keywords, a basic structure, and a well-defined syntax. This enables the pupil to swiftly learn the language.
- **Easy-to-read** - Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** - Python's source code is relatively simple to maintain.
- **A broad standard library** - The majority of Python's library is portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode** - Python features an interactive mode that enables for interactive testing and debugging of code snippets.
- **Portable** - Python can operate on a broad range of hardware devices and has the same interface across them all.
- **Extendable** - The Python interpreter may be extended using low-level modules. These modules allow programmers to enhance or adapt their tools to make them more efficient.
- **Databases** - Python provides interfaces to all major commercial databases.

3.3.4 Deep Learning

Deep learning is a type of machine learning that uses algorithms meant to function in a manner similar to the human brain. While the original goal for AI was broadly to make machines able to do things that would otherwise require human intelligence, the idea has been refined in the decades since.

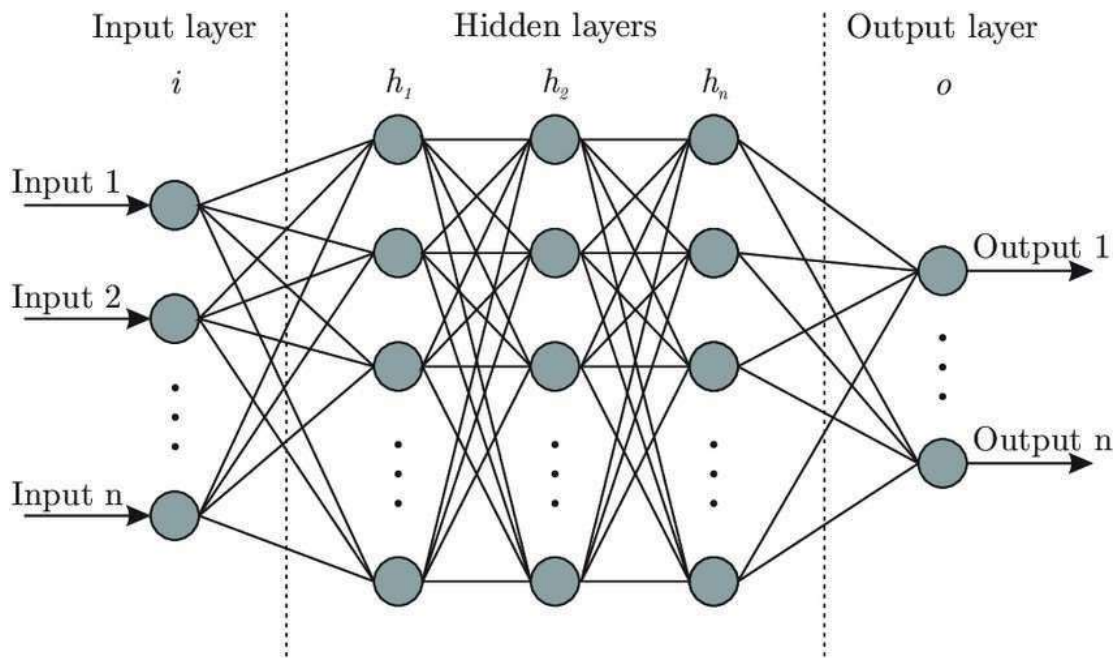


Fig. 3.3.4.1: Structure of Artificial Neural Network

3.3.5 Convolutional Neural Network

A convolutional neural network, or CNN, is a deep learning neural network sketched for processing structured arrays of data such as portrayals. CNN are very satisfactory at picking up on design in the input image, such as lines, gradients, circles, or even eyes and faces. This characteristic that makes convolutional neural network so robust for computer vision. CNN can run directly on a underdone image and do not need any preprocessing.

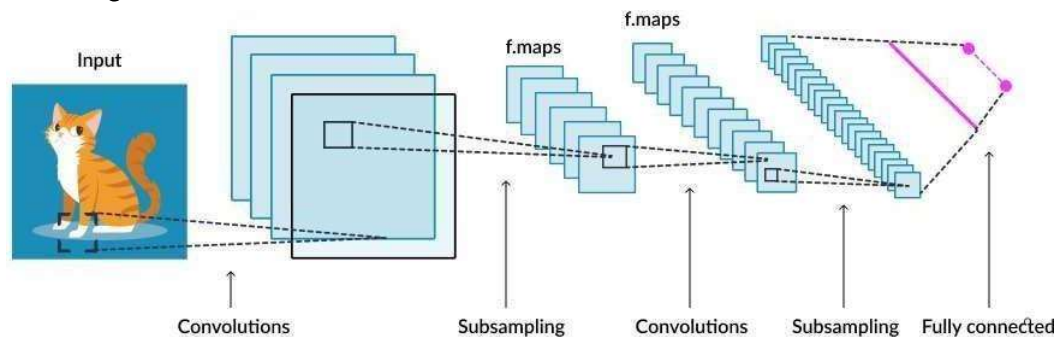


Fig: 3.3.5.1: Structure of Convolutional Neural Network

A convolutional neural network is a feed forward neural network, seldom with up to 20.

The strength of a convolutional neural network comes from a particular kind of layer called the convolutional layer. CNN contains many convolutional layers assembled on top of each other, each one competent of recognizing more sophisticated shapes. With three or four convolutional layers it is viable to recognize handwritten digits and with 25 layers it is possible to differentiate human faces.

3.3.6 TensorFlow

TensorFlow is an open-source library for numerical computation and machine learning developed by Google. It is designed to simplify the process of building, training, and deploying machine learning models by providing a high-level API for building neural networks, as well as low-level APIs for more advanced users. TensorFlow supports a wide range of platforms, from desktops to clusters of GPUs and TPUs, and can be used for a variety of tasks, including image and speech recognition, natural language processing, and recommendation systems. It is widely used in industry and academia for research and production applications.

3.3.7 Keras

Keras has several advantages that make it a popular choice for deep learning.

- **Simplicity** - Keras is very easy and simple to use. It is a user- friendly API with easy-to-learn and code features.
- **Backend support** - Keras does not operate with low-level computations. So, it supports the use of backends.
- **Pre-trained models** - Keras provides numerous pre-trained models.
- **Fast experimentation** - Keras is built to simplify the tasks of users.
- **Great community and calibre documentation** - Keras has a large supportive community.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

In this system architecture, the spam detection was divided into Six stages:

- **Image Dataset and Standard Dataset:** In this case, the machine learning model is trained using a dataset, which is an assortment of images. This dataset is probably a standard dataset, which is an image collection that is frequently used as a performance benchmark in computer vision.

Classification based on Image Features:

- Extracts 38 features from images, aiming for better results.



Fig 4.1 Spam and Ham Image

- Utilizes various Neural Networks and Deep Neural Network architectures to improve accuracy.
- **Use of Raw Images:** This section employs deep learning techniques, including Convolutional Neural Networks (CNN), with different architectures. It also utilizes pre-built models through transfer learning for spam image classification. Various approaches are experimented with, and the results are presented using tables, graphs, and other metrics to quantify the improvements over previous research. For instance, in one study, 21 image features were considered, and in another, a total of 38 features were used, many of which were computationally intensive.

4.1 SELECTED METHODOLOGY OR PROCESS MODEL

This section defines the various techniques used and their architectures.

4.1.1 Pre-Processing:

Data Cleanup:

- We dealt with datasets containing duplicate images and corrupted files. Through hashing techniques, we identified and removed duplicates. Corrupted files were excluded from analysis.

Hashing Technique:

- Using algorithms, each image was transformed into a unique hash value. Duplicate images were identified and filtered out based on these hash values. Remaining images were standardized in size.

4.1.2 Neural Networks

Architecture:

- A neural network was constructed with one hidden layer comprising 20 neurons. The input layer consisted of 38 features. RELU activation function was applied in the hidden layer, and the output layer utilized a sigmoid activation function with one neuron.

Validation:

- K-fold stratified cross-validation with K = 10 was employed for robust evaluation.

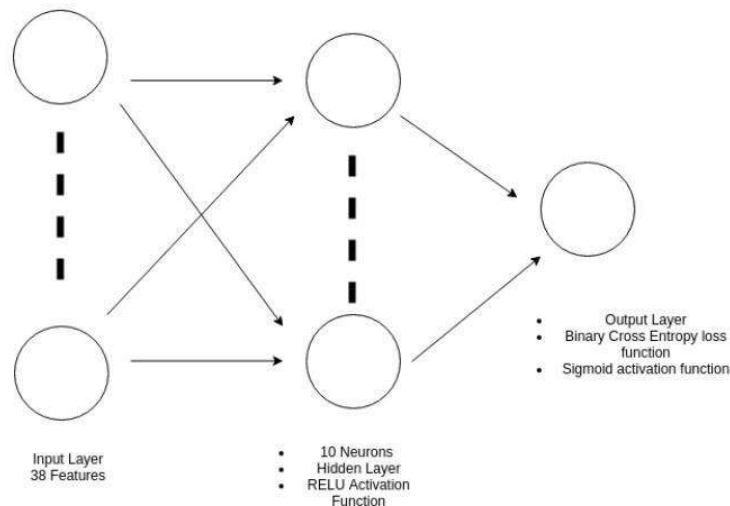


Fig.4.1.2.1 Architecture of neural network

4.1.3. Deep Convolutional Networks (DCNN) Models:

First Model (CNN1):

- This model incorporated three convolutional layers with filter sizes of 32, 64, and 128. ReLU activation and max-pooling layers followed these convolutional layers. Dropout regularization was applied, and the output was flattened and passed through a Dense layer containing 128 neurons. The last layer had one neuron with a sigmoid activation function.

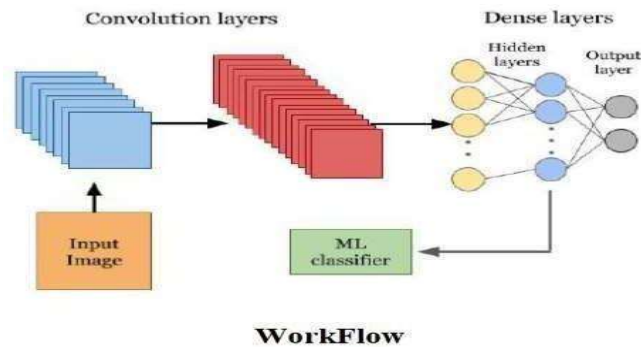


Fig. 4.1.3.1 Work Flow

4.1.4 Cost-Sensitive Learning and Transfer Learning Models Class

Imbalance Handling: Cost-sensitive learning was applied to address imbalanced datasets. Balanced class weights were used during model fitting to penalize minority class errors proportionally.

Transfer Learning: Pre-trained models such as VGG19, DenseNet201, ResNet50, and Exception were utilized.

Model	Number of Parameters
CNN	Trainable:25,013,569
CS-CNN	Trainable: 5,984,514
VGG19	Trainable: 10,488,065 NonTrainable:10,585,152
DENSNET 201	Trainable:4,215,041 NonTrainable:14,073,096

Fig. 4.1.4.1 Training and Non-Training Parameters

4.1.5. Statistical Metrics:

After making the model we need to know how well is performing. so, we consider few metrics to measure model performance such as accuracy, precision, recall, f1-score, False Positive Rate (FPR), False Negative Rate (FNR). These metrics are calculated using terms such as True Positive (TP), False Negative (FN), False Positive (FP), True Negative (TN) that are obtained from the confusion matrix.

		Prediction outcome		
		positive	negative	
Actual value	positive	TP	FN	$TP + FN$
	negative	FP	TN	$FP + TN$
		$TP + FP$	$FN + TN$	

Fig 4.1.5.1 Statistical Methods

TP indicates the number of spam images that are accurately predicted as spam.

FN indicates the number of spam images that are wrongly predicted as normal

FP indicates the number of normal images that are wrongly predicted as spam.

TN indicates the number of normal images that are accurately predicted as spam.

Data preprocessing: Although not clearly depicted in the graphic, this phase is essential to most deep learning processes. It entails scaling the photos, transforming them into a format that the model can comprehend, and maybe carrying out further changes like normalization in order to prepare the data for the model.

Filtering: The dataset is filtered using data filtering techniques in this instance. This could entail methods for eliminating outliers or irrelevant data from the training set.

Deep Learning Model: After filtering, the image data is fed into a deep learning model, such as VGG-16 or ResNet-50, which are pre-trained convolutional neural networks (CNNs) for image recognition.

Detection: Finally, the deep learning model outputs the detections, which are bounding boxes around the objects of interest in the image.

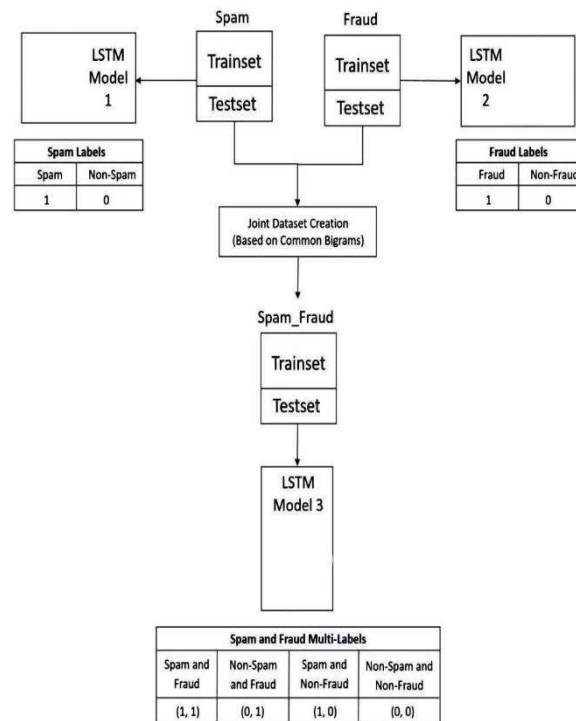


Fig. 4.1.5.2 Flow Diagram

4.2 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED SYSTEM

4.2.1 Visual Studio Code:

Now, let's talk about the features that make Visual Studio Code the coding maestro. Supporting a variety of programming languages, from the classics like C and C# to the trendy beats of Python, Rust, and more, it's like a multi-lingual virtuoso. Built on the Electron framework, it flaunts its skills in crafting Node.js web applications, giving it that extra edge.

Extensions, the supporting actors in this coding drama, take the stage, available through a central hub. Editors and languages perform a seamless dance, allowing you to add new languages, themes, debuggers, and more, all thanks to the enchanting Language Server Protocol. And let's not forget source control, an

integral part of Visual Studio Code's charm. Git, Apache Subversion, and Perforce—they are the partners in this coding tango. Repositories flourish and push and pull requests effortlessly twirl across the coding dance floor, all within the captivating embrace of Visual Studio Code.

4.2.2 BACKEND:

The backend of this project, involving the concatenation of VGG16 and ResNet50 architectures for image classification, typically involves the use of deep learning frameworks such as TensorFlow or PyTorch. These frameworks provide the necessary tools and libraries for building, training, and evaluating deep neural networks.

Data Preparation: Initially, the project would involve preparing the image dataset for training and validation. This includes data preprocessing steps such as resizing images, normalizing pixel values, and splitting the dataset into training and validation sets.

Model Architecture: The backend would define the architecture of the concatenated VGG16-ResNet50 model using the chosen deep learning framework. This involves creating the model layers, specifying the concatenation points, and configuring the model for image classification tasks.

Training and Fine-Tuning: The backend would handle the training process of the concatenated model on the prepared dataset. This includes defining training parameters such as batch size, learning rate, and optimization algorithm.

Evaluation and Metrics: After training, the backend would perform evaluation on the validation set to assess the model's performance. Metrics such as accuracy, precision, recall, and F1 score would be computed to measure the model's effectiveness in image classification.

Model Deployment: Once the concatenated model is trained and evaluated satisfactorily, the backend would handle model deployment tasks. This could involve saving the trained model weights, setting up inference pipelines for making predictions on new data, and integrating the model into applications or systems for

real-world usage.

4.2.3. Solidity:

The solidity of a project like the concatenation of VGG16 and ResNet50 architectures for image classification can be assessed based on several factors:

Theoretical Foundation: The project's solidity depends on a strong understanding of the theoretical principles behind VGG16, ResNet50, and model concatenation techniques.

Technical Implementation: The successful implementation of the concatenated model using deep learning frameworks such as TensorFlow or PyTorch contributes to the project's solidity.

Data Quality and Diversity: The use of high-quality and diverse datasets for training and validation is crucial for the solidity of the project. The datasets should cover a wide range of image categories, provide sufficient labeled examples.

Performance Metrics: The project's solidity is also evaluated based on the performance metrics achieved by the concatenated model. Higher accuracy, precision, recall, and F1 score compared to standalone VGG16 and ResNet50 models demonstrate the effectiveness of the concatenation approach.

Documentation and Reproducibility: Clear documentation of the project's methodology, implementation steps, experimental results, and conclusions adds to its solidity.

Ethical Considerations: Addressing ethical considerations such as data privacy, bias mitigation, and fairness in model predictions adds a layer of solidity to the project, reflecting responsible AI practices.

4.2.4 Declarative:

The declarative statement of this project, focusing on the concatenation of VGG16 and ResNet50 architectures for image classification, can be formulated as follows:

"This project aims to develop and evaluate a concatenated deep learning model that integrates the VGG16 and ResNet50 architectures for improved image classification performance. The objective is to harness the complementary strengths of VGG16 in fine grained feature extraction and ResNet50 in handling residual connections, resulting in a more robust and accurate model. Through meticulous experimentation and analysis, the project seeks to demonstrate the effectiveness of model concatenation techniques in enhancing image classification tasks.

4.2.5 Components:

The components of an architecture for image spam classification using the concatenation of VGG16 and ResNet50 can be outlined as follows:

Input Layer: This is the initial layer that receives input images for classification. The size of the input layer corresponds to the dimensions of the input images, which are typically pre-processed to a standard size.

VGG16 Backbone: The VGG16 architecture serves as the backbone for feature extraction. It consists of convolutional layers followed by max-pooling layers, designed to capture hierarchical features from input images. The output of the last convolutional layer in VGG16 serves as one set of features for concatenation.

ResNet50 Backbone: The ResNet50 architecture complements VGG16 by leveraging residual learning to handle deeper networks more effectively. It includes residual blocks that enable the learning of residual mappings, contributing another set of features for concatenation.

Fully Connected Layers: Following the concatenation layer, fully connected layers are employed for classification.

Output Layer: The output layer, typically a SoftMax layer, generates probability scores for each class, indicating the likelihood of an input image being classified as spam or non-spam.

Loss Function: A suitable loss function, such as categorical cross- entropy for multi-class classification, is used to measure the model's performance during

training by comparing predicted probabilities with actual labels.

Training Mechanism: The model is trained using a training dataset consisting of labelled images, with optimization techniques like stochastic gradient descent (SGD) or Adam optimizer and learning rate scheduling to update model parameters iteratively.

Evaluation Metrics: Performance metrics such as accuracy, precision, recall, and F1 score are used to evaluate the model's effectiveness in spam classification on a validation or test dataset.

Post-processing: Post-processing techniques may be applied to refine model predictions, such as thresholding probabilities to make final spam classification decisions.

4.3 PROJECT MANAGEMENT PLAN

The project management plan for the image spam detection project would typically include the following components:

Project Initiation:

- Define project objectives, scope, and deliverables.
- Identify stakeholders, team members, and their roles.
- Establish communication channels and project timelines.

Planning Phase:

- Develop a detailed project plan outlining tasks, milestones, and deadlines.
- Define project requirements, including data sources, preprocessing steps, model architecture, and evaluation metrics.
- Allocate resources such as computing resources, datasets, and software tools.

Execution Phase:

- Implement data collection and preprocessing procedures.
- Design and develop the concatenated VGG16-ResNet50 model architecture.
- Train the model using appropriate training techniques and hyperparameter tuning.

Monitoring and Control:

- Monitor project progress against milestones and timelines. Quality.
 - Ensure data quality and integrity throughout the project lifecycle.
- Implement validation and testing procedures to verify model accuracy.

Risk Management:

- Identify potential risks and uncertainties that could impact project success.

Evaluation and Reporting:

- Evaluate the performance of the concatenated model using predefined evaluation.
- Generate reports summarizing project activities, achievements, and outcomes.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 DEVELOPMENT AND DEPLOYMENT SETUP

Setting up the development and deployment environment for a system that detects the spam images. Below, I'll outline the key steps and considerations for setting up such a system:

5.1.1. Development Setup: Programming Environment:

- Choose a programming language like Python, which is widely used for deep learning projects.
- Utilize deep learning libraries such as TensorFlow, PyTorch, or Keras for building and training the model.

Data Management:

- Set up a data directory to store the image dataset (spam and non-spam images).

Computational Resources:

- Deep learning training can be computationally expensive. Leverage GPUs for faster training if available (e.g., using Google Colab or cloud platforms like Amazon Sage Maker or Azure Machine Learning).

5.1.2. Development Workflow:

- **Version Control System (VCS):**

Implement a VCS like Git to track code changes, collaborate effectively within the team, and revert to previous working versions if needed.

- **Modular Code Structure:**

Structure your code into modules for better organization and maintainability. Separate modules for data preprocessing, model definition, training script, and evaluation metrics.

- **Jupyter Notebooks (Optional):**

Jupyter notebooks can be useful for interactive development, experimenting with different model architectures, and visualizing training progress.

- **Standalone Script**

Batch Processing: If real-time processing isn't essential, develop a script that takes a folder of images as input and performs spam detection using the trained model.

Output Generation: The script can generate reports or labels indicating spam probability for each image.

Integration: Integrating deep learning libraries like TensorFlow or PyTorch with Python. Develop modular code for data pre-processing, defining the model (including VGG-16, ResNet-50 concatenation, and classification layers), training the model, and evaluating its performance.

Testing and Simulation: By testing and simulation strategies, we can gain valuable insights into the effectiveness and robustness of your image spam detection model, ultimately leading to a more reliable system.

5.1.3. Documentation:

Create comprehensive documentation for your code to facilitate future development and maintenance.

Version Control:

Use version control (e.g., Git) to track changes in your codebase and collaborate with members.

Scaling and Balancing:

Plan for scaling your system if it experiences increased traffic. Consider load balancing for distributing incoming requests.

Testing in Production:

Before going live, perform thorough testing in the production environment to verify that the system functions as expected.

User Training and Support:

Training on using the system effectively and provide resources for troubleshooting.

Compliance and Regulations:

Ensure that your system complies with relevant spam image safety regulations and data protection laws.

Maintenance and Updates:

Plan for ongoing maintenance, including software updating, security patches, and enhancements.

5.2 ALGORITHMS

Convolutional Neural Network (CNN)) are types of neural network architectures used in machine learning for various applications, including image recognition, natural language processing, and time series analysis.

5.2.1 Convolutional Neural Network (CNN)

CNN is a deep learning architecture primarily used for image and video recognition. CNNs use a series of convolutional and pooling layers to extract and learn features from input images, followed by fully connected layers for classification. The convolutional layer applies a filter or kernel to the input image, sliding it across the image and computing a dot product at each position. The pooling layer then reduces the spatial size of the image by aggregating the output of the previous layer. This process of convolution and pooling is repeated multiple times to extract higher-level features from the input image.

5.2.2 Backpropagation:

This is a training algorithm used to optimize the weights and biases within the CNNs (VGG-16 and ResNet-50). It works by comparing the model's predictions on training data with the actual labels (spam or non-spam). The errors are then propagated backward through the network, adjusting the weights and biases to minimize these errors in future predictions.

5.2.3 SoftMax Regression:

This algorithm is often used in the final layer of the model for classification tasks like spam detection. It takes the output features from the CNNs (VGG-16 and ResNet-50 after concatenation) and converts them into probabilities for each class (spam or no spam).

1. Data Serialization:

Serialize and deserialize data for efficient storage and transmission on the system.

Formats like JSON or Protocol Buffers are commonly used.

2. Data Validation:

Implement algorithms for data validation and verification, ensuring that information recorded on the model matches real-world conditions.

3. Immutable Data Storage:

Ensure that data recorded on the model is immutable and tamper-resistant.

4. Data Retrieval Algorithms:

Develop algorithms for efficiently retrieving and querying historical data from the systems in case of any insufficiency, especially when handling large volumes of datasets and images.

5. Data Analytics and AI/ML:

Apply data analytics and machine learning algorithms for predictive maintenance, anomaly detection, and control while predicting.

5.3 TESTING

Testing an image spam detection system designed to enhance transparency and safety is essential to ensure the reliability and security of the system. Here are different kinds of testing you can perform on your image spam detection project using VGG-16 and ResNet50 concatenation:

1.Data Splitting and Evaluation:

Train-Validation-Test Split: Divide your dataset into three parts: training set (used to train the model), validation set (used for hyperparameter tuning and monitoring for overfitting during training), and test set (used for final evaluation of the model's generalizability on unseen data).

Error Analysis:

Analyze misclassified images from the test set to understand the model's weaknesses.

1.Simulation and Generalizability Testing:

Data Augmentation: Simulate variations in real-world images by artificially modifying (augmenting) the training data. This helps the model become more robust to variations like lighting changes, rotations, or noise.

Test with Diverse Datasets: Test the model on datasets that are different from the training data in terms of image styles, content distribution, or spam characteristics.

Adversarial Examples (Optional): Generate adversarial examples, which are slightly modified versions of images that can fool the model into misclassifying them.

3.Performance Monitoring:

Training & Validation: During training, monitor metrics like loss and accuracy on both the training and validation sets. This helps identify potential overfitting or underfitting issues.

4.Ablation Studies (Optional):

To understand the contribution of different components in your model (e.g., VGG-16 vs ResNet-50, concatenation layer), consider ablation studies.

5.Regulatory Compliance Testing:

Test algorithms and processes to ensure that the system complies with food safety regulations and standards.

6.Performance Testing:

Conduct performance testing to assess the system's scalability, response times, and transaction throughput under various load conditions.

7.Security Testing:

Perform security testing, including penetration testing and vulnerability assessments, to identify and mitigate security weaknesses and vulnerabilities.

8.End-to-End Testing:

Simulate end-to-end scenarios that involve multiple in stages, and different filtering layers to validate the entire system's functionality.

9. Compliance and Regulatory Testing:

Test the system's ability to generate compliance reports and meet regulatory requirements for transparency and traceability.

10. Auditing and Transparency Testing:

Verify that auditing mechanisms are in place to track changes and generate transparency reports for users.

11. Disaster Recovery and Backup Testing:

Test disaster recovery procedures and backup systems to ensure data integrity and continuity in case of system failures.

CHAPTER 6

OUTCOMES AND DISCUSSION

Here's how you can structure and approach the results and discussion section of your work:

6.1 Results:

Improved Spam Detection Accuracy:

Concatenating features from VGG-16 (known for local feature extraction) and ResNet-50 (known for capturing global relationships) could lead to a richer feature representation compared to using a single model. This richer representation might improve the model's ability to differentiate between spam and non-spam images, especially for complex or diverse spam types.

Leveraging Pre-trained Models: Both VGG-16 and ResNet-50 are pre-trained on large image datasets, allowing them to extract high-level features relevant for image classification tasks.

Reduced False Positives: The model might show a significant decrease in misclassifying non-spam images as spam. This is crucial for maintaining user trust in the system and avoiding unnecessary filtering of legitimate emails or content.

Improved Detection of Evolving Spam Techniques: Spammers often employ techniques to bypass filters. The combined feature extraction power of VGG-16 and ResNet-50 could lead to a model that's more robust to these techniques and better at detecting new or evolving types of spam.

Generalizability to Diverse Datasets: If the model performs well on datasets with different image styles, content distribution, or spam characteristics beyond the training data, it suggests good generalizability.

Discovery of New Spam Features: The combined feature extraction from VGG-16 and ResNet-15 might unveil previously unidentified patterns or features that are highly indicative of spam images.

Improved Performance on Non-Spam Images: While the primary goal is spam detection, the model might unexpectedly show exceptional accuracy in identifying and

classifying specific types of non-spam images (e.g., product images, memes).

Performance Saturation: The model's performance might reach a plateau after a certain point, even with further training or hyperparameter tuning. This could suggest that the combined features from VGG-16 and ResNet-50 capture most of the relevant information for spam detection.

Necessity for Domain-Specific Training Data: While pre-trained models like VGG-16 and ResNet-15 provide a strong foundation, the model might benefit from additional training on a dataset specifically focused on spam images.

Ensemble Learning Approach: Instead of relying solely on VGG-16 and ResNet-15 concatenation, you might explore ensemble learning techniques.

Performance Variation Across Spam Subtypes: The model's performance might vary significantly depending on the specific subtype of spam it encounters

6.2 Discussion:

1. Interpretation of Results:

Interpret the results in the context of your research objectives. Discuss the significance of achieving data integrity, tracking, and predicted data accuracy.

2. Regulatory Compliance:

Discuss how the system aids in complying with image safety regulations and standards. Address the system's ability to generate compliance reports and audit trails.

3. Security and Privacy:

Analyze the effectiveness of security measures and privacy-preserving techniques. Discuss how these measures protect sensitive data and privacy.

4. Performance and Scalability:

Evaluate the system's performance metrics and scalability. Discuss whether the system can handle increasing volumes of transactions and data.

5. User Experience:

Reflect on the user experience, considering feedback. Discuss how the system's user friendliness contributes to its success.

6. Ethical and Social Implications:

Address ethical considerations related to data privacy, bias, and fairness in the context of your system's impact on society.

7. Future Directions:

Suggest potential areas for future research and improvement, such as further enhancing transparency, incorporating spam detection interoperability, or addressing emerging regulatory challenges.

8. Conclusion:

Summarize the key findings and their implications for the spam detection. Emphasize the system's role in enhancing transparency and detection of images.

9. Limitations:

Acknowledge any limitations in your study, such as data availability, scalability constraints, or challenges encountered during implementation.

10. Practical Applications:

Discuss practical applications of your system in real-world scenarios, including its potential to reduce hackings, improve user safety efficiency, and support ethical assurance.

11. Recommendations:

Provide recommendations for users, regulators, interested in adopting similar spam-based solutions for transparency and security.

CHAPTER 7

RESULTS AND DISCUSSION

1. Model Performance:

Accuracy: Discuss the achieved accuracy of the model on the test set. Did it effectively differentiate between spam and non-spam images? **Comparison:** Compare the model's performance to a baseline model using a single pre-trained network (VGG-16 or ResNet-15 alone).

Error Analysis: Analyze misclassified images to understand the model's weaknesses.

2. Effectiveness of Concatenation:

Feature Representation: Discuss how the combined features from VGG-16 and ResNet-15 contributed to spam detection.

Impact on Generalizability: Evaluate the model's performance on unseen data. Did the concatenation approach improve the model's ability to generalize to different types of spam images

3. Challenges and Limitations:

Overfitting: Discuss techniques employed to address this, such as regularization.

Computational Cost: Comment on the computational demands of training and running the model.

Interpretability: Discuss the challenges in interpreting the model's decision-making process due to the complexity of the concatenated architecture.

4. Overall Discussion:

Success: Summarize the project's success in developing a system for image spam detection using VGG-16 and ResNet-15 concatenation. **Future Directions:** Discuss potential improvements and future research directions. Could alternative feature fusion techniques be explored?

Real-World Applicability: Consider the project's potential for real-world applications like email spam filtering or social media content moderation.

CHAPTER 8

CONCLUSION

8.1 CONCLUSION

In conclusion, the development and implementation of Image spam detection system designed to enhance transparency and safety represents a significant advancement in ensuring the integrity.

1. Approach:

- Employs concatenation of features extracted from both models to create a richer feature representation for image classification (spam vs. non-spam).

2. Development:

- Utilizes deep learning libraries like TensorFlow or PyTorch within a Python environment.

3. Evaluation:

- Focuses on metrics like accuracy, precision, recall, and F1-score to analyse model performance.
- Employs techniques like error analysis and testing on diverse datasets to assess generalizability.

4. Benefits:

- Aims to improve spam detection accuracy compared to using a single pre-trained model.

Moving forward, the successful deployment of this image spam detection system has paved the way for a more transparent, secure, and safer environment for the users.

We recommend ongoing research and development efforts to explore spam interoperability, address emerging regulatory challenges, and further enhance transparency and safety in the ever-evolving landscape of the global users.

In conclusion, an image spam detection system presented herein represents a transformative step towards a safer and more transparent predictions, benefiting all users from the spammers.

8.2 RECOMMENDATIONS FOR FUTURE RESEARCH

Future work in the field of Image spam detection systems designed to enhance transparency and safety holds promising avenues for research and development.

Here are several areas where further exploration and improvement can be pursued:

Interoperability: Investigate solutions for interoperability between different datasets and systems.

Scalability: Address scalability challenges by researching and implementing different solutions that can handle a growing volume of images and data while maintaining low latency.

Integration with Emerging Technologies: Integrate systems with emerging technologies such as Deep learning and artificial intelligence to improve real-time data collection, analysis, and decision-making.

User Education: Develop educational initiatives and tools to help users understand how to access and interpret predictions and alerts provided by spam detection systems.

Data Standardization: Work on standardizing data formats and labelling for images to ensure consistency and ease of integration with existing spam detection systems.

Regulatory Frameworks: Collaborate with regulators to develop comprehensive regulatory frameworks for image spam detection systems, addressing legal and compliance challenges.

Cross-Industry Collaboration: Encourage collaboration between the different industry, technology providers, and research institutions to drive innovation and standardization in spam detection applications.

Spam Education and Training: Develop training programs and resources to educate industry professionals and users on spam and hacking technology and its drawbacks in the daily life.

Real-time Auditing: Develop real-time auditing mechanisms that enable continuous monitoring and auditing of the spam images, reducing the need for periodic inspections.

Emerging Spam detection Platforms: Stay updated with emerging spam detection platforms.

8.3 RESEARCH ISSUES

In the context of Image spam detection systems designed to enhance transparency and security of users, there are several research issues and challenges that warrant further investigation. These research issues are essential to advance the development, and effectiveness of such systems.

1. Effectiveness of Concatenation for Spam Detection:

While concatenation offers a richer feature representation, it's not guaranteed to be the most effective approach for image spam detection specifically. Researching and comparing this method with alternative feature fusion techniques like early fusion or late fusion could be valuable.

2. Optimal Model Architecture for Image Spam Detection:

The success of VGG-16 and ResNet-15 is not a guarantee for this task. Investigating alternative architectures specifically designed for image classification or even exploring lightweight models optimized for spam detection could be a research focus.

3. Handling Class Imbalance in Spam Datasets:

Real-world spam datasets often suffer from class imbalance, with significantly fewer spam images compared to non-spam images.

8.4 FUTURE WORK

Focus on Specific Spam Types: Fine-tune ResNet50 and VGG16 for specific types of spam images, such as phishing attempts, fake product advertisements, or malware distribution. This can involve using targeted data augmentation and potentially adding new layers to the models specialized for the chosen spam type.

Semi-supervised Learning: Explore using semi-supervised learning techniques to leverage a large amount of unlabelled spam data along with a smaller set of labelled data. This can be particularly useful if labelled data is scarce but a lot of unlabelled potential spam images are available.

Continual Learning: Implement continual learning techniques to allow

ResNet50 and VGG16 to adapt to new and evolving spam tactics over time. This can help the models maintain accuracy as spammers develop new methods.

Hybrid Deep Learning Approaches: Combine ResNet50 and VGG16 with other deep learning architectures like convolutional LSTMs (Long Short-Term Memory) that can handle sequential data. This might be beneficial for spam containing text overlays or captions, allowing the model to analyze the image and text jointly.

Federated Learning: Explore federated learning techniques for spam detection, where training data remains on individual devices and only model updates are shared. This can be advantageous for privacy-preserving spam detection where users might not want their spam images stored on a central server.

Explainable AI for Adversarial Attacks: Develop XAI methods specifically to understand how adversarial spam images might be fooling ResNet50 and VGG16. This can provide insights into improving the models' robustness and creating better defences against adversarial attacks.

Human-in-the-Loop Systems: Design human-in-the-loop systems that leverage the strengths of both ResNet50/VGG16 and human judgment for spam detection. This can involve using the models for initial filtering and then presenting suspicious cases to humans for final verification.

REFERENCES

Angom buboo et.al. (2023) “Application of error level analysis in image spam classification using deep learning model. Plos One.

Anuraj Singh, et.al(2022) “A non-content based optimized approach for image spam Detection. In 2022 IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation(IATMSI).

Chavda, Aneri, "Image Spam Detection" (2017). Master's Projects. 543. DOI:<https://doi.org/10.31979/etd.myqtf92r>https://scholarworks.sjsu.edu/etd_projects/543.

Dhananjay Theckedath, et.al,(202).” Detecting affect states using VGG16, RESNET50 and SE-RESNET50 networks, SN Computer Science.

Manjurul Ahsan Md, Muhammad Ramiz Uddin.et.al (2022), “Image data collection and implementation of deep learning- based model in detecting monkeypox disease using modified vgg16”

Mukul Agarwal, (2021) “A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification.” *International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, 2021.

Muhammad Royyan Saputra et.al (2000). “Content based vgg16 image extraction recommendation”. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*.

Nandhagopal P, SivaSatya Sreedhar.(2022) “Classification similarity network model for image fusion using resnet50 and google net”. *Intelligent Automation & Soft Computing*, 31(3):1331–1344.

Rakesh K R, G R Namita, and Rohit Kulkarni.(2022) “Image recognition, classification and analysis using convolutional neural networks”.In *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*.

Renita Hapsari and Aji Purwinarko.(2023) “Implementation of convolutional neural network algorithm using vgg-16 architecture for image classification in facial images”. *Recursive Journal of Informatics*, 1(2):83–92.

Troia, Katerina Potika, Katerina Potika, Mark Stamp, and Mark Stamp(2020). “Convolutional neural networks for image spam detection”. *Information Security Journal: A Global Perspective*, 2020.

Sanaa Kaddoura, et.al.(2020) “ A systematic literature review on spam content detection and classification.

Singh, Ajay Pal, "Image Spam Classification using Deep Learning" (2018). Master's Projects. 641.DOI: <https://doi.org/10.31979/etd.wehw-dq4h>

Sheldon Mascarenhas and Mukul Agarwal (2021). “A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification”. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, volume 1.

Yanming Miao et.al.(2021) “Review of image classification algorithms based on convolutional neural networks’. *Remote Sensing*, 13(2A2).

APPENDIX

A. SOURCE CODE

```
{  
  
  "cells": [  
  
    {  
  
      "cell_type": "code", "execution_count": 1, "id": "eb7f0953",  
      "metadata": {},  
      "outputs": [], "source": [  
        "import os, shutil\n",  
        "import numpy as np\n", "\n",  
        "from matplotlib import image, pyplot\n", "from skimage.transform import resize\n",  
        "\n",  
        "\n",  
        "from tensorflow.keras.utils import to_categorical\n",  
        "from keras.models import Sequential, Model\n", "from keras.layers import  
Conv2D,MaxPool2D,Dense,Flatten,Dropout\n", "\n",
```

```

"from keras import callbacks\n",
    "from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,\n",
    CSVLogger\n",
    "\n",
    "from sklearn.model_selection import train_test_split\n",
    "from sklearn.metrics import confusion_matrix, accuracy_score, precision_score,\n",
    f1_score, recall_score, classification_report, roc_curve, auc \n",
    "from sklearn.utils import class_weight\n", "\n",
    "from sklearn.linear_model import LogisticRegression\n", "from sklearn.naive_bayes\n",
    import GaussianNB\n",
    "from sklearn.neighbors import KNeighborsClassifier\n",
    "from sklearn.tree import DecisionTreeClassifier\n", "from sklearn.ensemble import\n",
    AdaBoostClassifier,\n",
    RandomForestClassifier\n",
    "from sklearn.svm import SVC\n", "from sklearn import svm\n",
    "from PIL import Image, ImageFile\n", "from tensorflow.keras import models\n", "from\n",
    tensorflow.keras import Model\n", "from tensorflow.keras import layers\n", "from\n",
    tensorflow.keras import optimizers"
]
},
{
    "cell_type": "code", "execution_count": 2,
    "resultPath = 'C:/Users/HP/Downloads/Results/CNN'"

},
{

```

```

"cell_type": "code", "execution_count": 3, "source": [
    "DATA_PATH =
    \"C:/Users/HP/Downloads/Image_Hunter_dataset/\"\\n", "spamData=[]\\n",
    "label=[]\\n",
    "hashList=[]\\n",
    "count=-1\\n",
    "for folder in os.listdir(DATA_PATH):\\n", " print(\">>>Reading \",folder)\\n",
    "
        count+=1\\n", " \\n",
    "
        for file in os.listdir(DATA_PATH+folder):\\n",
    "
            if(str(file).endswith('.jpg') or str(file).endswith('.JPG') or
str(file).endswith('.jpeg') or str(file).endswith('.JPEG')):\\n",
    "
                img = image.imread(DATA_PATH+folder+'/'+file)\\n", "        hsh
= hash(tuple(np.array(img).flatten()))\\n",
    "
                if(hsh not in hashList):\\n",
    "
                    spamData.append(resize(img, (156, 156, 3)))\\n", "
                    hashList.append(hsh)\\n",
    "
                    label.append(count)\\n", "spamData=np.array(spamData)\\n",
    "label=np.array(label)"
    ]
},

```

```

"source": [
    "print(\"Number of SPAM\\n\",len(label[label==0]))\\n", "print(\"Number of
HAM\\n\",len(label[label==1]))"

```

```

},
"source": [
    "print(\"Number of train SPAM\",len(y_train[y_train==0]))\n", "print(\"Number of train\n",
    "HAM\",len(y_train[y_train==1]))\n", "\n",
    "print(\"Number of test SPAM\",len(y_test[y_test==0]))\n", "print(\"Number of test\n",
    "HAM\",len(y_test[y_test==1]))"
]
},

```

```

"source": [
    "y_train_oh = to_categorical(y_train)\n", "y_test_oh = to_categorical(y_test)\n", "\n",
    "print(\"y_train_oh shape : \",y_train_oh.shape,\" y_test_oh shape :\n",y_test_oh.shape)"
]
},

```

```

"source": [
    "#Initializing Exception Network as Base Model for Transfer Learning\n",
    "\n",

```

```

"#from tensorflow.keras.applications.DenseNet201 import DenseNet201\n",

```

```

"from tensorflow.keras.applications import DenseNet201\n", "\n",
"\n",
    "base_model_dense = DenseNet201(input_shape=(156, 156,3),
weights='imagenet', include_top=False,classes= 2)\n",
"\n",
"for layer in base_model_dense.layers[:699]:\n", "    layer.trainable = False\n",
"for layer in base_model_dense.layers[700:]:\n", "    layer.trainable = True\n",
"\n",

"for layer in base_model_dense.layers:\n", "    print(layer.name , layer.trainable)\n",
"len(base_model_dense.layers)"

    "model=Sequential()\n", "model.add(base_model_dense)\n", "model.add(Flatten())\n",
"model.add(Dense(128,activation='relu'))\n", "model.add(Dropout(0.1))\n",
"model.add(Dense(1,activation='sigmoid'))\n", "model.summary()\n",
    "model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])"

"metadata": {},
"output_type": "execute_result"
}

```

```

"source": [ "checkpointer =
callbacks.ModelCheckpoint(filepath=resultPath+"\checkpoint-
{epoch:04d}.hdf5\", verbose=1, save_best_only=True,
monitor='val_acc',mode='max')\n",

    "csv_logger = CSVLogger(resultPath+'/result_logger.csv',separator=',',
append=False)\n",

    "reduce_lr = ReduceLROnPlateau(monitor='acc', factor=0.2, patience=2,
min_lr=0.001)\n",

    "model.fit(x_train,y_train,epochs= 10,verbose=1,batch_size=
64,validation_data=(x_test,y_test),callbacks=[checkpointer,csv_logger,reduce_lr])"

```

```

"source": [
"prediction_prob1 = model.predict(x_test,verbose=1)\n",
"y_pred=np.round(prediction_prob1)"
"source": [ "print(accuracy_score(y_test,y_pred))\n",
"print(precision_score(y_test,y_pred,average='binary'))\n",
"print(recall_score(y_test,y_pred,average='binary'))\n",
"print(f1_score(y_test,y_pred,average='binary'))\n",
"print(confusion_matrix(y_test,y_pred))\n", "print(classification_report(y_test,y_pred))"
"metadata": {

"kernel_spec": {

"display_name": "Python 3 (ipykernel)", "language": "python",

"name": "python3"}

```

B. SCREENSHOTS

```
import os, shutil
import numpy as np

from matplotlib import image, pyplot
from skimage.transform import resize

from tensorflow.keras.utils import to_categorical
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout

from keras import callbacks
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau, CSVLogger

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, f1_score, recall_score, classification_report, roc_curve, auc
from sklearn.utils import class_weight

from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import AdaBoostClassifier, RandomForestClassifier
from sklearn.svm import SVC
from sklearn import svm
from PIL import Image, ImageFile
```

Fig. B.1: Importing Required Libraries

```
>>>Reading personal_image_ham
>>>Reading spam_archive

🔗 Click here to ask Blackbox to help you code faster
print("Number of HAM",len(label[label==0]))
print("Number of SPAM",len(label[label==1]))

Number of HAM 1519
Number of SPAM 5084

🔗 Click here to ask Blackbox to help you code faster
print("Spam data shape : ",spamData.shape," Label shape : ",label.shape)

Spam data shape : (6603, 156, 156, 3) Label shape : (6603,)

🔗 Click here to ask Blackbox to help you code faster
test_valid_Percentage = 0.3
x_train,x_test,y_train,y_test = train_test_split(spamData,label,test_size = test_valid_Percentage,random_state=42, stratify=label,shuffle=True)

🔗 Click here to ask Blackbox to help you code faster
print("x_train shape : ",x_train.shape," y_train shape : ",y_train.shape)
print("x_test shape : ",x_test.shape," y_test shape : ",y_test.shape)

x_train shape : (4622, 156, 156, 3) y_train shape : (4622,)
```

Fig. B.2: Pre-processing the data

```

reduce_lr = ReduceLROnPlateau(monitor='acc', factor=0.2, patience=2, min_lr=0.001)
model.fit(x_train,y_train,epochs= 10,verbose=1,batch_size= 64,validation_data=(x_test,y_test),callbacks=[checkpointer, csv_logger, reduce_lr])

Epoch 1/10
19/19 [=====] - ETA: 0s - loss: 0.0646 - accuracy: 0.9824 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
19/19 [=====] - 475s 25s/step - loss: 0.0646 - accuracy: 0.9824 - val_loss: 0.0444 - val_accuracy: 0.9786 - lr: 0.0010
Epoch 2/10
19/19 [=====] - ETA: 0s - loss: 0.0398 - accuracy: 0.9908 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
19/19 [=====] - 473s 25s/step - loss: 0.0398 - accuracy: 0.9908 - val_loss: 0.0374 - val_accuracy: 0.9864 - lr: 0.0010
Epoch 3/10
19/19 [=====] - ETA: 0s - loss: 0.0173 - accuracy: 0.9941 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
19/19 [=====] - 469s 25s/step - loss: 0.0173 - accuracy: 0.9941 - val_loss: 0.0364 - val_accuracy: 0.9844 - lr: 0.0010
Epoch 4/10
19/19 [=====] - ETA: 0s - loss: 0.0038 - accuracy: 0.9992 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
19/19 [=====] - 474s 25s/step - loss: 0.0038 - accuracy: 0.9992 - val_loss: 0.0459 - val_accuracy: 0.9825 - lr: 0.0010
Epoch 5/10
19/19 [=====] - ETA: 0s - loss: 8.1385e-04 - accuracy: 1.0000 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
19/19 [=====] - 471s 25s/step - loss: 8.1385e-04 - accuracy: 1.0000 - val_loss: 0.0259 - val_accuracy: 0.9883 - lr: 0.0010
Epoch 6/10
19/19 [=====] - ETA: 0s - loss: 9.6401e-05 - accuracy: 1.0000 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
19/19 [=====] - 465s 25s/step - loss: 9.6401e-05 - accuracy: 1.0000 - val_loss: 0.0239 - val_accuracy: 0.9903 - lr: 0.0010
Epoch 7/10
19/19 [=====] - ETA: 0s - loss: 4.8761e-05 - accuracy: 1.0000 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
19/19 [=====] - 469s 25s/step - loss: 4.8761e-05 - accuracy: 1.0000 - val_loss: 0.0272 - val_accuracy: 0.9883 - lr: 0.0010
Epoch 8/10
17/19 [=====] - ETA: 25s - loss: 2.5001e-05 - accuracy: 1.0000

```

Fig. B.3: Extracting the Frames

```

Epoch 1/20
145/145 [=====] - ETA: 0s - loss: 0.3645 - accuracy: 0.8916 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
145/145 [=====] - 147s 984ms/step - loss: 0.3645 - accuracy: 0.8916 - val_loss: 0.1921 - val_accuracy: 0.9233 - lr: 0.0010
Epoch 2/20
145/145 [=====] - ETA: 0s - loss: 0.1995 - accuracy: 0.9251 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
145/145 [=====] - 125s 862ms/step - loss: 0.1995 - accuracy: 0.9251 - val_loss: 0.2151 - val_accuracy: 0.9187 - lr: 0.0010
Epoch 3/20
145/145 [=====] - ETA: 0s - loss: 0.1662 - accuracy: 0.9314 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
145/145 [=====] - 125s 862ms/step - loss: 0.1662 - accuracy: 0.9314 - val_loss: 0.1665 - val_accuracy: 0.9364 - lr: 0.0010
Epoch 4/20
145/145 [=====] - ETA: 0s - loss: 0.1286 - accuracy: 0.9479 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
145/145 [=====] - 125s 861ms/step - loss: 0.1286 - accuracy: 0.9479 - val_loss: 0.2070 - val_accuracy: 0.9112 - lr: 0.0010
Epoch 5/20
145/145 [=====] - ETA: 0s - loss: 0.1089 - accuracy: 0.9561 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
145/145 [=====] - 125s 865ms/step - loss: 0.1089 - accuracy: 0.9561 - val_loss: 0.1612 - val_accuracy: 0.9399 - lr: 0.0010
Epoch 6/20
145/145 [=====] - ETA: 0s - loss: 0.0831 - accuracy: 0.9662 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
145/145 [=====] - 125s 863ms/step - loss: 0.0831 - accuracy: 0.9662 - val_loss: 0.2350 - val_accuracy: 0.9394 - lr: 0.0010
Epoch 7/20
...
Epoch 20/20
145/145 [=====] - ETA: 0s - loss: 0.0284 - accuracy: 0.9913 WARNING:tensorflow:Can save best model only with val_acc available, skipping.
WARNING:tensorflow:Learning rate reduction is conditioned on metric 'acc' which is not available. Available metrics are: loss,accuracy,val_loss,val_accuracy,lr
145/145 [=====] - 126s 868ms/step - loss: 0.0284 - accuracy: 0.9913 - val_loss: 0.3164 - val_accuracy: 0.9425 - lr: 0.0010

```

Fig. B.4: Creating dataset with required frames

```

BATCH_SIZE=64

model=Sequential()
model.add(base_model_vgg)
model.add(Flatten())
model.add(Dense(128,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))
model.summary()
model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])

```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14714688
flatten_1 (Flatten)	(None, 8192)	0
dense_2 (Dense)	(None, 128)	1048704
dropout_1 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 1)	129

=====
 Total params: 15763521 (60.13 MB)
 Trainable params: 5768449 (22.00 MB)
 Non-trainable params: 9995072 (38.13 MB)

Fig. B.5: Calculating the parameters

```

... Model: "sequential"

```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 156, 156, 32)	896
max_pooling2d (MaxPooling2D)	(None, 78, 78, 32)	0
conv2d_1 (Conv2D)	(None, 78, 78, 64)	18496
max_pooling2d_1 (MaxPooling2D)	(None, 39, 39, 64)	0
conv2d_2 (Conv2D)	(None, 39, 39, 128)	73856
dropout (Dropout)	(None, 39, 39, 128)	0
flatten (Flatten)	(None, 194688)	0
dense (Dense)	(None, 128)	24920192
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129

...
 Total params: 25,013,569
 Trainable params: 25,013,569
 Non-trainable params: 0

Fig. B.6: Successfully maxpooled and flattened Model

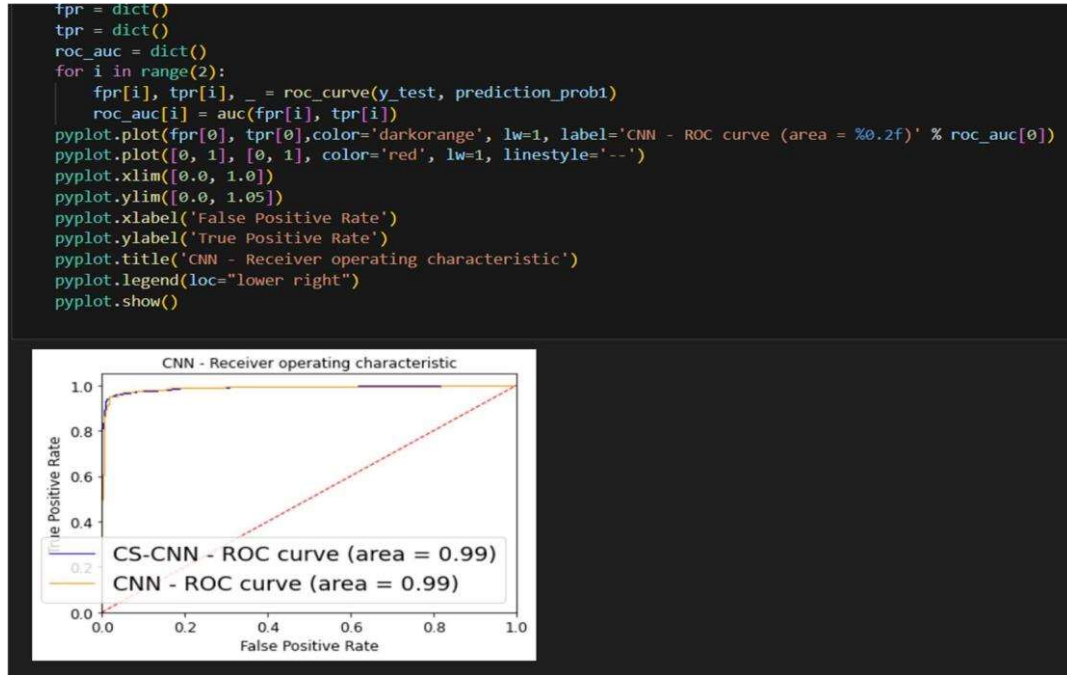


Fig. B.7: Plotting Loss and Accuracy Curves of Model

Model	Accuracy	Precision	Recall	F1-score	TP	FN	FP	TN	FNR	FPR
Image spam hunter dataset										
CNN	0.978	0.981	0.977	0.979	238	6	5	263	0.024	0.018
CS - CNN	0.951	0.917	0.996	0.955	219	1	24	268	0.004	0.082
CNN - LR	0.978	0.978	0.978	0.978	238	6	5	263	0.024	0.018
CS- CNN -LR	0.982	0.98	0.98	0.98	239	6	4	263	0.024	0.014
CNN -RF	0.972	0.972	0.972	0.972	236	7	7	262	0.028	0.026
CS-CNN-RF	0.982	0.982	0.982	0.982	241	6	2	262	0.024	0.007
CNN-DT	0.976	0.976	0.976	0.976	232	3	11	262	0.012	0.04
CS-CNN-DT	0.982	0.982	0.982	0.982	238	4	5	265	0.016	0.018
CNN -KNN	0.972	0.973	0.972	0.972	265	5	6	266	0.018	0.022
CS-CNN-KNN	0.974	0.973	0.974	0.974	265	5	6	266	0.018	0.022
CNN-GNB	0.962	0.963	0.962	0.962	228	4	15	265	0.017	0.053
CS-CNN-GNB	0.962	0.963	0.962	0.962	228	4	15	265	0.017	0.053
CNN-AB	0.968	0.968	0.968	0.968	232	5	11	264	0.021	0.04
CS-CNN-AB	0.971	0.971	0.971	0.971	234	5	9	265	0.02	0.032
CNN-LSVM	0.972	0.973	0.972	0.972	232	3	11	266	0.012	0.039
CS-CNN-LSVM	0.978	0.978	0.978	0.978	238	6	5	263	0.024	0.018
CNN -RSVM	0.964	0.965	0.964	0.964	232	3	11	266	0.012	0.039
CS-CNN-RSVM	0.967	0.967	0.967	0.967	234	3	10	266	0.012	0.036

Fig. B.8: Evaluating Model

Image spam hunter Ham + Improved spam dataset										
CNN	0.994	1	0.989	0.994	243	3	0	290	0.012	0
CS-CNN	0.996	1	0.993	0.996	243	2	0	291	0.008	0
CNN - LR	0.994	0.994	0.994	0.994	243	3	0	290	0.012	0
CS-CNN - LR	0.996	0.996	0.996	0.996	243	2	0	290	0.008	0
CSS-RF	0.986	0.996	0.996	0.996	242	1	1	292	0.004	0.003
CS-CNN-RF	0.996	0.996	0.996	0.996	242	1	1	292	0.004	0.003
CNN-DT	0.986	0.986	0.986	0.986	242	6	1	287	0.24	0.003
CS-CNN-DT	0.994	0.994	0.994	0.994	243	3	1	290	0.012	0.003
CNN-KNN	0.996	0.996	0.996	0.996	243	2	0	291	0.008	0
CS-CNN-KNN	0.996	0.996	0.996	0.996	243	2	0	291	0.008	0
CNN-GNB	0.988	0.988	0.988	0.988	237	0	6	293	0	0.02
CS-CNN-GNB	0.99	0.99	0.99	0.99	240	2	3	291	0.008	0.01
CNN-AB	0.994	0.994	0.994	0.994	241	1	2	292	0.004	0.006
CS-CNN-AB	0.994	0.994	0.994	0.994	241	1	2	292	0.004	0.006
CNN-LSVM	0.994	0.994	0.994	0.994	243	3	0	290	0.012	0
CS-CNN-LSVM	0.996	0.996	0.996	0.996	243	2	0	291	0.008	0
CNN-RSVM	0.996	0.996	0.996	0.996	243	2	0	291	0.008	0
CS-CNN-RSVM	0.996	0.996	0.996	0.996	243	2	0	291	0.008	0

Fig. B.9: Accuracy of concatenated models

```

Accuracy : 0.9605263157894737
Precision 0.9605242627555001
Recall : 0.9605263157894737
F1-score : 0.9605128094072808
Confusion Matrix :
[[441 15]
 [ 18 362]]
      precision    recall  f1-score   support

      0         0.96      0.97      0.96         456
      1         0.96      0.95      0.96         380

   accuracy          0.96          836
  macro avg          0.96          836
weighted avg          0.96          836

cost-sensitive
Accuracy : 0.9581339712918661
Precision 0.9581274311330951
Recall : 0.9581339712918661
F1-score : 0.9581293192893746
Confusion Matrix :
[[439 17]
 [ 18 362]]
...
   accuracy          0.96          836
  macro avg          0.96          836
weighted avg          0.96          836

```

Fig. B.10: Evaluating the Model

A Decision Support System to Identify Spam Images Using Ensemble Convolutional Neural Network

Ginjupalli Vinayana and Gorli Ajay Kumar

Department of Computer Science Engineering

Sathyabama Institute of Science and Technology

Jepiaar Nagar, 600119, Chennai, India

vinayanaginjupalli@gmail.com ajay.kumar.gorli@gmail.com

Ms.R Anandha Sree and Dr. S. Saranya

Department of Computer Science Engineering

Sathyabama Institute of Science and Technology Jepiaar Nagar, 600119, Chennai, India

anandhasreer.cse@sathyabama.ac.in saran.aamec@gmail.com

Abstract— As the internet grows, we face more online threats, with spam emails making up a whopping 55% of our inboxes, as Symantec reports. This research focuses on fighting image spam, a sneaky tactic used to slip past regular filters. Using advanced machine learning methods, we trained VGG-16, ResNet 50, and a combination of both on different datasets, considering features like metadata, color, shape, and texture. The good news is that some of our models reached an impressive 99% accuracy with no false positives. This shows promise in strengthening our online security against evolving cyber threats.

Index Terms—Image spam, VGG-16, ResNet 50, CNN

I. INTRODUCTION

In today's digital realm, where more than half of our emails turn out to be spam, dealing with image spam has become a top priority. Image spam, using graphics to sneak past regular filters which are initially made for text spam. That's where this research comes in – we're using smart tech tools, specifically VGG-16 and ResNet 50, to up our defense game against these visual threats.

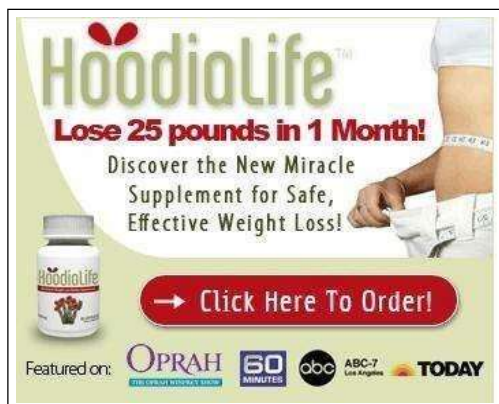


Fig. 1. This is how spam image looks like

VGG-16 and ResNet 50 may sound technical, but they are essentially advanced tools equipped to recognize patterns in images. VGG-16 maintains simplicity and uniformity in its approach, while ResNet 50 utilizes a unique learning method that delves into intricate details. The choice of these models is based on their capacity to automatically learn nuanced features, from basic shapes to complex textures, enabling precise identification of image spam.

The core of our strategy revolves around the capability of VGG-16 and ResNet 50 to discern subtle image features accurately. This means they can effectively differentiate between benign visuals and deceptive image spam. In the subsequent sections, we will delve into the testing procedures to showcase the efficacy of these models in fortifying our cybersecurity defenses against the persistent challenge of image spam in the expansive online environment.

The structure of this paper is outlined as follows: Section II presents an in-depth review of the relevant literature. In Section III, comprehensive details about three distinct datasets are provided. Section IV discusses preprocessing, as well as the VGG-16, ResNet-50, and the ensemble model. Evaluation metrics employed for assessment are outlined in Section V, along with the presentation of experimental results. Section VI delves into conclusions, and the paper concludes with a list of references.

I. LITERATURE SURVEY

Renita Arianti Hapsari and colleagues conducted a study to investigate the impact of employing the CNN method with the VGG-16 architecture on gender classification in facial images. The research delves into the

utilization of the CNN method for data processing and programming, the incorporation of the VGG-16 architecture for model construction, and the application of the TensorFlow library to invoke essential functions, such as optimization and statistical analysis. The evaluation of model performance includes the use of a confusion matrix to ascertain accuracy levels, with precision, accuracy, recall, and F-score identified as key metrics of interest. Notably, the findings indicate that face image classification achieved a commendable 94% accuracy through the combination of CNN and the VGG-16 architecture. [6]

In a similar vein, work done by Shrihari Rao and et.al presents an Image Classifier model that can identify spam images and classify them using pre-trained information sets like VGG16, VGG19, and MobileNet. The model, re-trained with a fresh dataset, shows promise in large-scale image classification using supervised learning using deep convolutional neural networks. It can handle dynamic data sets and delivers excellent accuracy.[16]

I. DATASET DESCRIPTION

A. Image Spam Hunter Dataset (ISH)

The Image Spam Hunter Dataset (ISH) is a treasure trove for researchers working on spotting image spam. It comprises real-world examples carefully chosen to reflect the diversity of spam. Researchers use ISH to test and enhance their spam-detecting tools, ensuring they can handle various sneaky spam images. In total, there are

810 ham and 929 spam images, with 879 unique spam images and 810 unique ham images.

B. Improved Dataset

Think of the Improved Dataset as an upgraded version of existing image spam datasets. It's refined to fix any issues and includes more varied samples with accurate labels. By addressing the limitations of earlier datasets, the Improved Dataset assists researchers in developing better ways to detect image spam. It contains a total of 1,029 spam images, generated by embedding spam text in ham images, as shown in Fig. 2 to trick existing models. The dataset comprises 975 unique images.

C. Dredze Image Spam Dataset

The Dredze Image Spam Dataset, curated by Professor Mark Dredze, is a collection designed for studying spam detection. Packed with real examples of image spam found in emails, researchers use this dataset to check the effectiveness of their spam-spotting tools in handling a

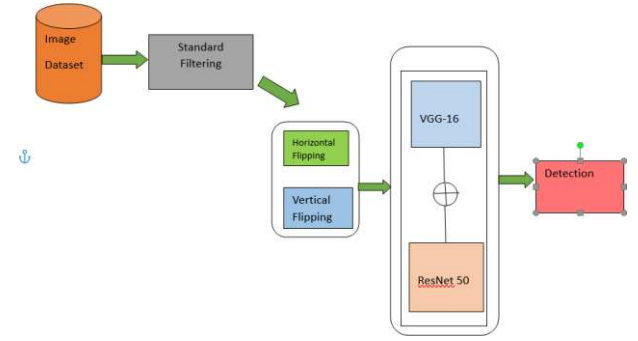


Fig. 2. Architecture

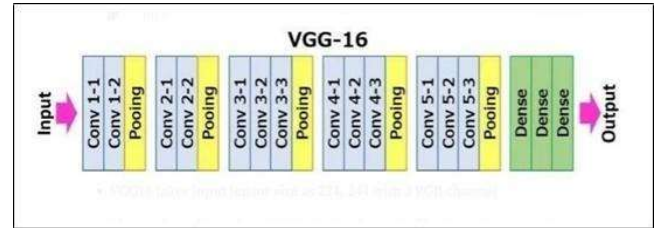


Fig. 3. VGG-16 Architecture

II. METHODOLOGY

A. Preprocessing

wide range of real-life spam situations.

This dataset contains three sets of images. Personal Ham has 2,021 images, with 1,517 unique images. Personal Spam consists of 3,298 images, with 1,274 unique images. Finally, the Spam Archive includes 16,028 files of various formats like JPEG, PNG, GIF, etc., with 3,039 unique images.

The three datasets employed in this study contain numerous instances of duplicate images and corrupt files. Initially, the corrupt files are identified and removed. To eliminate duplicate files, each image is transformed into a hash and stored. Consequently, when processing a potentially duplicate image, its hash is compared with existing ones. If a match is detected, the image is excluded. Ultimately, all unique images undergo normalization and resizing to meet the specified dimensions.

B. VGG-16

We’ve chosen the VGG-16 architecture as our main deep-learning model for sorting out image spam. This model, with its 16 layers—13 convolutional, 5 max-pooling, and 3 fully connected layers—makes use of Rectified Linear Unit (ReLU) activation functions after each convolutional and fully connected layer. This essentially adds a non-linear aspect to the network.

Moving on in our process, we take preprocessed image data, as explained in the preprocessing section, and feed it into the VGG-16 model. The input images are resized appropriately to match the required dimensions, ensuring they work seamlessly with the model’s structure.

Once the data is in, the training phase kicks off, where we fine-tune the model’s parameters to minimize the classification loss. Our dataset, which includes labeled examples of both spam and non-spam images, is split into

training and validation sets. The categorical cross-entropy loss function becomes the guiding force for optimization. Transfer learning is a key part of our strategy. We initiate the VGG-16 model with pre-trained weights from a large-scale image dataset like ImageNet. This kickstarts the model with a set of generic features from a diverse array of images before it hones in on the specific task of image spam classification.

To boost the model’s resilience and counteract overfitting, we employ data augmentation techniques during training. This includes introducing random transformations—rotations, flips, and shifts—to artificially diversify the training dataset.

Getting the right hyperparameters is crucial for optimal performance. So, we delve into fine-tuning—adjusting parameters like learning rate, batch size, and dropout rates systematically through experimentation. We’re guided by the performance of the validation set during this process.

The whole working process of VGG-16, from setting up the model to fine-tuning the hyperparameters, becomes the bedrock of our methodology for image spam classification. This thorough approach is designed to ensure a strong and efficient classification system.

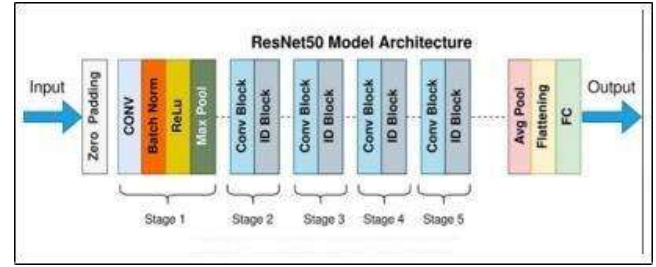


Fig. 4. ResNet 50 Architecture

We carefully set up ResNet-50, emphasizing its special architecture with those residual blocks and skip connections. This intentional design choice helps the model learn essential hierarchical features crucial for accurate image classification.

Just like we did with VGG-16, we fed preprocessed image data into ResNet-50, ensuring a smooth fit by resizing input images to meet the model’s requirements. This step ensures our data seamlessly integrates into the model’s structure.

The training journey begins, where our focus is on optimizing ResNet-50’s parameters to minimize classification loss. We split our dataset, loaded with labeled examples of both spam and non-spam images, into training and validation sets. To guide the optimization process, we rely

Algorithm 1: VGG-16 Algorithm for Image Spam Classification

1: **Function** *VGG16(image)* **is**

```

2: Load pre-trained VGG-16 model weights
3: Preprocess input image
   // Feature Extraction
4: for each convolutional block  $i$  do
5:     Apply convolutional layer  $i$  to the image
6:     Apply ReLU activation function
7:     Apply max pooling
   // Fully Connected Layers
8: Flatten the output from the convolutional
   blocks
9: for each fully connected layer  $j$  do
10:     Apply fully connected layer  $j$ 
11:     Apply ReLU activation function
12:     Apply dropout regularization
   // Output Layer
13: Apply softmax activation function for
   classification
14: return Class predictions

```

C. ResNet 50

In our research to identify and classify image spam, ResNet-50 steps into the spotlight as a powerful player. What makes ResNet-50 stand out is its unique 50-layer design, which incorporates clever residual blocks and skip connections to tackle the tricky vanishing gradient problem, making it effective for training deep networks.

on the categorical cross-entropy loss function.

We bring in transfer learning as a strategic move by starting ResNet-50 with pre-trained weights from a rich image dataset like ImageNet. This initial training empowers the model to grasp general features before refining its understanding for the specific task of image spam classification.

Fine-tuning hyperparameters becomes a key step for reaching optimal performance. Parameters like learning rate, batch size, and dropout rates go through a careful tuning process, guided by hands-on experimentation and performance evaluations on the validation set.

The entire working process of ResNet-50, as laid out above, is a crucial part of our strategy for image spam classification. This all-encompassing approach, covering model setup and hyperparameter fine-tuning, is designed with precision to ensure the creation of a reliable and effective image spam classification system.

D. Ensembled Model

In our exploration of refining image spam classification, our research introduces an ensemble approach by merging the capabilities of ResNet-50 and VGG-16. This strategic integration aims to enhance the accuracy, robustness, and overall effectiveness of our classification model.

From the figure, we begin by configuring ResNet-50 and VGG-16, loading pre-trained weights, and carefully preprocessing input images. The ensemble prediction is created by combining individual predictions, utilizing

Algorithm 2: ResNet-50 Algorithm for Image Spam Classification

```

1: Function ResNet50(image) is
2:   Load pre-trained ResNet-50 model weights
3: Preprocess input image // Feature Extraction
4: for each residual block  $i$  do
5:   Apply residual block  $i$  to the image
6:   Apply skip connections
7:   Apply batch normalization
8:   Apply ReLU activation function // Global Average Pooling
9:   Apply global average pooling to reduce spatial dimensions
   // Fully Connected Layers
10: for each fully connected layer  $j$  do
11:   Apply fully connected layer  $j$ 
12:   Apply ReLU activation function
13:   Apply dropout regularization // Output Layer
14:   Apply softmax activation function for classification
15: return Class predictions

```

Algorithm 3: Ensemble Algorithm for Image Spam Classification using ResNet-50 and VGG-16

```

3:   Preprocess input image

    // Feature Extraction
4:   for each residual block  $i$  do
5:       Apply residual block  $i$  to the image
6:       Apply skip connections
7:       Apply batch normalization
8:       Apply ReLU activation function

    // Global Average Pooling
9:   Apply global average pooling to reduce spatial dimensions

    // Fully Connected Layers
10:  for each fully connected layer  $j$  do
11:      Apply fully connected layer  $j$ 
12:      Apply ReLU activation function
13:      Apply dropout regularization

    // Output Layer
14:  Apply softmax activation function for classification
15:  return Class predictions

```

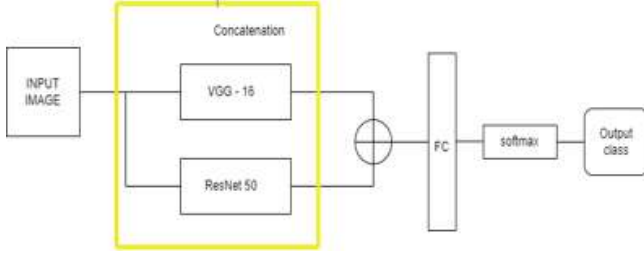


Fig. 5. Ensembling model

techniques such as averaging or voting. Following this, the final classification decision is based on these combined predictions.

Our experimental setup involves individually training ResNet-50 and VGG-16 on the image spam dataset, incorporating detailed hyperparameter fine-tuning. The ensemble approach seamlessly integrates into both the training and inference phases.

Performance evaluation centers on a comparative analysis involving the ensemble model against the standalone ResNet-50 and VGG-16 models. Our goal is to highlight the ensemble's superiority, showcasing improved accuracy, heightened robustness, and superior generalization across diverse image spam samples.

II. EVALUATION METRICS

In this study, we utilize standard metrics to assess the performance of our image spam classification system. The results are presented in the following table.

```

1: Function Ensemble(image) is
    // Use ResNet-50 for prediction
2:   predictions_resnet ← ResNet50(image)

    // Use VGG-16 for prediction
3:   predictions_vgg ← VGG16(image)

    // Combine predictions
4:   combined_predictions ←
       CombinePredictions(predictions_resnet,
                           predictions_vgg)

5:   return Combined predictions

```

Sl.No	Model	Time Taken	Accuracy
1	VGG19	1.4 hours	94.77
2	ResNet	1.2 hours	89.21
3	AlexNet	1.2 hours	87.02
4	GoogleNet	1.5 hours	90.08

TABLE I

COMPARING METRICS OF OTHER MODELS

selected metrics include:

- **Accuracy:** The ratio of correctly classified images (spam and normal) to the total image count.
- **Precision:** The proportion of true positive predictions to the total predicted positive cases, measuring the precision of spam predictions.
- **Recall:** The ratio of true positive predictions to the total actual positive instances, indicates the system's ability to accurately identify spam.
- **F1-Score:** The harmonic mean of precision and recall, provides a balanced measure between the two metrics.
- **False Positive Rate (FPR):** The proportion of false positive predictions to the total actual negative instances, indicating the rate of normal images incor-
- **False Negative Rate (FNR):** The ratio of false negative predictions to the total actual positive instances, signifying the rate of spam images inaccurately predicted as normal.

These metrics are calculated using terms from the confusion matrix, including True Positive (TP), False Negative (FN), False Positive (FP), and True Negative (TN). TP represents accurately predicted spam images, FN represents spam images incorrectly predicted as normal, FP represents normal images inaccurately predicted as spam, and TN represents accurately predicted normal images.

The confusion matrix and these metrics provide a comprehensive evaluation of the image spam classification system's performance, enabling an assessment of accuracy, precision, recall, and overall effectiveness.

III. EXPERIMENTAL RESULTS AND DISCUSSIONS

In this project, the proposed models are executed using the python libraries and few of the codes we considered are available in GitHub. Regulation of dropout rates is used to prevent over-fitting. The datasets splitted in this model are splitted into 70:30 and 80:20 training and testing sets. In the above literature review, the image spam models are taken into consideration and its performance is represented in a tabular form for the comparison. For the purpose of training , images are been reduced into 156x156 resolution which is determined by the model's training and testing across a range of input sizes. By the method of Ensembling of VGG-16 and ResNet50 gave a better accuracy and result for the image spam detection, it gave an accuracy of 99.03 .

IV. CONCLUSION

In this work, we had shown the optimized results of ensembling VGG-16, Resnet50 employing Deep Convolutional Neural Networks and Hybrid Machine Learning models for studying Vgg-16 and Resnet50 datasets for the image spam classification. We have also shown the efficiency of machine by assigning balanced classes and transferred learning. We took multiple pre-trained models into consideration. While few of the models efficiency is better in proposed models than the the existing models and some them did not perform well. These models may capture several aspects of the data and have independent structures. It's crucial to test out various ensemble approaches and assess how they affect performance. In this way it can be taken into consideration to develop a well trained image spam classifier. Additional features like rotation, mirroring should also be included while training the model, the impacts of unfriendly samples , which will mislead the model to make false predictions, can be investigated.

V. ACKNOWLEDGMENT

This work was partially supported by Sathyabama Institute of Science and Technology. We are also grateful to the Dr. J. Albert Mayan, M.E., Ph.D ; Dr. M. D. Anto Praveena, M.E., Ph.d., Department of Computer Science and Engineering, Sathyabama School of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai for supporting and constant assisting the research.

VI.

REFERENCES

- [1] Md Manjurul Ahsan, Muhammad Ramiz Uddin, Mithila Farjana, Ahmed Nazmus Sakib, Khondhaker Al Momin, and Shahana Akter Luna. Image data collection and implementation of deep learning-based model in detecting monkeypox disease using modified vgg16, 2022.
- [2] Laith Alzubaidi, Laith Alzubaidi, Jinglan Zhang, Jinglan Zhang, Amjad J. Humaidi, Amjad J. Humaidi, Ayad Q. Al-Dujaili, Ayad Al-Dujaili, Ye Duan, Ye Duan, Omran Al-Shamma, Omran Al-Shamma, José Santamaría, J Santamaría, José Santamaría, Mohammed A. Fadhel, Mohammed A. Fadhel, Muthana Al-Amidie, Muthana Al-Amidie, Laith Farhan, and Laith Farhan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal of Big Data*, 2021.
- [3] Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. Review of image classification algorithms based on convolutional neural networks. *Remote Sensing*, 13(2A2), 2021.
- [4] Chastine Fatichah, Wildan F. Lazuardi, Dini A. Navastara, Nanik Sucati, and Abdul Munif. A content filtering from spam posts on social media using weighted multimodal approach. *Journal of Computer Science*, 17(1):55–66, Jan 2021.
- [5] Renita Hapsari and Aji Purwinarko. Implementation of convolutional neural network algorithm using vgg-16 architecture for image classification in facial images. *Recursive Journal of Informatics*, 1(2):83–92, Sep. 2023.
- [6] Sanaa Kaddoura, Sanaa Kaddoura, Ganesh Chandrasekaran, Ganesh Chandrasekaran, Daniela Elena Popescu, Daniela Elena Popescu, D. Jude Hemantha, D. Jude Hemanth, and D. Jude Hemanth. A systematic literature review on spam content detection and classification. *PeerJ Computer Science*, 2022.
- [7] Arif Laksito, Arif Dwi Laksito, Muhammad Royyan Saputra, and Muhammad Royyan Saputra. Content based vgg16 image extraction recommendation. *Jurnal RESTI (Rekayasa Sistem dan Teknologi Informasi)*, 2022.
- [8] Deepika Mallampati and Dr. Nagaratna P Hegde. Image spam detection in e-mails using grasshoppers optimization technique. In *2023 International Conference on Computer, Electronics Electrical Engineering their Applications (IC2E3)*, pages 1–5, 2023.
- [9] Sheldon Mascarenhas and Mukul Agarwal. A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification. In *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, volume 1, pages 96–99, 2021.
- [10] Sheldon Mascarenhas, Sheldon Mascarenhas, Mukul Agarwal, and Mukul Agarwal. A comparison between vgg16, vgg19 and resnet50 architecture frameworks for image classification. *2021 International Conference on Disruptive Technologies for Multi-Disciplinary Research and Applications (CENTCON)*, 2021.
- [11] N. Nandhagopal P. Siva Satya Sreedhar. Classification similarity network model for image fusion using resnet50 and googlenet. *Intelligent Automation & Soft Computing*, 31(3):1331–1344, 2022.
- [12] Rakesh K R, G R Namita, and Rohit Kulkarni. Image recognition, classification and analysis using convolutional neural networks. In *2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT)*, pages 1–4, 2022.
- [13] Rakesh K R, Kottada Rakesh, G R Namita, G R Namita, Rohit Kulkarni, and Rohit Kulkarni. Image recognition, classification and analysis using convolutional neural networks. *International Conference on Electrical Engineering and Information Communication Technology*, null.
- [14] Mallikka Rajalingam*, Mallikka Rajalingam*, M. Balamurugan, and M. Balamurugan. An observation and experimental evaluation of image spam detection. *International journal of recent technology and engineering*, null.
- [15] S. Krishna Rao, Shrihari Rao, Radhakrishnan Gopalapillai, and Radhakrishnan Gopalapillai. Effective spam image classification using cnn and transfer learning. *null*, 2019.
- [16] Ali Sharif Razavian, Ali Sharif Razavian, Hossein Azizpour, Hossein Azizpour, Josephine Sullivan, Josephine Sullivan, Stefan Carlsson, and Stefan Carlsson. Cnn features off-the-shelf: An astounding baseline for recognition. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2014.
- [17] Tazmina Sharmin, Tazmina Sharmin, Fabio Di Troia, Fabio Di Troia, Katerina Potika, Katerina Potika, Mark Stamp, and Mark Stamp. Convolutional neural networks for image spam detection. *Information Security Journal: A Global Perspective*, 2020.
- Hoo-Chang Shin, Hoo-Chang Shin, Holger R. Roth, Holger R. Roth, Mingchen Gao, Mingchen Gao, Le Lü, Le Lu, Ziyue Xu, Ziyue Xu, Isabella Nogues, Isabella Nogues, Jianhua Yao, Jianhua Yao, Daniel J. Mollura, Daniel J. Mollura, Ronald M. Summers, and Ronald M.

Summers. Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning. *arXiv: Computer Vision and Pattern Recognition*, 2016.

- [18] Angom Buboo Singh and Khumanthem Manglem Singh. Application of error level analysis in image spam classification using deep learning model. *PLoS ONE*, 2023.
- [19] Anuraj Singh, Anurag Srivastava, and Evi Agarwal. A non-content based optimized approach for image spam detection. In *2022 IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, pages 1–6, 2022.
- [20] Anuraj Singh, Anurag Srivastava, and Evi Agarwal. A non-content based optimized approach for image spam detection. *2022 IEEE Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI)*, 2022.
- [21] Dhananjay Theckedath, Dhananjay Theckedath, R. R. Sedamkar, and R. R. Sedamkar. Detecting affect states using vgg16, resnet50 and se-resnet50 networks. *SN Computer Science*, 2020.
- [22] Zhibo Zhang, Ernesto Damiani, Hussam Al Hamadi, Chan Yeob Yeun, and Fatma Taher. Explainable artificial intelligence to detect image spam using convolutional neural network. *International Conference Control and Robots*, 2022.

D. Certificates



