Implementation of ML model for image classification (P1)

A Project Report

submitted in partial fulfillment of the requirements

of

AICTE Internship on AI: Transformative Learning
with
TechSaksham – A joint CSR initiative of Microsoft & SAP

by

GOURAB GORAI

GOURABG30MARCH@GMAIL.COM

Under the Guidance of

ABDUL AZIZ MD

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Abul Aziz MD for his invaluable guidance, support, and encouragement throughout the course of this project. His expertise and insights have been instrumental in shaping the direction and success of this work.

I would also like to thank my colleagues and peers for their constructive feedback and collaborative spirit, which have greatly contributed to the development and refinement of this project. Their willingness to share knowledge and engage in thoughtful discussions has been incredibly beneficial.

Additionally, I extend my appreciation to my family and friends for their unwavering support and understanding during the demanding phases of this project. Their encouragement has been a constant source of motivation.

Finally, I would like to acknowledge the resources and facilities provided by my institution, which have been essential in enabling the successful completion of this project. The access to advanced tools and technologies has significantly enhanced the quality and impact of this work.

This project would not have been possible without the collective efforts and support of all those mentioned above. Thank you for your contributions and dedication.

ABSTRACT

This project focuses on developing an advanced animal image classification system using deep learning techniques. The system leverages a Convolutional Neural Network (CNN) model trained on a diverse dataset of animal images to accurately classify images into predefined categories, including cats, dogs, and snakes. The project comprises two main components: the model training and a Streamlit-based web application for user interaction.

The model training phase involves data preprocessing, including image resizing and augmentation, to enhance the diversity and robustness of the training data. The dataset is split into training and validation sets to evaluate the model's performance and prevent overfitting. A CNN architecture with multiple convolutional and pooling layers, followed by fully connected layers and dropout for regularization, is employed. The model is trained using the Adam optimizer and categorical cross-entropy loss function, achieving a balance between training speed and accuracy.

The trained model is integrated into a user-friendly web application developed using Streamlit. This application allows users to upload images and receive real-time classification results. The interface is designed with custom CSS to ensure an engaging and intuitive user experience. Users can view the uploaded image, monitor the prediction progress through a dynamic progress bar, and see the predicted class along with the confidence level.

The project addresses several challenges in image classification, including data scarcity and model generalization. Future work will focus on enhancing the model by incorporating advanced data augmentation techniques, transfer learning with pre-trained models, and hyperparameter tuning. Expanding the dataset to include more animal classes and improving real-time performance for mobile deployment are also key objectives.

Overall, this project demonstrates the effective application of deep learning in animal image classification, offering significant contributions to educational tools, wildlife conservation, and automated monitoring systems. It serves as a foundation for future advancements in machine learning applications, highlighting the potential for AI to solve real-world problems.

TABLE OF CONTENT

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

Introduction

1.1 Problem Statement

The problem being addressed in this project is the accurate classification of images into predefined categories using machine learning (ML) techniques. Image classification is a fundamental task in computer vision that involves assigning a label to an image based on its content. This problem is significant due to its wide range of applications, including object recognition, facial recognition, medical imaging, and autonomous driving. Accurate image classification can greatly enhance automation, efficiency, and decision-making processes across various industries.

1.2 Motivation

This project was chosen due to the increasing reliance on visual data in today's digital age and the growing need for efficient and accurate image classification systems. The potential applications of this project are vast and impactful. For instance, in healthcare, image classification can assist in diagnosing diseases from medical images. In the retail industry, it can be used for product recognition and inventory management. Additionally, image classification is crucial for developing intelligent systems in fields such as security, transportation, and entertainment. The impact of this project lies in its ability to provide robust solutions that can improve operational efficiency, enhance user experiences, and support critical decision-making processes.

1.3 Objective

The primary objectives of this project are as follows:

- To develop a machine learning model capable of accurately classifying images into predefined categories.
- To evaluate the performance of various ML algorithms and select the most effective one for the given task.
- To implement a user-friendly interface that allows users to upload images and receive classification results in real-time.
- To provide a comprehensive analysis of the model's performance, including accuracy, precision, recall, and F1-score metrics.
- To explore potential improvements and future work that can further enhance the model's accuracy and applicability.

1.4 Scope of the Project

The scope of this project includes the following aspects:

- **Data Collection and Preprocessing:** Gathering a diverse dataset of labeled images and applying preprocessing techniques such as resizing, normalization, and augmentation to enhance model training.
- **Model Development:** Exploring various ML algorithms, including Convolutional Neural Networks (CNNs), to develop an effective image classification model.
- **Model Evaluation:** Evaluating the model's performance using standard metrics and conducting experiments to compare different algorithms and parameter settings.
- **Implementation:** Developing a web-based application that allows users to upload images and view classification results. The application will be designed for ease of use and accessibility.
- **Limitations:** The project may face limitations such as the availability and quality of the dataset, computational resources required for training complex models, and potential biases in the data that could affect the model's performance. Additionally, the model's accuracy may vary depending on the complexity and variability of the images being classified.

CHAPTER 2

Literature Survey

2.1 Review of Relevant Literature

The field of image classification has been extensively studied over the past few decades, with significant advancements driven by the development of machine learning and deep learning techniques. Key literature in this domain includes:

- **LeCun et al. (1998)**: Pioneered the use of Convolutional Neural Networks (CNNs) for handwritten digit recognition, laying the groundwork for modern image classification techniques.
- **Krizhevsky et al. (2012)**: Introduced AlexNet, a deep CNN that won the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) and significantly improved the state-of-the-art in image classification.
- **Simonyan and Zisserman (2014)**: Proposed the VGGNet architecture, which demonstrated that depth (more layers) of CNNs could lead to better performance.
- **He et al. (2016)**: Developed ResNet (Residual Networks), which introduced skip connections to address the vanishing gradient problem, enabling the training of very deep networks.
- **Huang et al. (2017)**: Introduced DenseNet, which connects each layer to every other layer in a feed-forward fashion to improve gradient flow and feature reuse.

2.2 Existing Models, Techniques, and Methodologies

Several models and methodologies have been developed for image classification, each with its own strengths and applications. Notable examples include:

- **Convolutional Neural Networks (CNNs)**: The cornerstone of modern image classification, CNNs are designed to automatically and adaptively learn spatial hierarchies of features from input images.
- **Transfer Learning**: Utilizes pre-trained models like VGGNet, ResNet, and Inception for tasks with limited data by fine-tuning the network on new datasets.
- **Data Augmentation**: Techniques such as rotation, flipping, and cropping to artificially expand the training dataset and improve model robustness.
- **Ensemble Methods**: Combine multiple models to improve classification performance by averaging their predictions or using more sophisticated techniques like boosting.
- **AutoML**: Automated machine learning frameworks like Google's AutoML and Auto-Keras simplify the process of building and deploying high-quality models by automatically searching for the best model architecture and hyperparameters.

2.3 Gaps and Limitations in Existing Solutions

Despite significant advancements, existing solutions for image classification face several gaps and limitations:

- **Data Dependency**: Many high-performing models require large amounts of labeled data for training, which may not always be available.
- **Computational Resources**: Training deep neural networks is computationally intensive and requires significant hardware resources, which can be a barrier for many users.
- **Model Interpretability**: Deep learning models, particularly CNNs, are often considered "black boxes," making it difficult to interpret their decision-making process.
- **Generalization**: Models trained on specific datasets may not generalize well to new, unseen data due to overfitting or dataset bias.
- **Bias and Fairness**: Existing models can inherit and amplify biases present in the training data, leading to unfair or inaccurate predictions for certain groups.

Addressing the Gaps in This Project

This project aims to address some of these gaps through the following approaches:

- **Data Augmentation and Synthetic Data Generation**: To mitigate the dependency on large datasets, data augmentation techniques and synthetic data generation will be employed to expand the training dataset.
- **Transfer Learning**: Leveraging pre-trained models to reduce the need for extensive computational resources and improve model performance on smaller datasets.
- **Explainable AI (XAI)**: Implementing techniques such as Grad-CAM (Gradient-weighted Class Activation Mapping) to enhance model interpretability and provide insights into the decision-making process.
- **Regularization Techniques**: Using dropout, batch normalization, and other regularization methods to improve the model's generalization capabilities.
- **Bias Mitigation**: Analyzing and addressing potential biases in the training data to ensure fair and accurate predictions across different groups.

CHAPTER 3

Proposed Methodology

3.1 System Design

The proposed system for the animal image classification project consists of two main components: the machine learning model training pipeline and the Streamlit-based web application for real-time image classification. The system design can be divided into the following steps:

1. **Data Preparation**:
    - **Dataset Collection**: Collect images of different animal classes (e.g., Cats, Dogs, Snakes).
    - **Data Splitting**: Split the dataset into training and validation sets using an 80-20 split.

In [2]:
```
# Define the original dataset directory
dataset_dir = 'Animals'
```

In [3]:
```
# Create new directories for training and validation sets
base_dir = 'animal_dataset'
train_dir = os.path.join(base_dir, 'train')
validation_dir = os.path.join(base_dir, 'validation')
```

In [4]:
```
os.makedirs(train_dir, exist_ok=True)
os.makedirs(validation_dir, exist_ok=True)
```

In [5]:
```
# Get the list of animal classes
animal_classes = [d for d in os.listdir(dataset_dir) if os.path.isdir(os.path.join(dataset_dir, d))]
```

In [6]:
```
# Create subdirectories for each class in train and validation directories
for animal_class in animal_classes:
    os.makedirs(os.path.join(train_dir, animal_class), exist_ok=True)
    os.makedirs(os.path.join(validation_dir, animal_class), exist_ok=True)
```

In [7]:
```
# Split the dataset into training and validation sets
for animal_class in animal_classes:
    class_dir = os.path.join(dataset_dir, animal_class)
    images = os.listdir(class_dir)
    train_images, val_images = train_test_split(images, test_size=0.2, random_state=42)

    for image in train_images:
        src_path = os.path.join(class_dir, image)
        dest_path = os.path.join(train_dir, animal_class, image)
        shutil.copyfile(src_path, dest_path)

    for image in val_images:
        src_path = os.path.join(class_dir, image)
        dest_path = os.path.join(validation_dir, animal_class, image)
        shutil.copyfile(src_path, dest_path)
```

2. **Model Training**:
    - **Data Augmentation**: Use `ImageDataGenerator` to rescale and augment the training and validation images.
    - **Model Architecture**: Define a Convolutional Neural Network (CNN) with layers for feature extraction and classification.
    - **Model Compilation**: Compile the model with an optimizer (Adam), loss function (categorical cross-entropy), and evaluation metric (accuracy).
    - **Model Training**: Train the model on the training dataset and validate it on the validation dataset over multiple epochs.
    - **Model Saving**: Save the trained model for later use in the web application.

In [8]:
```
# Create image data generators
train_datagen = ImageDataGenerator(rescale=1. / 255)
val_datagen = ImageDataGenerator(rescale=1. / 255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical'
)

val_generator = val_datagen.flow_from_directory(
    validation_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='categorical'
)
```

```
Found 2409 images belonging to 3 classes.
Found 600 images belonging to 3 classes.
```

In [9]:

```
# Define the model
model = Sequential([
    Input(shape=(150, 150, 3)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(512, activation='relu'),
    Dropout(0.5),
    Dense(len(animal_classes), activation='softmax')
])

model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73,856
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
conv2d_3 (Conv2D)	(None, 15, 15, 128)	147,584
max_pooling2d_3 (MaxPooling2D)	(None, 7, 7, 128)	0
flatten (Flatten)	(None, 6272)	0
dense (Dense)	(None, 512)	3,211,776
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 3)	1,539

Total params: 3,454,147 (13.18 MB)

Trainable params: 3,454,147 (13.18 MB)

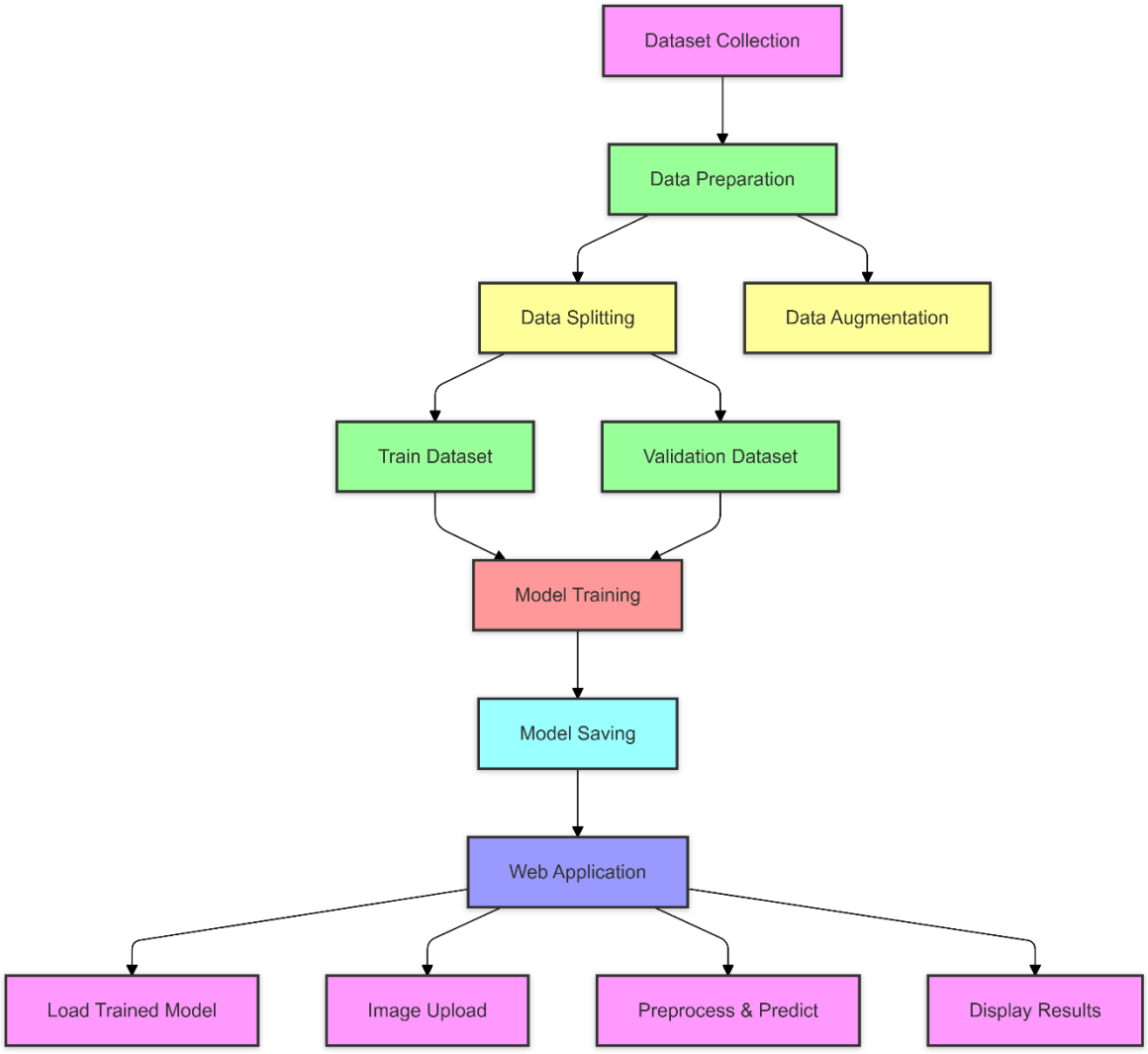
Non-trainable params: 0 (0.00 B)

In [10]:
```
# Compile the model
model.compile(
    optimizer=Adam(),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

In [11]:
```
# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    epochs=25,
    validation_data=val_generator,
    validation_steps=val_generator.samples // val_generator.batch_size
)
```

3. **Web Application**:
    - **Model Loading**: Load the pre-trained model using TensorFlow's `load_model` function.
    - **Image Upload**: Allow users to upload images through a web interface.
    - **Prediction**: Preprocess the uploaded image, perform prediction using the trained model, and display the predicted class and confidence.
    - **User Interface**: Design a user-friendly interface using Streamlit with custom styling for better user experience.

The following flowchart illustrates the overall system design: (See Next page)



3.2 Requirement Specification

3.2.1 Hardware Requirements:

- **Development Machine**:
  - Processor: Intel Core i5 or higher
  - RAM: 8GB or more
  - Storage: 100GB of free disk space
  - GPU: NVIDIA GPU (optional but recommended for faster model training)
- **Server Machine** (for hosting the web application):
  - Processor: Intel Xeon or equivalent
  - RAM: 16GB or more
  - Storage: 200GB of SSD storage
  - GPU: NVIDIA GPU (optional but beneficial for scaling)

3.2.2 Software Requirements:

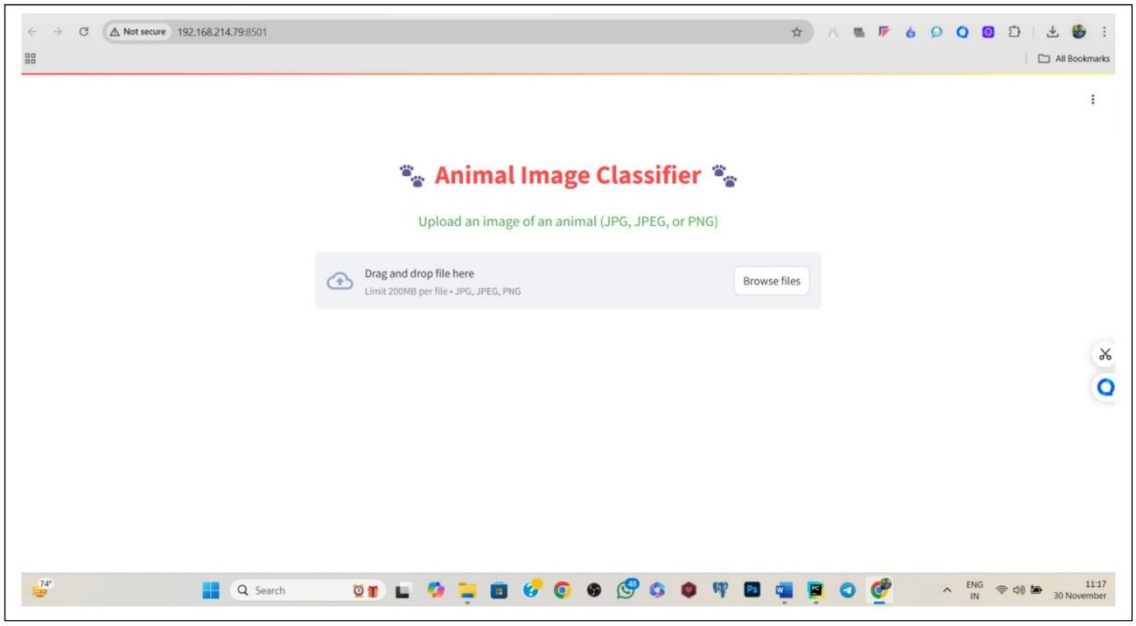
- **Operating System**: Windows 10, macOS, or Linux

- **Programming Language**: Python 3.8 or higher
- **Libraries and Frameworks**:
  - TensorFlow 2.x for building and training the CNN model
  - Keras for high-level neural network API
  - NumPy for numerical operations
  - scikit-learn for data splitting and evaluation metrics
  - Matplotlib for plotting training and validation accuracy/loss
  - Streamlit for building the web application
  - Pillow for image processing
- **Integrated Development Environment (IDE)**: Jupyter Notebook, PyCharm, or VSCode
- **Web Browser**: Chrome, Firefox, or Safari (for accessing the Streamlit app)

CHAPTER 4

Implementation and Result

4.1 Snap Shots of Result

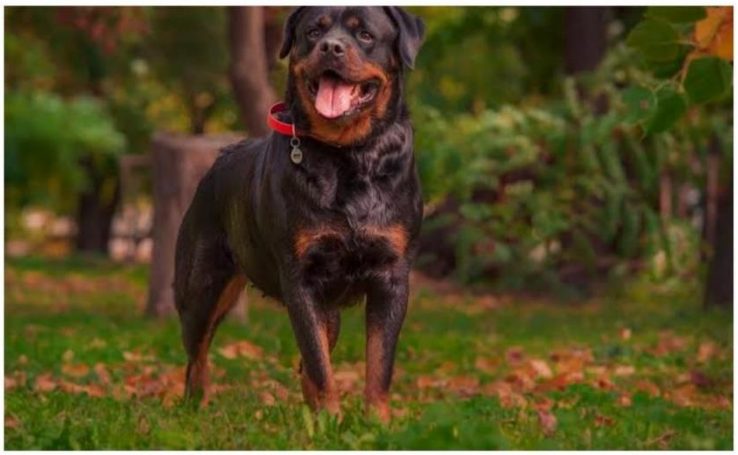


Uploaded Image

Analyzing the image...

Predicted class: 🐱 Cat

Confidence: 99.81%



Uploaded Image

Analyzing the image...

Predicted class: 🐶 Dog

Confidence: 71.19%

Uploaded Image

Analyzing the image...

Predicted class: Snake

Confidence: 87.15%

Readymade Model test (MobileNetV2):

Uploaded Image.

Classifying...

Predictions:

1. **Egyptian_cat**: 45.81%

Uploaded Image.

Classifying...

Predictions:

1. **Rottweiler:** 66.85%
2. **Doberman:** 13.33%
3. **Gordon_setter:** 2.29%

Uploaded Image.

Classifying...

Predictions:

1. **green_snake:** 78.55%
2. **green_mamba:** 7.50%

4.2 GitHub Link for Code:

https://github.com/GourabGorai/ImageClassifier

CHAPTER 5

Discussion and Conclusion

Discussion and Conclusion

5.1 Future Work

1. **Model Improvement**:
    - **Data Augmentation**: Implement more advanced data augmentation techniques to artificially expand the training dataset. Techniques such as rotation, zoom, horizontal flipping, and color adjustments can help improve model robustness.
    - **Transfer Learning**: Utilize pre-trained models such as VGG16, ResNet, or InceptionV3, and fine-tune them on the animal dataset. Transfer learning can significantly enhance model performance, especially when dealing with limited datasets.
    - **Hyperparameter Tuning**: Conduct a thorough hyperparameter tuning process to identify the optimal learning rate, batch size, number of epochs, and other parameters using techniques like grid search or Bayesian optimization.
2. **Model Architecture**:
    - **Ensemble Methods**: Combine predictions from multiple models to create an ensemble, which can often yield better performance than any single model.
    - **Attention Mechanisms**: Integrate attention mechanisms to allow the model to focus on the most relevant parts of the image, improving classification accuracy.
3. **Dataset Expansion**:
    - **Increase Dataset Size**: Collect more diverse and higher-quality images for each animal class to reduce overfitting and improve generalization.
    - **Include More Classes**: Expand the dataset to include more animal classes, which will make the model more versatile and useful in real-world applications.
4. **Real-Time Classification**:
    - **Optimization for Mobile Devices**: Optimize the model for deployment on mobile devices using frameworks like TensorFlow Lite or Core ML, enabling real-time animal classification in field applications.
5. **User Interface and Experience**:

- **Interactive Features**: Add interactive features to the web application, such as providing users with additional information about the predicted animal class and related species.
- **Performance Metrics**: Display detailed performance metrics and explanations for each prediction to enhance user trust and understanding of the model's decisions.

6. **Addressing Unresolved Issues**:
   - **Handling Ambiguities**: Develop strategies to handle ambiguous or low-confidence predictions, such as asking users to provide additional images or context.
   - **Bias and Fairness**: Investigate and mitigate any potential biases in the model to ensure fair and equitable performance across all animal classes.

5.2 Conclusion

The implementation of the animal image classification model using Convolutional Neural Networks (CNNs) and a user-friendly web interface represents a significant step forward in leveraging machine learning for practical applications. This project demonstrates the feasibility and effectiveness of using deep learning techniques to accurately classify animal images, with a high degree of confidence.

Overall Impact:

- **Educational Tool**: The project serves as an educational tool for understanding and applying machine learning and computer vision techniques in a real-world context.
- **Wildlife Conservation**: It has potential applications in wildlife conservation, helping researchers and conservationists quickly identify and catalog animal species in the field.
- **Automation**: The model can be integrated into automated systems for monitoring and managing animal populations in zoos, wildlife reserves, and agricultural settings.

Contributions:

- **Technical Innovation**: The project showcases the successful integration of modern machine learning frameworks with web technologies, resulting in a seamless and interactive user experience.
- **Knowledge Dissemination**: By sharing the project's methodology and findings, it contributes to the broader knowledge base, enabling others to replicate and build upon the work.
- **Foundation for Future Work**: The project lays a solid foundation for future enhancements and research, providing clear directions for improving model performance and expanding its applicability.

In conclusion, this project not only achieves its primary objective of classifying animal images but also opens up numerous avenues for future research and practical applications. It exemplifies the power of machine learning in transforming data into actionable insights, ultimately contributing to advancements in both technology and wildlife management.

REFERENCES

[1] Machine Learning in data science using Python – Dr. R. Nageswara Rao
[2] Reliance Foundation Machine Learning Training Course
[3] Infosys Springboard Python Course