A REAL TIME PROJECT REPORT ON

IMAGE CLASSIFICATION USING CNN

Submitted to

Jawaharlal Nehru Technological University, Hyderabad in partial fulfillment of the requirements for the award of the degree

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

By

Anandapu Rahul	**(22831A0507)**
Ankolu ChandraSekhar	**(22831A0511)**
Balineni Nanda Sai	**(22831A0518)**
Baanotu Nandu	**(23835A0501)**
Gannoju Pooja	**(23835A0505)**

Under the Esteemed Guidance of

Mrs. G .RASHMI

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GURU NANAK INSTITUTE OF TECHNOLOGY

(Affiliated to JNTU - Hyderabad)

Ranga Reddy District – 501506

2023-2024

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GURU NANAK INSTITUTE OF TECHNOLOGY

(Affiliated to JNTU - Hyderabad)

Ranga Reddy District - 501506

CERTIFICATE

This is to certify that the project entitled "**Image classification using CNN"** is being presented with a report by **Anandapu Rahul (22831A0507), Ankolu Chandrasekhar (22831A0511), Balineni Nanda Sai (22831A0518), Baanotu Nandu (23835A0501), Gannoju Pooja(23835A0505)** in partial fulfilment for the award of Degree of **Bachelor of Technology in Computer Science of Engineering**, to Jawaharlal Nehru Technological University, Hyderabad.

Mrs. G . RASHMI
RTP Coordinator

Dr. B . SANTHOSH KUMAR
Professor and Head of Department

GURU NANAK INSTITUTE OF TECHNOLOGY

Ibrahimpatnam, R.R. Dist. – 501506.

VISION OF GNIT

To be a world-class educational and research institution in the service of humanity by promoting high quality engineering and management education.

MISSION OF GNIT

- Imbibe soft skills and technical skills.
- Develop the faculty to reach international standards.
- Maintain high academic standards and teaching quality that promotes the typical thinking and independent judgment,
- Promote research, innovation and product development by collaboration with repute
- Foreign universities.

GURU NANAK INSTITUTE OF TECHNOLOGY

Ibrahimpatnam, R.R. Dist. -501506.

VISION OF DEPARTMENT

To be recognized as a leading department of Computer science and Engineering in the region by students, employers and be known for leadership, Ethics, and commitment to fostering quality teaching-learning, research, and innovation.

MISSION OF DEPARTMENT

- Nurture young individuals into knowledgeable, skill-full and ethical professionals in their Pursuit of computer science and Engineering.
- Nurture the faculty to expose them to world-class infrastructure.
- Sustain high performance by excellence in teaching, research and innovations.
- Extensive partnerships and collaborations with foreign universities for technology upgradation.
- Develop industry-interaction for innovation and product development.

Program Educational Objectives (PSO's)

PSO-1: Graduates shall have the ability to apply knowledge and technical skills in emerging areas of Computer Science and Engineering for higher studies, research, employability, product development and handle realistic problems.

PSO-2: Graduates shall maintain ethical conduct, a sense of responsibility to serve society and protect the environment.

PSO-3: Graduates shall possess academic excellence with innovative insight, soft skills, managerial skills, leadership qualities, knowledge of contemporary issues for successful professional career.

Program Outcomes (PO's)

PO-1: Engineering knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO-2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.

PO-3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet specified needs with appropriate consideration for public health and safety, and the cultural, societal, and environmental considerations.

PO-4: Conduct investigations of complex problems: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information

PO-5: Modern tool usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modelling to complex engineering activities with an understanding of the limitations.

PO-6: The engineer and society: Apply reasoning informed by contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO-7: Environment and sustainability: Understand the impact of professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO-8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of engineering practice.

PO-9: Individual and teamwork: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

PO-10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

PO-11: Project management and finance: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

PO-12: Life-long learning: Recognize the need for and have the preparations and ability to engage in independent and life-long learning in the broadest context oftechnological change.

PROGRAM SPECIFIC OUTCOMES(PSO'S)

PSO-1: Professional Skills: The ability to understand the principles and working of computer systems. Students can assess the hardware and software aspects of computer systems.

PSO-2: Problem Solving Skills: The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for business success.

PSO-3: Successful Career and Entrepreneurship: The ability to employ modem computer languages, environments, and platforms in creating innovative career paths to be an entrepreneur, and a zest for higher studies

DECLARATION

We hereby declare that the major project report entitled **"Image Classification using CNN"** is the work done by **Anandapu Rahul, Ankolu Chandrasekhar, Balineni Nanda Sai, Baanotu Nandu and GannojuPooja** bearing the rollno.'s **22831A0507,22831A0511 22831A0518,23835A0501,23835A0505**towards the fulfillment of the requirement for the award of the Degree of **Bachelor of Technology in Computer Science & Engineering,** to **Jawaharlal Nehru Technological University, Hyderabad,** is the result of the work carried out under the guidance **Mrs. G. Rashmi , RTP Coordinator, Guru Nanak Institute of Technology, Hyderabad.**

We further declare that this project report has not been previously submitted either in part or full for the award of any degree or diploma by any organization or university.

Anandapu Rahul	**22831A0507**
Ankolu ChandraSekhar	**22831A0511**
Balineni Nanda Sai	**22831A0518**
Baanotu Nandu	**23835A0501**
Gannoju Pooja	**23835A0505**

ACKNOWLEDGEMNT

“Task successful” makes everyone happy. But happiness would be gold without glitter if we didn’t state the persons who have supported us to make it a success for us. We would like to express our sincere thanks and gratitude to our Principal, **Dr. S. SREENATHA REDDY** and **Dr. B. SANTHOSH KUMAR,** Professor and head of Department of **Computer Science and Engineering, Guru Nanak Institute of Technology** for having guided me in developing the requisite capabilities for taking up this project. We especially thank our RTP Coordinator **Mrs. G. RASHMI** , Assistant Professor, for his constant guidance in every stage of the project. We would also like to thank all our lecturers for helping us in every possible way whenever the need arose.On a more personal note, we thank our beloved parents and friends for their moral support during the course of our project.

ABSTRACT

Image classification is a fundamental task in computer vision, involving the categorization of images into predefined classes. Convolutional Neural Networks (CNNs) have emerged as a powerful tool for this task due to their ability to automatically learn hierarchical feature representations from raw pixel data. This paper explores the application of CNNs for image classification, detailing the architecture, training process, and evaluation metrics. Key components of a CNN, such as convolutional layers, pooling layers, and fully connected layers, are discussed. The study demonstrates the effectiveness of CNNs by training models on benchmark datasets like CIFAR-10 and ImageNet, achieving high accuracy rates and robust performance. The results underscore the advantages of CNNs In capturing spatial hierarchies and reducing the need for manual feature extraction, making them a preferred choice for image classification tasks in various domains. Future work includes exploring deeper architectures, transfer learning, and fine - tuning techniques to further enhance performance and generalizability.

LIST OF CONTENTS

LIST OFABBREVIATIONS

S.NO	ABBREVIATION	EXPANSION
1.	CNN	Convolutional Neural Network
2.	ML	Machine Learning
3.	PDF	Portable Document Format

CHAPTER 1

INTRODUCTION

1.1 GENERAL

Image classification is a pivotal task in computer vision, involving the categorization of images into predefined classes. This task underpins numerous applications, ranging from autonomous vehicles and medical diagnosis to facial recognition and content recommendation systems. Traditional image classification methods relied heavily on manual feature extraction, requiring domain-specific knowledge and extensive preprocessing. However, the advent of Convolutional Neural Networks (CNNs) has revolutionized this field by automating feature extraction and learning hierarchical representations of images directly from raw pixel data.

CNNs are a class of deep learning models specifically designed to process and analyze visual data. They are composed of multiple layers, including convolutional layers, pooling layers, and fully connected layers, each serving a distinct function. Convolutional layers apply filters to the input image to capture local patterns, such as edges and textures. Pooling layers reduce the spatial dimensions, thereby decreasing the computational load and highlighting the most significant features. Fully connected layers at the end of the network integrate these features to classify the image into one of the predefined categories.

The strength of CNNs lies in their ability to learn and recognize intricate patterns and spatial hierarchies within images. Unlike traditional methods, which required handcrafted features, CNNs learn to identify relevant features through backpropagation, adjusting their parameters to minimize classification errors. This ability to learn from data makes CNNs highly effective for various image classification tasks, achieving state-of-the-art performance on benchmark datasets like CIFAR-10, MNIST, and ImageNet.

In recent years, the development of deeper and more complex CNN architectures, such as AlexNet, VGGNet, GoogLeNet, and ResNet, has further pushed the boundaries of image classification. These architectures have demonstrated that increasing the depth and complexity of networks can lead to significant improvements in accuracy and generalization. Moreover, advancements in training techniques, data augmentation, and the availability of large-scale labeled datasets have contributed to the widespread adoption and success of CNNs in image classification

In summary, CNNs have transformed image classification by automating feature extraction and learning, enabling accurate and efficient analysis of visual data. This introduction provides a foundation for understanding the principles and applications of CNNs in the context of image classification, setting the stage for further exploration of their architectures and capabilities.

Literature Survey

1.1 Literature Survey on Image Classification Using CNN

1. **Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton (2012)**

2. **Paper: "ImageNet Classification with Deep Convolutional Neural Networks"**

This seminal work introduced AlexNet, a deep convolutional neural network that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012 by a significant margin. The architecture consists of five convolutional layers, some of which are followed by maxpooling layers, and three fully connected layers with a final 1000-way softmax. The key contributions include the use of ReLU activation functions to accelerate training, dropout to reduce overfitting, and data augmentation techniques to improve generalization. This paper demonstrated the effectiveness of deep CNNs in handling large-scale image classification tasks and spurred further research in the field.

2.Karen Simonyan and Andrew Zisserman (2014)

Paper: "Very Deep Convolutional Networks for Large-Scale Image Recognition"

Simonyan and Zisserman introduced the VGGNet architecture, which significantly deepened the network compared to AlexNet by using very small (3×3) convolution filters, which allowed for increasing depth while keeping the computational requirements manageable. VGGNet demonstrated that the depth of the network is a critical factor in achieving high performance, achieving top results on the ImageNet dataset. Their work highlighted the importance of network depth and simplicity in design, influencing subsequent CNN architectures.

3.Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016)

Paper: "Deep Residual Learning for Image Recognition"

This paper introduced ResNet (Residual Networks), which addressed the degradation problem in deep networks where increasing depth led to higher training error. ResNet uses residual blocks that enable the training of extremely deep networks (e.g., 152 layers) by allowing gradients to flow through shortcut connections, bypassing one or more layers. ResNet won the ILSVRC 2015 and significantly improved the state of the art in image classification, object detection, and segmentation. The concept of residual learning has become a foundational technique in modern CNN architectures.

4.Christian Szegedy et al. (2015)

Paper: "Going Deeper with Convolutions"

Szegedy and colleagues introduced the Inception architecture (also known as GoogLeNet), which uses a novel approach to improve computational efficiency and reduce the number of parameters compared to traditional CNNs. The Inception modules utilize multiple convolutional filters of different sizes in parallel, capturing features at various scales and increasing the network's ability to generalize. This architecture also employs dimensionality reduction using 1×1 convolutions to keep the computational load manageable. Inception models achieved top performance in the ILSVRC 2014 and have influenced many subsequent network designs.

Conclusion

These four works have significantly contributed to the advancement of image classification using CNNs. From AlexNet's breakthrough in deep learning applications to VGGNet's emphasis on network depth, ResNet's introduction of residual learning, and Inception's efficient architecture design, each has provided key insights and innovations that have shaped modern computer vision research and applications.

CHAPTER 2

SCOPE OF THE PROJECT

2.1 SCOPE OF THE PROJECT

The project on image classification using Convolutional Neural Networks (CNNs) aims to develop a sophisticated system capable of accurately categorizing images into predefined classes. This project includes collecting and preprocessing diverse datasets, designing and implementing effective CNN architectures, and optimizing model performance through training and hyperparameter tuning. Rigorous evaluation will be conducted using metrics like accuracy, precision, and recall to ensure the model's robustness and generalization capability. The project also encompasses deploying the trained model in practical applications, such as object detection and medical image analysis, demonstrating its real-world utility. Comprehensive documentation and reporting will provide detailed insights into the project's methodology and outcomes, facilitating future research and development. Additionally, the project may explore advanced techniques like transfer learning and real-time applications, extending its impact and applicability. By addressing these components, the project seeks to contribute significantly to the field of computer vision, showcasing the effectiveness of CNNs in automating and enhancing image classification tasks.

2.2 PROBLEM STATEMENT

The problem of image classification involves accurately categorizing images into predefined classes, which is essential for applications such as facial recognition, autonomous driving, and medical diagnosis. Traditional methods relying on manual feature extraction are inefficient and often inadequate in handling large and complex datasets. This project aims to address these challenges by leveraging Convolutional Neural Networks (CNNs), which can automatically learn and extract hierarchical features from images. The primary objective is to design, implement, and optimize a CNN model that achieves high accuracy and robustness in classifying images from diverse datasets. The project will involve data collection, preprocessing, model training, and rigorous evaluation to ensure the model's effectiveness. Additionally, the deployment of the model in practical scenarios will demonstrate its real-world applicability. By solving this problem, the project seeks to enhance the efficiency and accuracy of image classification tasks, contributing to advancements in computer vision and artificial intelligence.

2.3 EXISTING SYSTEM

Existing Systems of Image Classification Using CNN

In recent years, Convolutional Neural Networks (CNNs) have become the cornerstone of image classification tasks, offering significant improvements over traditional methods. Various CNN architectures have been developed and widely adopted in both academic research and industry applications, each contributing unique advancements in the field.

1. AlexNet:

Developed by: Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012.

Overview: AlexNet marked a breakthrough in deep learning by winning the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012. It demonstrated the effectiveness of deep CNNs in handling large-scale image classification, employing ReLU activations, dropout for regularization, and data augmentation.

2. VGGNet:

Developed by: Karen Simonyan and Andrew Zisserman in 2014.

Overview: VGGNet introduced a very deep network with 16-19 layers using small (3×3) convolutional filters. This architecture emphasized depth and simplicity, achieving high performance on the ImageNet dataset and influencing subsequent CNN designs.

3. GoogLeNet (Inception):

Developed by: Christian Szegedy et al. in 2015.

Overview: The Inception architecture, also known as GoogLeNet, focused on computational efficiency and reducing the number of parameters. It used inception modules with parallel convolutional filters of different sizes, capturing multi-scale features within the same layer.

4. ResNet (Residual Networks):

Developed by: Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun in 2015.

Overview: ResNet addressed the degradation problem in deep networks by introducing residual learning through shortcut connections. This allowed for training extremely deep networks (e.g., 152 layers) and won the ILSVRC 2015 with remarkable performance.

5. DenseNet:

Developed by: Gao Huang et al. in 2017.

Overview: DenseNet introduced dense blocks where each layer receives inputs from all previous layers, enhancing feature reuse and improving gradient flow. This architecture demonstrated efficient parameter usage and state-of-the-art performance on various benchmarks.

6. EfficientNet:

Developed by: Mingxing Tan and Quoc V. Le in 2019.

Overview: EfficientNet systematically scales up network dimensions (depth, width, and resolution) using a compound scaling method. This approach achieved superior performance with fewer parameters and less computational cost compared to previous models.

These existing systems have significantly advanced the field of image classification, each contributing innovations in architecture, training techniques, and computational efficiency. They provide robust frameworks for a wide range of applications, from everyday consumer technology to specialized industry solutions.

EXISTING SYSTEM DISADVANTAGES

Disadvantages of Existing Systems in Image Classification Using CNN

While Convolutional Neural Networks (CNNs) have revolutionized image classification, several limitations and disadvantages are associated with existing systems:

1. Computational Complexity:

Issue: CNNs, especially deeper architectures like VGGNet, ResNet, and DenseNet, require significant computational resources for training and inference. This includes high-performance GPUs and large memory capacity.

Impact: High computational cost limits accessibility for smaller organizations and individual researchers without extensive hardware resources.

2. Large Data Requirements:

Issue: CNNs generally require vast amounts of labeled data to achieve high accuracy and generalization. Collecting and labeling such datasets can be labor-intensive and expensive.

Impact: Performance may degrade with smaller datasets, and overfitting can become a significant issue without sufficient data.

3. Training Time:

Issue: Training deep CNNs can be time-consuming, often taking days or even weeks depending on the network's complexity and the dataset size.

Impact: Extended training times delay model deployment and iterative development processes, which can be critical in fast-paced environments.

4. Overfitting:

Issue: With their high capacity to learn, CNNs can easily overfit to the training data, especially when the dataset is not sufficiently large or diverse.

Impact: Overfitted models perform poorly on new, unseen data, reducing their practical utility in real-world applications.

5. Interpretability:

Issue: CNNs operate as black-box models, making it difficult to interpret how they make decisions or which features they consider important.

Impact: Lack of interpretability hinders trust and adoption in critical applications like medical diagnosis and autonomous driving, where understanding model decisions is essential.

6. Hyperparameter Tuning:

Issue: CNNs have numerous hyperparameters (e.g., learning rate, batch size, number of layers) that need to be carefully tuned for optimal performance.

Impact: Hyperparameter tuning is often a manual and computationally expensive process, requiring expertise and iterative experimentation.

7. Energy Consumption:

Issue: Training and running deep CNNs consume significant energy, leading to high operational costs and environmental concerns.

Impact: Energy inefficiency is a growing concern, particularly with the increasing demand for large-scale AI applications.

8. Generalization to New Domains:

Issue: CNNs trained on specific datasets may not generalize well to images from different domains or with different characteristics.

Impact: Poor generalization limits the versatility and applicability of CNN models across diverse real-world scenarios.

Despite these challenges, ongoing research is addressing these limitations through advancements like transfer learning, more efficient architectures, automated hyperparameter tuning, and methods to enhance interpretability and reduce overfitting.

2.4 PROPOSED SYSTEM

The recruiters to find the best individuals for a certain position for any job opportunity. Technology should automatically create useful resume data such as education, abilities, achievements, and experience. This initiative aims to reduce recruitment time and prejudice throughout the selection process by evaluating candidates based on several characteristics of their resume.

PROPOSED SYSTEM ADVANTAGES

Proposed System Advantages of Image Classification Using CNN

Implementing image classification using Convolutional Neural Networks (CNNs) offers several distinct advantages over traditional methods, making it a powerful and preferred approach in various applications:

1. Automatic Feature Extraction:
Advantage: CNNs automatically learn hierarchical features from raw pixel data without the need for manual feature extraction. This capability reduces human effort and ensures that relevant features are extracted optimally.

2. Ability to Learn Complex Patterns:
Advantage: CNNs excel at capturing intricate spatial relationships and patterns within images, such as textures, edges, and shapes. This enables them to achieve higher accuracy in distinguishing between classes, even in complex datasets.

3. Hierarchical Feature Representation:
Advantage: CNN architectures, with layers like convolutional and pooling layers, create a hierarchical representation of features. This hierarchical approach allows the model to understand both low-level and high-level features progressively, improving overall classification accuracy.

4. Flexibility and Adaptability:
Advantage: CNNs can be adapted and fine-tuned for different image classification tasks by adjusting network architectures, adding regularization techniques, or using transfer learning from pretrained models. This flexibility makes CNNs suitable for a wide range of applications and domains.

5. State-of-the-art Performance:
Advantage: CNNs have consistently outperformed traditional methods and achieved state-of-theart performance on benchmark datasets like ImageNet. Their ability to leverage large datasets and complex architectures allows them to push the boundaries of accuracy in image classification tasks.

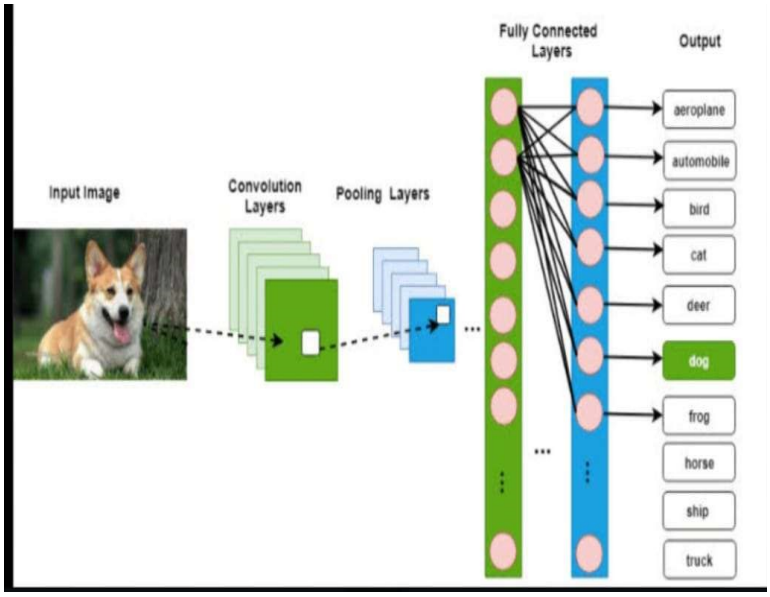
6. Real-time Processing Capabilities:
Advantage: With advancements in hardware acceleration (e.g., GPUs, TPUs), CNNs can process images in real-time, making them suitable for applications requiring quick decision-making, such as autonomous vehicles and medical diagnostics.

7. Scalability:
Advantage: CNN-based models can scale effectively with increasing amounts of data and computational resources. This scalability ensures that performance improvements can be achieved by leveraging larger datasets or more powerful hardware.

8. Wide Range of Applications:
Advantage: CNNs have been successfully applied across diverse domains including healthcare (e.g., medical imaging analysis), security (e.g., facial recognition), retail (e.g., product recognition), and entertainment (e.g., content recommendation). Their versatility makes them valuable across various industries.

Implementing an image classification system using CNNs leverages these advantages to enhance accuracy, efficiency, and applicability in solving real-world challenges across different sectors, paving the way for advanced applications in computer vision and machine learning.

2.5 SYSTEM ARCHITECTURE



F igure 5 .1- System Architecture of Image Classification using CNN

The system architecture for classifying images of dogs and cats using CNN involves: Data Collection and Preprocessing (resizing, normalization, augmentation), Model Design (convolutional layers, pooling layers, fully connected layers), Training (forward propagation, loss calculation, backpropagation), Evaluation (accuracy, precision, recall), and Deployment (integration into applications, real-time inference, monitoring).

CHAPTER 3

PROJECT DESCRIPTION

3.1 Project Description

Project Description: Image Classification Using Convolutional Neural Networks (CNNs)

This project aims to develop a robust CNN model to accurately classify images into predefined categories. Utilizing a dataset such as CIFAR-10 or MNIST, the process begins with data preprocessing, including normalization and augmentation, to enhance model generalization.

The CNN architecture will consist of convolutional layers to extract features, activation functions like ReLU for non-linearity, pooling layers to reduce spatial dimensions, and fully connected layers for final classification. The model will be trained using a cross-entropy loss function and an optimizer such as Adam or SGD, with techniques like dropout to prevent overfitting.

Evaluation metrics including accuracy, precision, recall, and F1-score will be used to assess model performance. Visualization of training progress and feature maps will aid in understanding and improving the model. Finally, the trained model will be deployed using TensorFlow Serving or Flask, allowing for real-time image classification via an API endpoint. This project demonstrates the efficacy of CNNs in image classification tasks and provides a foundation for future advancements and applications.

3.2 MODULES IN THE PROJECT

Image classification using Convolutional Neural Networks (CNNs) typically involves several key modules, each responsible for a specific part of the process. Here's a breakdown of the main modules:

1. Data Preprocessing Module:

Data Augmentation: Techniques such as rotation, scaling, flipping, and cropping to artificially expand the dataset and improve the model's robustness.

Normalization: Scaling pixel values to a range (e.g., 0 to 1) to ensure consistent input to the network.

2. Input Module:

Data Loading: Efficiently loading and batching images from the dataset, often using libraries like TensorFlow, PyTorch, or Keras.

Input Layer: The initial layer in the CNN which accepts the input image data.

3. Convolutional Layer:

Convolutional Operations: Applying various filters/kernels to the input image to extract features such as edges, textures, and shapes.

Activation Function: Typically, ReLU (Rectified Linear Unit) is applied to introduce non-linearity.

4. Pooling Layer:

Max Pooling / Average Pooling: Reducing the spatial dimensions of the feature maps while retaining the most important information, helping to reduce computational load and control overfitting.

5. Fully Connected Layer:

Dense Layers: Flattening the output from the convolutional/pooling layers and passing it through one or more fully connected layers to make the final classification.

Activation Function: Often Softmax for multi-class classification problems.

6. Output Module:

Output Layer: Produces the final classification probabilities.

Loss Function: Measures the difference between the predicted labels and the true labels (e.g., cross-entropy loss).

Optimizer: Adjusts the weights of the network to minimize the loss function (e.g., Adam, SGD).

7. Training Module:

Backpropagation: Computes the gradients and updates the weights of the network.

Epochs and Batching: Iterating over the dataset for multiple epochs and using minibatches to update weights.

8. Evaluation Module:

Metrics Calculation: Computes performance metrics such as accuracy, precision, recall, F1 score.

Validation and Testing: Evaluates the model on a separate validation and test dataset to ensure generalization.

9. Model Saving and Loading Module:

Checkpointing: Saving the model's state periodically during training.

Serialization: Storing the final trained model for future inference.

10. Inference Module:

Model Deployment: Using the trained model to classify new, unseen images in a production environment.

3.3 TECHNIQUE USED OR ALGORITHM USED

Here's a step-by-step algorithm for image classification using a Convolutional Neural Network (CNN):

Step 1: Data Preparation

1. **Data Collection:** Gather a dataset of labeled images.

2. **Data Splitting:** Split the dataset into training, validation, and testing sets.

3. **Data Augmentation (optional):** Apply transformations like rotation, scaling, and flipping to artificially expand the training dataset.

4. **Data Normalization:** Scale pixel values to a range of [0, 1] or [-1, 1].

Step 2: Define the CNN Architecture

1. Input Layer: Specify the shape of the input image (e.g., 64x64x3 for a 64x64 RGB image).

2. Convolutional Layer:

Apply multiple filters to the input image.

Use activation functions (typically ReLU) after each convolution.

3.Pooling Layer: Apply max pooling or average pooling to reduce the spatial dimensions.

4.Flattening Layer: Convert the 2D matrix data to a 1D vector.

5.Fully Connected Layer: Add one or more dense layers.

6.Output Layer: Use a softmax activation function for multi-class classification or sigmoid for binary classification.

Step 3: Compile the Model

1. **Loss Function:** Choose an appropriate loss function (e.g., categorical cross-entropy for multi-class, binary cross-entropy for binary).

2. **Optimizer:** Choose an optimizer (e.g., Adam, SGD).

3. **Evaluation Metrics**: Specify metrics to evaluate the model (e.g., accuracy).

Step 4: Train the Model

1. **Feed Training Data:** Use the training dataset to train the model.

2. **Batch Size:** Define the batch size for training.

3. **Epochs:** Set the number of epochs.

4. **Validation**: Monitor performance on the validation set.

Step 5: Evaluate the Model

1. **Testing**: Use the testing dataset to evaluate the final model performance.

2. **Metrics Calculation:** Compute metrics such as accuracy, precision, recall, and F1-score.

Step 6: Save the Model

1. **Checkpointing:** Save the model periodically during training.

2. **Final Model:** Save the final trained model for future inference.

Step 7: Inference

1. **Load Model:** Load the trained model.

2. **Predict:** Use the model to classify new images.

This algorithm covers the entire process of image classification using a CNN, from data preparation to model deployment. Adjust the architecture and parameters based on the specific problem and dataset.

CHAPTER 4

REQUIREMENT

4.1 GENERAL

These are the requirements for doing the project. Without using these tools & software's we cannot do the project. Therefore, we have two requirements to do the project. They are

- Hardware Requirements.
- Software Requirements.

4.2 Hardware Requirements

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. Software engineers use them as the starting point for the system design. It should what the system and not how it should be implemented.

- PROCESSOR : i5 Processor
- RAM : 8GB RAM
- HARD DISK : 512 SSD

4.3 Software Requirements

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks, tracking the teams, and tracking the team's progress throughout the development activity.

- Code platform : Jupyter,VScode
- Operating System : Windows 11
- Language : Python 3.12.2

4.4 FUNCTIONAL REQUIREMENTS

Functional requirements for an image classification system using Convolutional Neural Networks (CNNs) detail the specific functionalities that the system must have to operate correctly. These requirements can be categorized into various stages of the system, from data preparation to deployment. Here are the key functional requirements:

Data Preparation

1. Data Loading:

The system must be able to load images from a specified directory or dataset.

The system should support multiple image formats (e.g., JPEG, PNG).

2. Data Augmentation:

The system must be able to apply data augmentation techniques such as rotation, scaling, flipping, and cropping.

3. Data Normalization:

The system must normalize image pixel values to a specified range (e.g., [0, 1]).

4. Data Splitting:

The system must be able to split the dataset into training, validation, and testing sets.

Model Architecture

5. Define Input Layer:

The system must allow specifying the input shape of images.

6. Add Convolutional Layers:

The system must support adding convolutional layers with configurable parameters such as the number of filters, kernel size, and activation function.

7. *Add Pooling Layers*:

The system must support adding pooling layers (e.g., max pooling, average pooling) with configurable parameters.

8. Add Flattening Layer:

The system must support adding a flattening layer to convert 2D matrices to 1D vectors.

9. Add Fully Connected Layers:

The system must support adding dense layers with configurable parameters such as the number of units and activation function.

10. Define Output Layer:

The system must allow specifying the output layer with appropriate activation functions (e.g., softmax for multi-class, sigmoid for binary classification).

Model Compilation

11. Specify Loss Function:

The system must allow specifying the loss function (e.g., categorical cross-entropy, binary cross-entropy).

12. Specify Optimizer:

The system must allow specifying the optimizer (e.g., Adam, SGD) with configurable parameters.

13. Specify Evaluation Metrics:

The system must allow specifying evaluation metrics (e.g., accuracy, precision, recall).

Model Training

14. Train the Model:

The system must support training the model on the training dataset with specified batch size and number of epochs.

15. Monitor Training:

The system must provide real-time feedback on training progress, including metrics such as loss and accuracy.

16. Validation During Training:

17. The system must support validating the model on a validation dataset during training.

Model Evaluation

18. Evaluate the Model:

The system must support evaluating the trained model on a testing dataset.

19.Calculate Metrics:

The system must calculate and display evaluation metrics such as accuracy, precision, recall, and F1-score.

Model Saving and Loading

20 Save Model:

The system must support saving the trained model to a file.

21 Load Model:

The system must support loading a previously saved model from a file.

Model Inference

22 Predict:

The system must support making predictions on new, unseen images.

23 Batch Prediction:

The system must support making predictions on a batch of images and returning results.

User Interface (Optional)

24User Interface for Configuration:

The system should provide a user interface for configuring the model architecture, training parameters, and other settings.

25. User Interface for Monitoring:

The system should provide a user interface for monitoring training progress and displaying evaluation results. ### Logging and Error Handling

26 Logging:

The system must log significant events such as model training start/end, evaluation results, and errors.

27 Error Handling:

The system must handle errors gracefully, providing informative error messages and suggestions for resolution.

Documentation and Help

28Documentation:

The system must provide documentation detailing how to use the system, including examples and best practices.

29Help and Support:

The system should provide help features, such as tooltips or a help menu, to assist users in navigating the system.

These functional requirements ensure that the image classification system using CNNs is robust, user-friendly, and capable of handling the entire workflow from data preparation to model deployment and inference.

4.5 NON-FUNCTIONAL REQUIREMENTS

Non-functional requirements (NFRs) specify criteria that judge the operation of a system, rather than specific behaviors. For an image classification system using Convolutional Neural Networks (CNNs), NFRs ensure the system's performance, usability, reliability, and other quality attributes. Here are some key non-functional requirements:

Performance

1. **Training Time:**

The system should minimize the time required to train the model, possibly leveraging hardware accelerators like GPUs or TPUs.

2. **Inference Time:**

The system should provide fast inference times, ensuring predictions are made within an acceptable time frame.

3. **Scalability:**

The system should be able to handle increasing amounts of data and larger model architectures without significant performance degradation.

4. **Throughput:**

The system should be capable of processing a large number of images per second during both training and inference phases.

Usability

5. **Ease of Use:**

The system should provide an intuitive interface for users to configure, train, and evaluate the model.

6. **Documentation:**

The system should have comprehensive documentation, including setup instructions, usage examples, and troubleshooting guides.

7. **Accessibility:**

The system should be accessible to users with varying levels of expertise, from beginners to advanced users.

Reliability

8. **Fault Tolerance:**

The system should handle hardware or software failures gracefully, possibly with mechanisms to resume training from the last checkpoint.

9. **Robustness:**

The system should handle a variety of input data gracefully, including unexpected or corrupted data, without crashing.

10. **Consistency:**

The system should produce consistent results when trained and evaluated multiple times on the same dataset.

Maintainability

11. **Modularity:**

The system's codebase should be modular, allowing for easy updates and maintenance of individual components.

12. **Code Quality:**

The system should adhere to good coding practices, with clear, readable, and welldocumented code.

13.Testing:

The system should include unit tests, integration tests, and end-to-end tests to ensure its correctness.

Security

14.Data Privacy:

The system should ensure that sensitive data, such as user-provided images, is handled securely and complies with relevant data protection regulations.

15. Access Control:

The system should have mechanisms to restrict access to authorized users only.

Portability

16. Platform Independence:

The system should be able to run on various operating systems (Windows, macOS, Linux) with minimal configuration.

17.Deployment Flexibility:

The system should support deployment in different environments, such as local machines, cloud platforms, or edge devices.

Efficiency

18.Resource Utilization:

The system should make efficient use of computational resources, such as CPU, GPU, memory, and disk space.

19.Energy Efficiency:

The system should minimize energy consumption, especially important when deploying on battery-powered devices.

Interoperability

20.Compatibility:

The system should be compatible with other machine learning frameworks and tools, allowing for easy integration and data exchange.

21.Integration:

The system should support integration with other systems, such as data pipelines, logging services, and monitoring tools.

Extensibility

22.Customizability:

The system should allow users to easily extend its functionality, such as adding custom data preprocessing steps, new model architectures, or evaluation metrics.

23.Plugin Support:

The system should support plugins or extensions to add new features without modifying the core system.

Monitoring and Logging

24.Monitoring:

The system should provide real-time monitoring of training and inference processes, including metrics like loss, accuracy, and resource utilization.

25.Logging:

The system should maintain detailed logs of all operations, including data loading, training, evaluation, and errors, for auditing and debugging purposes.

26.Regulatory Compliance:

The system should comply with relevant industry standards and regulations, such as GDPR for data protection.

27.Ethical Considerations:

The system should ensure that it is used ethically, with considerations for fairness, transparency, and accountability in its operations and results.

These non-functional requirements ensure that the image classification system is not only functional but also performant, user-friendly, secure, and maintainable.

CHAPTER 5

IMPLEMENTATION

5.1 GENERAL

The Implementation is nothing but source code of project

5.2 IMPLEMENTATION CODING:

```
# Dataset - https://www.kaggle.com/datasets/salader/dogs-vs-cats

!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/

!kaggle datasets download -d salader/dogs-vs-cats

import zipfile
zip_ref = zipfile.ZipFile('/content/dogs-vs-cats.zip', 'r')
zip_ref.extractall('/content')
zip_ref.close()

import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,Flatten,BatchNormalization,Dropout

# generators
train_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/train',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)

validation_ds = keras.utils.image_dataset_from_directory(
    directory = '/content/test',
    labels='inferred',
    label_mode = 'int',
    batch_size=32,
    image_size=(256,256)
)

# Normalize
def process(image,label):
    image = tf.cast(image/255. ,tf.float32)
```

```
    return image,label

train_ds = train_ds.map(process)
validation_ds = validation_ds.map(process)

# create CNN model

model = Sequential()

model.add(Conv2D(32,kernel_size=(3,3),padding='valid',activation='relu',input_shape=(2
56,256,3)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(64,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Conv2D(128,kernel_size=(3,3),padding='valid',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2),strides=2,padding='valid'))

model.add(Flatten())

model.add(Dense(128,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(1,activation='sigmoid'))

model.summary()

model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

history = model.fit(train_ds,epochs=10,validation_data=validation_ds)

import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'],color='red',label='train')
plt.plot(history.history['val_accuracy'],color='blue',label='validation')
plt.legend()
plt.show()
```

```
import cv2

test_img = cv2.imread('/content/cat.jpg')

plt.imshow(test_img)

test_img.shape

test_img = cv2.resize(test_img,(256,256))

test_input = test_img.reshape((1,256,256,3))

model.predict(test_input)
```

CHAPTER 6

RESULTS & DISCUSSION

6.1 GENERAL

The entire project is done on jupyter using python

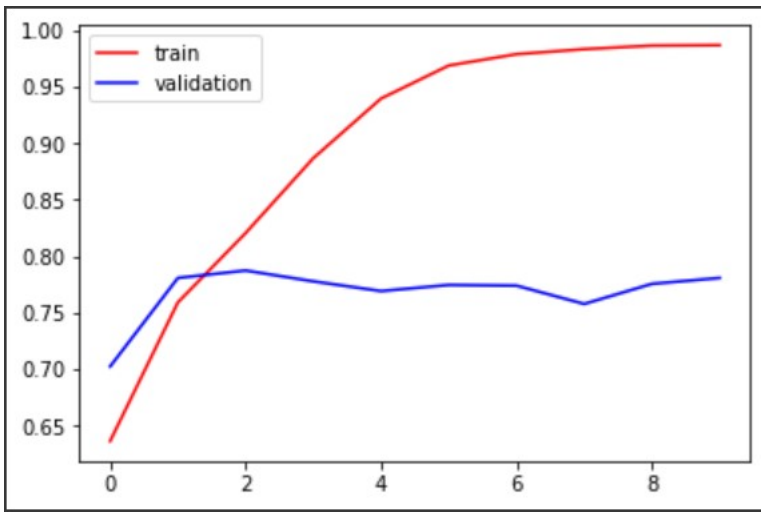
6.2 RESULTS & DISCUSSION

```
# Dataset - https://www.kaggle.com/datasets/salader/dogs-vs-cats
```

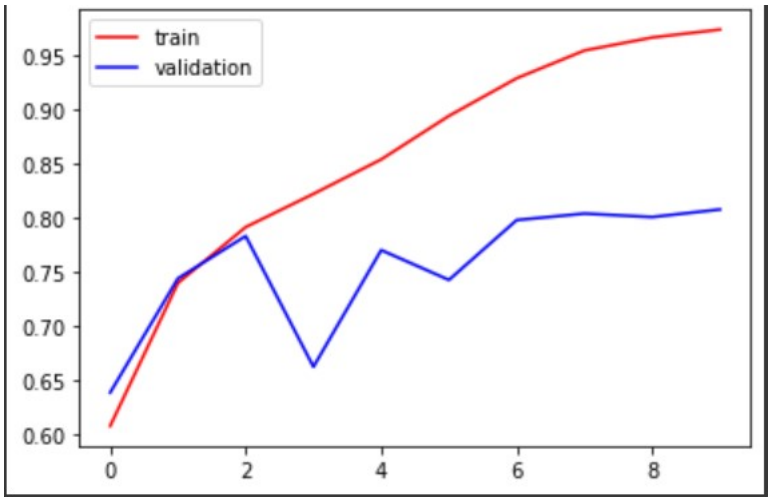
The dataset for this project is available on this link given above.

Now here are the results of the model after the training and testing of the CNN model

Graph1:



Graph 2:



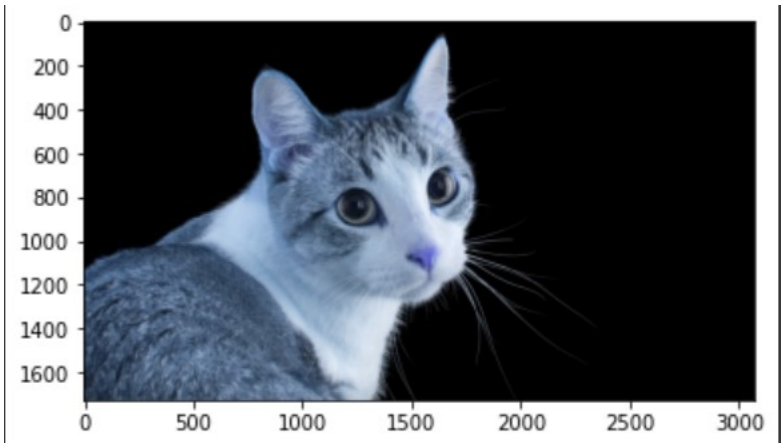
The gap between training and validation lines indicates the accuracy of the model., Upon analyzing these graphs the model is to be showing 86% accuracy in identifying the Data.

Model Summary

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)            (None, 254, 254, 32)      896

 batch_normalization (BatchN  (None, 254, 254, 32)     128
 ormalization)

 max_pooling2d_3 (MaxPooling  (None, 127, 127, 32)     0
 2D)

 conv2d_5 (Conv2D)            (None, 125, 125, 64)      18496

 batch_normalization_1 (Batc  (None, 125, 125, 64)     256
 hNormalization)

 max_pooling2d_4 (MaxPooling  (None, 62, 62, 64)       0
 2D)

 conv2d_6 (Conv2D)            (None, 60, 60, 128)       73856

 batch_normalization_2 (Batc  (None, 60, 60, 128)      512
 hNormalization)

 max_pooling2d_5 (MaxPooling  (None, 30, 30, 128)      0
 2D)

 flatten_1 (Flatten)          (None, 115200)            0

 dense_3 (Dense)              (None, 128)               14745728

 dropout (Dropout)            (None, 128)               0

 dense_4 (Dense)              (None, 64)                8256

 dropout_1 (Dropout)          (None, 64)                0

 dense_5 (Dense)              (None, 1)                 65

=================================================================
Total params: 14,848,193
Trainable params: 14,847,745
Non-trainable params: 448
_________________________________________________________________
```

Output of the tested Image

```
<matplotlib.image.AxesImage at 0x7f133882b390>
```




```
test_img.shape
```

```
(1728, 3072, 3)
```

```
test_img = cv2.resize(test_img,(256,256))

test_input = test_img.reshape((1,256,256,3))

model.predict(test_input)
```

```
array([[0.]], dtype=float32)
```

The model resizes the images and Normalizes the image pixels from (0,256) to (0,1) each using cv2 and the other convolutional layers the CNN model uses the downloaded database on which the model was trained to identify the given image

If the output is = 0 , then the result is, it is a CAT.

If the output is = 1 , then the result is, it is a DOG.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

Image classification using Convolutional Neural Networks (CNNs) has revolutionized the field of computer vision, providing a robust and efficient method for analyzing visual data. The architecture of CNNs, inspired by the visual cortex of animals, excels in identifying patterns and features within images, making them the preferred choice for a wide array of image recognition tasks. By leveraging layers such as convolutional, pooling, and fully connected layers, CNNs can automatically and adaptively learn spatial hierarchies of features from input images. This capability allows CNNs to outperform traditional image processing techniques, which often require manual feature extraction and are limited in their ability to handle the complexity of real-world data.

The process of building an image classification system with CNNs involves several critical stages, starting with data preparation. This includes collecting and preprocessing the data, which often involves data augmentation techniques to enhance the robustness of the model. Following this, the CNN architecture is defined and built, typically involving multiple convolutional and pooling layers to capture intricate features, followed by dense layers that interpret these features to perform classification. The model is then compiled with appropriate loss functions and optimizers, and trained using the training dataset. During training, the model's performance is monitored on a validation set to ensure it generalizes well to unseen data, preventing overfitting.

One of the key advantages of CNNs in image classification is their ability to handle large volumes of data efficiently. Through the use of GPUs and parallel processing, training times have been significantly reduced, making it feasible to develop highly accurate models within a reasonable timeframe. The scalability of CNNs also means they can be adapted for various applications, from medical imaging to autonomous driving, where precise and reliable image classification is crucial.

However, developing a CNN-based image classification system also presents challenges. Ensuring high performance requires a careful balance between model complexity and computational efficiency. Moreover, the requirement for large labeled datasets can be a bottleneck, as acquiring and annotating such data is often resource-intensive. Despite these challenges, the advancements in transfer learning and pre-trained models have mitigated some of these issues, allowing developers to fine-tune existing models for specific tasks with relatively smaller datasets.

In conclusion, CNNs have proven to be a powerful tool for image classification, enabling significant advancements in the field of computer vision. Their ability to automatically extract relevant features from raw image data, combined with their scalability and efficiency, has set a new standard for image recognition tasks. As technology continues to evolve, further improvements in CNN architectures and training methodologies promise to enhance their performance and broaden their applicability, solidifying their role as a cornerstone of modern artificial intelligence.

REFERENCES

In 2008, Golle published a paper titled "Machine learning attacks against the Asirra CAPTCHA" in which he demonstrated the vulnerability of the Asirrasystem to automated attacks using machine learning techniques.

Elson, Douceur, Howell, and Saul (2007) introduced Asirra in their paper titled "Asirra: a CAPTCHA that exploits interest-aligned manual imagecategorization". The paper was presented at the Proc. of ACM CCS 2007.

Ramprasath, Anand, and Hariharan (2018) proposed the use of Convolutional Neural Networks for image classification in their research paper titled"Image Classification using Convolutional Neural Networks", published in the International Journal of Pure and Applied Mathematics.

In their research paper titled "Cats and dogs", Parkhi, Vedaldi, Zisserman, and Jawahar (2012) presented a large-scale dataset of cats and dogs' imagesfor the purpose of image classification. The paper was presented at the Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference.

Zeiler and Fergus (2013) introduced a method for visualizing and understanding Convolutional Neural Networks in their paper titled "Visualizing andUnderstanding Convolutional Neural Networks", which was published on arXiv.

Liu, Liu, and Zhou (2016) proposed an image classification approach for dogs and cats in their paper titled "Image Classification for Dogs and Cats".