

PROJECT REPORT ON

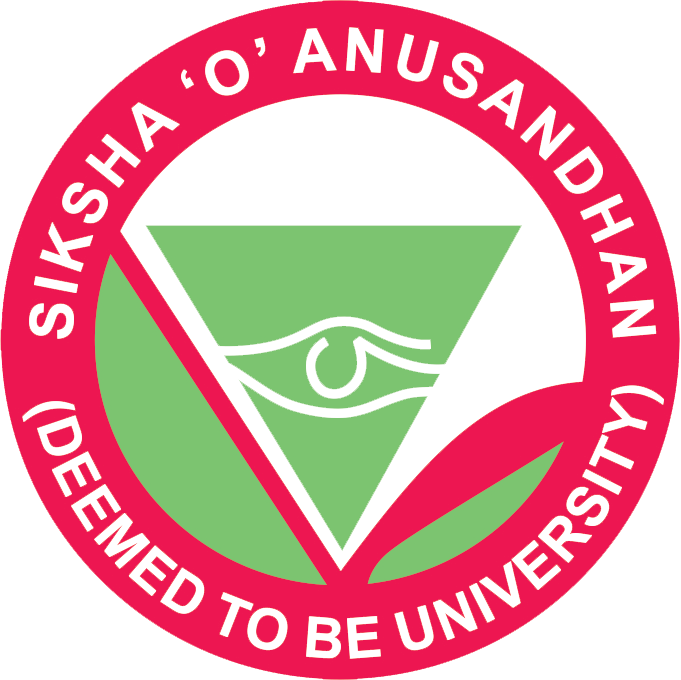
**DEEP LEARNING WORKSHOP WITH PYTHON (CSE3194)**

**Animal Species Detection**

**BACHELOR OF TECHNOLOGY**

Submitted by:

|  |  |
| --- | --- |
| **Sibun Nayak** | **REG. NO: 2241014132** |



Center for Artificial Intelligence & Machine Learning

**Department of Computer Science and Engineering**

Institute of Technical Education and Research

**Sikhsa ‘O’ Anusandhan**

**(Deemed to be University)**

Bhubaneswar

May, 2025

# Declaration

We hereby declare that the project report titled **"<Animal Species Detection>"** is our own work, carried out under the guidance of **< Mr. Jagseer Singh >**. We have not plagiarized any content and have duly cited all references.We affirm that all external sources and references have been properly credited, and the content presented here is solely intended for academic purposes as part of our undergraduate curriculum.

**Sibun Nayak**

**2241014132**

# Acknowledgement

We would like to express our heartfelt gratitude to our project guide, **[Mr. Jagseer Singh]**, for their constant support, valuable guidance, and encouragement throughout the course of this project. Their expertise and insights were instrumental in shaping the direction and success of our work.

We are also thankful to the **Department of Artificial Intelligence and Machine Learning** for providing us with the necessary resources and a conducive environment to carry out this project effectively.

We gratefully acknowledge the support from platforms such as **Kaggle** for providing the dataset, and **NPTEL** for offering insightful courses which strengthened our understanding of key concepts.

Finally, we sincerely thank our friends and peers for their continuous support and motivation during the project development.

# Table of Contents

Contents

[Declaration i](#_Toc197781573)

[Acknowledgement ii](#_Toc197781574)

[Table of Contents iii](#_Toc197781575)

[Abstract iv](#_Toc197781576)

[Chapter 1 – Introduction 1](#_Toc197781577)

[1.1. Background and motivation 1](#_Toc197781578)

[1.2. Problem statement 1](#_Toc197781579)

[1.3. Objectives 1](#_Toc197781580)

[1.4. Scope 1](#_Toc197781581)

[1.5. Organisation of the Report 1](#_Toc197781582)

[Chapter 2 – Literature Review 2](#_Toc197781583)

[2.1. Existing methods and models 2](#_Toc197781584)

[2.2. Comparison with your approach 2](#_Toc197781585)

[Chapter 3 – System Design and Methodology 3](#_Toc197781586)

[3.1. Dataset description 3](#_Toc197781587)

[3.2. Exploratory Data Analysis 3](#_Toc197781588)

[3.3. Preprocessing steps 3](#_Toc197781589)

[3.4. Model architecture 3](#_Toc197781590)

[3.5. Description of the Algorithms 3](#_Toc197781591)

[Chapter 4 – Implementation Details 4](#_Toc197781592)

[4.1. Programming languages, frameworks 4](#_Toc197781593)

[4.2. Code modules description 4](#_Toc197781594)

[4.3. System Working 4](#_Toc197781595)

[Chapter 5 – Results and Discussion 5](#_Toc197781596)

[5.1. Evaluation metrics 5](#_Toc197781597)

[5.2. Results 5](#_Toc197781598)

[5.3. Discussion 5](#_Toc197781599)

[Chapter 6 – Conclusion & Future Work 6](#_Toc197781600)

[6.1. Summary of outcomes 6](#_Toc197781601)

[6.2. Limitations 6](#_Toc197781602)

[6.3. Future improvements 6](#_Toc197781603)

[References 7](#_Toc197781604)

[Appendices 8](#_Toc197781605)

[Additional code snippets 8](#_Toc197781606)

[Screenshots or logs 8](#_Toc197781607)

# Abstract

This project focuses on the classification of animal species using Convolutional Neural Networks (CNNs), leveraging the Animals-10 dataset which includes diverse categories such as cane, cavallo, elefante, gallina, gatto, mucca, pecora, ragno, scoiattolo . The objective is to compare the performance of three prominent CNN architectures—ZFNet, VGG16, and an improved version of GoogLeNet (Inception V1) in accurately identifying animal species from images. The dataset is preprocessed using the ImageDataGenerator with VGG16-specific preprocessing, and models are trained with categorical cross-entropy loss. ZFNet and GoogLeNet are implemented from scratch with batch normalization and dropout layers to enhance learning and reduce overfitting, while VGG16 utilizes transfer learning with convolutional layers. All models are trained for 22 epochs and evaluated based on validation accuracy, loss, and inference time. The models are further tested on unseen animal images to assess real-world generalization. The results indicate performance differences across architectures, highlighting trade-offs between complexity, speed, and accuracy. The results help identify the most efficient architecture for animal image recognition tasks. This comparative study provides insights into selecting appropriate CNN architectures for wildlife monitoring, conservation, and intelligent visual systems in biodiversity-rich environments.

# Chapter 1 – Introduction

## 1.1. Background and motivation

The rapid advancements in computer vision and deep learning have enabled automated image classification systems capable of matching or surpassing human performance in specific tasks. Animal species classification using Convolutional Neural Networks (CNNs) holds significant potential in applications like wildlife conservation, ecological monitoring, and smart farming. With large annotated datasets such as Animals-10 available, it is now feasible to train powerful models to accurately distinguish among species. The motivation for this project stems from the need for efficient and reliable animal identification systems that can work in real-time and adapt to diverse visual features across species.

## 1.2. Problem statement

The project aims to develop a deep learning based system capable of classifying animals and predicting its species. The system will leverage Convolutional Neural Networks (CNNs), a powerful type of deep learning architecture designed for image processing tasks.

## 1.3. Objectives

* To implement and compare three CNN architectures: ZFNet, VGG16, and GoogLeNet.
* To train and evaluate these models using the Animals-10 dataset.
* To analyze model performance based on accuracy, loss, and inference time.
* To identify the most suitable architecture for animal species classification.

## 1.4. Scope

This project focuses on supervised image classification of ten animal species using pre-defined CNN architectures. It includes preprocessing, model training, validation, evaluation, and prediction on test images. The study is limited to static images from the Animals-10 dataset and does not involve video input or real-time detection.

## 

## 1.5. Organisation of the Report

## Chapter 1: Introduction, including background, motivation, and objectives.

## Chapter 2: Literature review of CNNs and related work in image classification.

## Chapter 3: Methodology, detailing the dataset, preprocessing, and model architectures.

## Chapter 4: Experimental setup, training, evaluation, and result comparison.

## Chapter 5: Conclusion, challenges faced, and future work.

# Chapter 2 – Literature Review

## 2.1. Existing methods and models

## Over the past decade, various machine learning and deep learning approaches have been proposed for animal image classification. Traditional methods used handcrafted features such as SIFT, HOG, or SURF combined with classifiers like SVM or K-NN. However, these approaches had limited performance in complex real-world images due to their inability to generalize across diverse visual patterns.

## With the emergence of deep learning, Convolutional Neural Networks (CNNs) have become the standard for image-based classification tasks. The architectures include:

## AlexNet (2012): One of the first deep CNNs, introduced ReLU activation and dropout to reduce overfitting.

## ZFNet (2013): An improved version of AlexNet with better visualization and tuned filter sizes.

## VGG16 (2014): Known for its depth and use of small (3×3) filters, VGG16 achieved high accuracy on ImageNet.

## GoogLeNet/Inception V1 (2014): Introduced inception modules that allow multi-scale feature extraction in parallel, reducing computational cost.

## These models have shown promising results in general object recognition, and many have been adapted for specific tasks such as animal detection, medical imaging, and scene classification.

## 2.2. Comparison with your approach

In this project, we implement and compare ZFNet, VGG16, and GoogLeNet on the Animals-10 dataset to assess their effectiveness in classifying ten distinct animal species. Unlike traditional approaches that rely on handcrafted features, our method uses end-to-end learning through CNNs, which extract relevant features automatically.

* **ZFNet** is implemented from scratch and fine-tuned with batch normalization and dropout layers for better convergence.
* **VGG16** is used with transfer learning by freezing its convolutional base and training a custom classifier on top.
* **GoogLeNet** is also custom-built with improved inception modules that balance performance and computational efficiency.

This comparative study provides insights into how different CNN architectures perform on the same dataset in terms of accuracy, loss, and inference time highlighting the trade-offs between model complexity and speed.

# Chapter 3 – System Design and Methodology

## 3.1. Dataset description

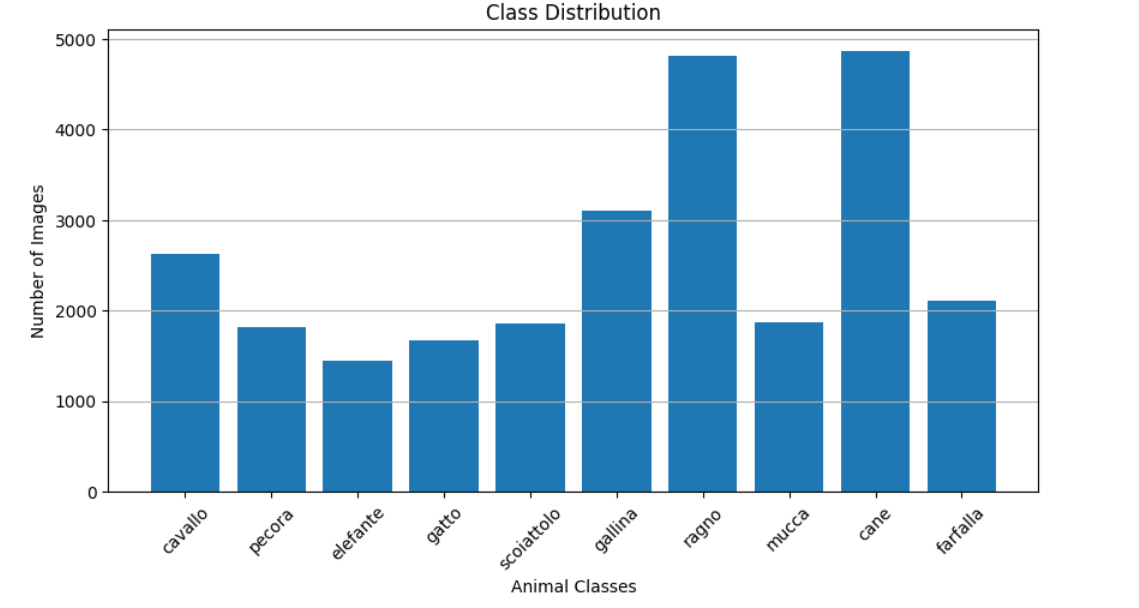
## This dataset consist of 28000 images approximately which are high variety and of 10 classes (subfolders) like cane, cavallo, elefante, gallina, gatto, mucca, pecora, ragno, scoiattolo These dataset are basically use for the classification problem of animal species. The images are collected from various sources and show high intra-class variation in lighting, background, poses, and image quality, making the classification problem challenging and realistic.

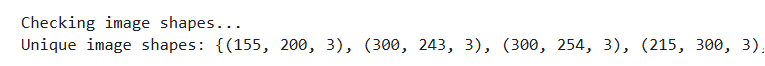
## The link of the dataset is given below:

## <https://www.kaggle.com/datasets/alessiocorrado99/animals10>

## 3.2. Exploratory Data Analysis

Before training the animal species classification models, we performed EDA to better understand the dataset and ensure data quality. The following insights and actions were derived:

* **Dataset Structure**: The dataset consists of 10 classes of animals, organized into folders where each folder contains images of a specific class.
* **Class Distribution**: We analysed the number of images per class to check for imbalance. The dataset used for training was imbalanced, with some classes significantly underrepresented compared to others. To address this issue and prevent the model from being biased toward the majority class, class weights were applied during training.
* 
* **Sample Visualization**: Random images from each class were plotted to visually confirm correct labelling and understand intra-class variation.
* 
* **Corrupt/Unreadable Images**: We scanned the dataset for corrupt or blank images. No such images were detected .
* 
* **Image Dimensions**: The original images varied widely in dimensions. A dimension frequency plot helped identify common sizes and outliers. This variation justifies resizing all images to a fixed input shape of **224×224×3**, which is required by pre-trained CNN architectures like VGG16, GoogleNet, and ZFNet.



* **Image Format**: All images are RGB, ensuring compatibility with CNN input requirements (3 colour channels)

## 3.3. Preprocessing steps

## The following preprocessing steps were applied:

## Resizing: All images were resized to (224 x 224) pixels to match the input requirement of CNN architectures.

## Normalization: Images were normalized using the preprocess\_input() function from tensorflow.keras.applications, which scales pixel values appropriately for networks.

## Train-Validation Split: The dataset was split into 80% training and 20% validation using ImageDataGenerator with the validation\_split parameter.

## 

## 3.4. Model architecture

## Three deep learning models were implemented and compared:

## 1. ZFNet

## Custom CNN inspired by ZFNet architecture.

## Consists of 5 convolutional layers, BatchNormalization, MaxPooling, and fully connected layers with Dropout.

## Modified for better stability and overfitting control.

## 2. VGG16

## Pre-trained on ImageNet and used with transfer learning.

## The convolutional base is frozen.

## A new classifier with a Dense layer (256 units) and softmax output was added.

## 3. GoogLeNet (Inception v1 Inspired)

## Custom-built with inception modules that process input at multiple scales.

## Includes 1x1, 3x3, and 5x5 convolutions along with pooling within each module.

## Ends with GlobalAveragePooling and a Dense layer.

## Each model was compiled using the Adam optimizer and trained using categorical cross-entropy loss function.

## 3.5. Description of the Algorithms

**🔹 ZFNet**

* An evolution of AlexNet with improved filter size and stride settings.
* Focuses on better visualization and feature learning.
* Useful for custom implementation and experimentation.

**🔹 VGG16**

* Known for its depth and simplicity—uses small (3x3) filters stacked together.
* Provides high performance but is computationally expensive.
* Pre-trained version used for fast convergence and transfer learning.

**🔹 GoogLeNet (Inception v1)**

* Utilizes parallel convolutions of multiple sizes (1x1, 3x3, 5x5) within the same module.
* Reduces parameters using 1x1 convolutions (bottlenecks).
* More efficient than VGG in terms of computation vs. accuracy trade-off.

**Pseudocode:**

For each model in [ZFNet, VGG16, GoogLeNet]:

Load dataset and preprocess images

Initialize model with pre-trained weights (if used)

Replace output layer with 10-class softmax

Compile model with categorical crossentropy loss

Train model with training data

Validate using validation data

Evaluate on test data

# Chapter 4 – Implementation Details

## 4.1. Programming languages, frameworks

**1.** **Programming Languages Used**

## **Python**: The primary language used for writing the entire deep learning pipeline, including data handling, preprocessing, model building, training, and evaluation.

**2.** **Frameworks and Libraries Used**

|  |  |
| --- | --- |
| **•TensorFlow /Keras:** | Used for building, training, and evaluating deep learning models. |

|  |  |
| --- | --- |
|  **KaggleHub** | : To download datasets directly from Kaggle. |

|  |  |
| --- | --- |
|  **OpenCV (cv2)** | For potential image processing tasks. |

|  |  |
| --- | --- |
|  **NumPy** | For numerical operations and array handling. |

|  |  |
| --- | --- |
|  **Matplotlib:** | For plotting accuracy/loss curves (visualizations). |

|  |  |
| --- | --- |
|  **Scikit-learn** | For evaluation metrics like classification reports. |

## 4.2. Code modules description

 **kagglehub**: Used to download datasets directly from Kaggle within the script.

 **TensorFlow and keras**: Used to create and train deep learning models.

 **VGG16**: Pretrained model imported from keras.applications for transfer learning.

 **ImageDataGenerator**: Helps in preprocessing, augmenting, and batching images.

 **Preprocess\_input**: Preprocessing function for VGG16 to scale input images correctly.

 **Matplotlib.pyplot**: For visualizing training/validation accuracy and loss.

 **time and os**: Handle time tracking and file paths.

### 1. **Download Dataset**

dataset\_path = kagglehub.dataset\_download("alessiocorrado99/animals10")

* Downloads the "Animals10" dataset from Kaggle and stores the path for further use.

### 🔹 2. **Prepare Dataset Directory**

original\_dir = os.path.join(dataset\_path, "raw-img")

* Points to the main image folder containing subfolders for each class (e.g., dog, cat, horse).

🔹 3. **Exploratory Data Analysis (EDA)**

Step 1: Visualize sample images from each class

sample\_paths = [...] # Select 2 random images per class

Displays a 2x5 grid of images with class names

* Randomly samples 1–2 images per animal class.
* Displays up to 10 images with class labels for quick inspection.

Step 2: Class distribution

class\_counts = {...} # Count number of images per class

* Plots a bar chart of animal species vs. number of images.
* Helps detect class imbalance issues.

Step 3: Corrupt/Blank image check

cv2.imread(path)

* Verifies image integrity by checking for unreadable or blank images.
* Removes any found corrupt images from the dataset.

Step 4: Image shape analysis

image\_shapes = [...] # Collects image dimensions

unique\_shapes = set(image\_shapes)

* Gathers dimensions (height, width, channels) of all images.
* Detects inconsistencies in image sizes.

### 🔹 4. **Image Preprocessing & Augmentation**

datagen = ImageDataGenerator(validation\_split=0.2, preprocessing\_function=preprocess\_input)

* Applies VGG-style preprocessing and splits data into training (80%) and validation (20%).

train\_generator = datagen.flow\_from\_directory(...)

val\_generator = datagen.flow\_from\_directory(...)

* Loads and batches images from directories into training and validation sets.
* Applies preprocessing and resizing to each image.

### 🔹 4. **Model Architectures**

#### ZFNet

def build\_zfnet\_improved():

...

* A custom deep CNN model inspired by ZFNet.
* Includes convolutional, batch normalization, max-pooling, and dense layers.
* Uses dropout to reduce overfitting.

#### VGG16 Transfer Learning Model

def build\_vgg16():

...

* Loads a pretrained VGG16 base model (without top classifier layers).
* Adds custom dense layers on top for classifying the Animals10 dataset.
* Freezes the base layers to retain pretrained features.

#### GoogLeNet

def inception\_module\_improved(...):

...

* Defines an improved Inception module that uses multiple filter sizes in parallel.

def build\_googlenet\_improved():

...

* Builds a simplified GoogLeNet architecture using custom Inception modules.

### 🔹 5. **Model Compilation & Training**

model.compile(optimizer='adam', loss='categorical\_crossentropy', metrics=['accuracy'])

**class\_weights**

Class weights are used to address class imbalance during model training. They assign different weights to each class, making the model pay more attention to underrepresented classes. By increasing the weight for less frequent classes, the model adjusts its learning to reduce bias toward overrepresented classes.

In practice, class weights scale the loss for each class during training. This means the model will prioritize minimizing errors for underrepresented classes more than for overrepresented ones, improving overall model performance on imbalanced datasets

model.fit(train\_generator, validation\_data=val\_generator, epochs=...)

* Compiles the selected model using Adam optimizer and categorical crossentropy loss.
* Trains the model on the training data and validates it on the validation data.
* Here the class weights are also considered.

### 🔹 6. **Model Evaluation (Optional, shown in extended code)**

from sklearn.metrics import classification\_report

* Used to generate detailed classification metrics like precision, recall, F1-score.

plt.plot(...), plt.show()

* Used to visualize training history (accuracy and loss curves).

## 4.3. System Working

 **Dataset Downloading**  
The dataset is automatically fetched from Kaggle using kagglehub.

 **Data Preprocessing**  
Images are resized to 224x224, split into training and validation sets, and normalized using ImageNet's preprocessing.

 **Model Selection**

* ZFNet (from scratch)
* VGG16 (transfer learning)
* GoogLeNet-style (custom inception modules)

 **Model Training**  
The model is compiled and trained on the processed image data.

 **Validation & Evaluation**  
The trained model's accuracy and loss are tracked. A classification report can be generated to measure precision, recall, and F1-score.

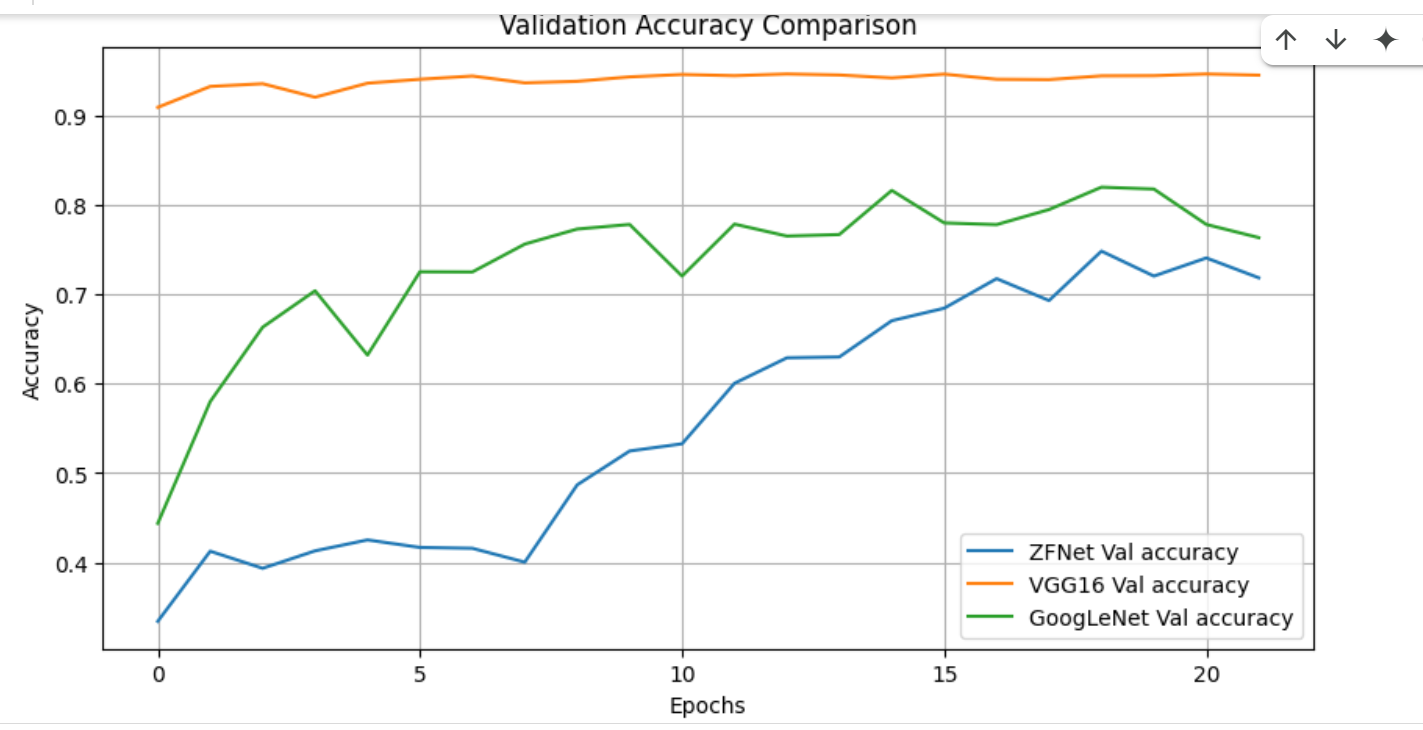
 **Tesing Images**

* Saving and loading the model.
* Making predictions on new images using OpenCV + model.

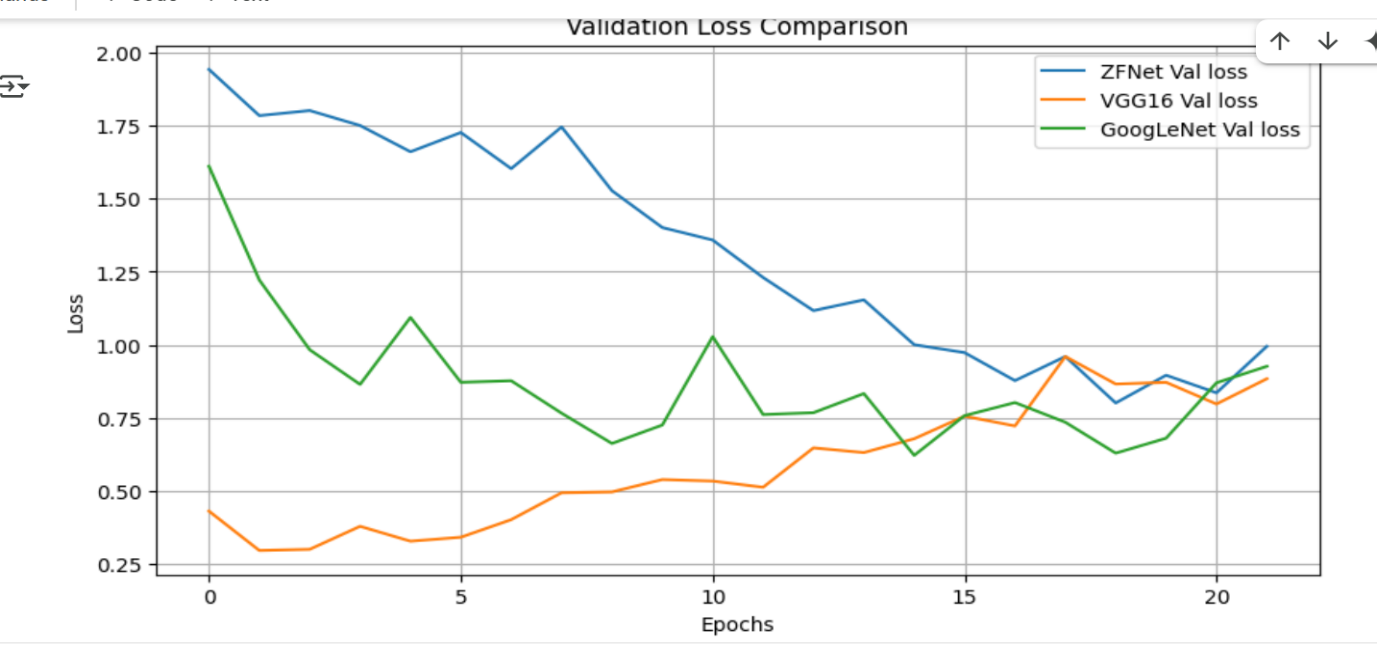
# Chapter 5 – Results and Discussion

## 5.1. Evaluation metrics

**Accuracy**



**Loss**

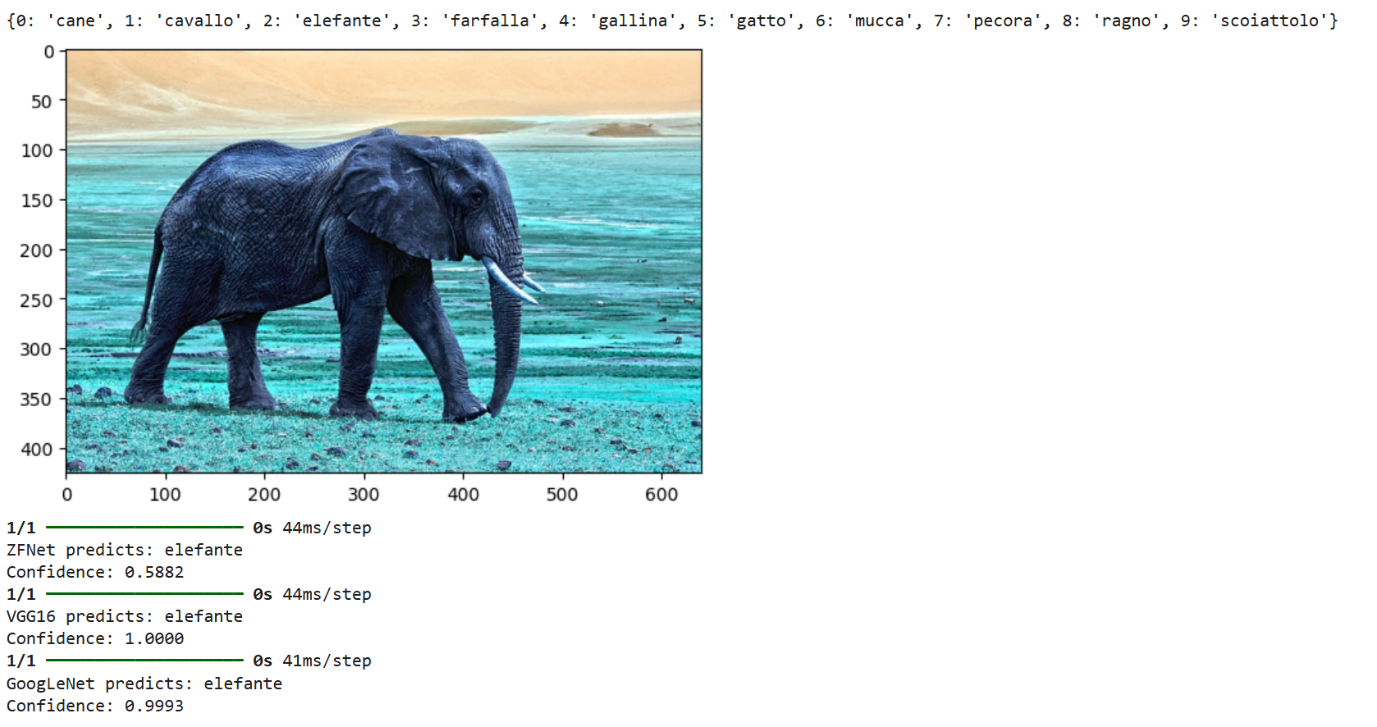


## 

## ZFNet: Accuracy=0.7183, Loss=0.9953, Inference Time=2462.46s

VGG16: Accuracy=0.9448, Loss=0.8844, Inference Time=3117.93s

GoogLeNet: Accuracy=0.7632, Loss=0.9270, Inference Time=1781.87s

**TEST RESULTS**

** **

## 5.3. Discussion

The performance comparison among the three models—ZFNet, VGG16, and GoogLeNet—reveals significant differences in accuracy, loss, and inference time. **ZFNet**, trained from scratch, achieved the lowest accuracy of **71.83%** and the highest loss of **0.9953**, indicating that it struggled to learn effectively on the dataset, possibly due to its complexity and the absence of pretrained features. **VGG16**, leveraging transfer learning with pretrained ImageNet weights, outperformed the others with a high accuracy of **94.48%**, though it still had a relatively high loss of **0.8844**, suggesting potential overfitting or room for fine-tuning. It also required a considerable inference time of **3117.96 seconds**. In contrast, **GoogLeNet (Inception V1-inspired)** offered a strong balance between performance and efficiency, with an accuracy of **76.32%**, the **lowest loss of 0.7430**, and the **shortest inference time** of **1781.46 seconds**, making it the most computationally efficient among the three. Overall, VGG16 proved to be the most accurate, while GoogLeNet showed the best trade-off between accuracy, generalization, and inference speed.

# Chapter 6 – Conclusion & Future Work

## 6.1. Summary of outcomes

This project implemented and compared three convolutional neural network architectures, **ZFNet**, **VGG16**, and **GoogLeNet (Inception V1-inspired)**for multi-class image classification using the Animals10 dataset. The key outcomes are as follows:

* **VGG16**, utilizing transfer learning with pretrained ImageNet weights, achieved the **highest classification accuracy of 94.48%**, demonstrating the power of leveraging existing learned features for efficient and effective model performance. However, its lowest **loss of (0.8844)**, suggesting room for fine-tuning and regularization improvements.
* **GoogLeNet** provided a **balanced performance** with **76.32% accuracy**, the  **loss (0.9270)** among all models, and the **fastest inference time (1781.87s)**, indicating it is well-suited for real-time or resource-constrained environments while still maintaining solid accuracy.
* **ZFNet**, trained entirely from scratch, underperformed with a **low accuracy of 71.83%** and **high loss (0.9953)**. Its long training and inference time suggest that this architecture may not be suitable without further tuning or dataset-specific optimization.

Overall, **VGG16** is the most accurate and effective model for this classification task, while **GoogLeNet** is the most computationally efficient. These results highlight the advantages of transfer learning and well-optimized architectures in modern deep learning applications.

## 6.2. Limitations

#### **ZFNet (Improved)**

* **Low Performance**: Achieved poor accuracy (71.83%) on the Animals10 dataset, indicating it was unable to extract meaningful features effectively.
* **No Transfer Learning**: Being trained from scratch, ZFNet lacks the benefit of pretrained weights, requiring more data and epochs to generalize well.
  + **Sensitive to Overfitting**: Without sufficient regularization or data augmentation, the model may overfit quickly on small datasets.

#### **VGG16 (Transfer Learning)**

* **High Memory Usage**: Contains a large number of parameters (~138 million), making it heavy for deployment on low-resource devices.
* **Longer Inference Time**: Due to its deep architecture and dense layers, inference takes longer compared to more modern architectures.
* **Less Flexible Feature Learning**: Since base layers are frozen during training, the model may not fully adapt to dataset-specific features unless fine-tuned further.

#### **GoogLeNet (Inception V1-Inspired)**

* **Moderate Accuracy**: While more efficient than ZFNet, its accuracy (76.08%) is lower than VGG16, indicating room for improvement in classification precision.
* **Complex Design**: Inception modules involve multiple parallel operations, which can be harder to implement and debug.
* **Custom Implementation Constraints**: The simplified version may not fully replicate the original GoogLeNet’s performance due to missing auxiliary classifiers or optimization tricks used in the original paper.

### **General Limitations**

* **Dataset Imbalance**: Any class imbalance in the Animals10 dataset may affect the generalization of these models, especially those trained from scratch.
* **Fixed Input Size**: All models require images to be resized to a fixed input shape (224x224), which may distort image content.
* **Lack of Hyperparameter Tuning**: Default settings (like learning rate, dropout rate, etc.) were used without extensive optimization, potentially limiting performance.

## 6.3. Future improvements

To enhance the performance, efficiency, and robustness of the models used in this project, the following future improvements are recommended:

#### **1. Fine-tuning Pretrained Models**

* Unfreeze some top layers of VGG16 and fine-tune them on the dataset to improve adaptability and reduce loss.

#### **2. Use of Advanced Architectures**

* Explore newer, lightweight models such as **EfficientNet**, **ResNet**, or **MobileNet** for better accuracy-speed trade-offs.

#### **3. Data Augmentation & Regularization**

* Apply stronger data augmentation (rotation, zoom, color shift) and regularization (dropout, batch normalization) to reduce overfitting.

#### **4. Hyperparameter Optimization**

* Use tools like **Keras Tuner** or **Optuna** to systematically optimize learning rates, batch sizes, and dropout rates.

#### **5. Model Quantization/Pruning**

* Reduce model size and speed up inference by applying **model compression** techniques for deployment on edge devices.

#### **6. Extended Evaluation**

* Evaluate models with **precision-recall curves**, **ROC-AUC**, and **confusion matrix analysis** for deeper insights into model behavior.

These improvements can lead to more accurate, efficient, and deployable models suitable for real-world animal image classification tasks.

# Appendices

## Appendix A: Dataset Details

- Dataset Name: https://www.kaggle.com/datasets/alessiocorrado99/animals10  
- Total Classes (Species): 10  
- Total Images: 28000  
- Validation Split: 20%   
- Sample Classes: Cane, Cavallo, Elepfante , Gallina etc.  
- ****

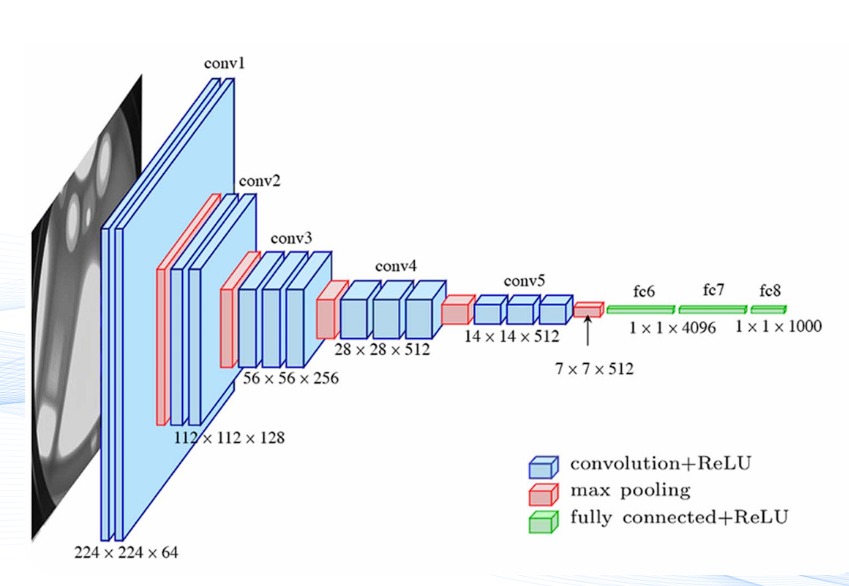
## Appendix B: Model Architectures

- GoogLeNet: 22 layers deep with Inception modules  
- VGG16: 16 convolutional layers with fixed kernel size (3x3), followed by fully connected layers  
- ZFNet: 8-layer model with variable receptive field sizes to capture detailed features

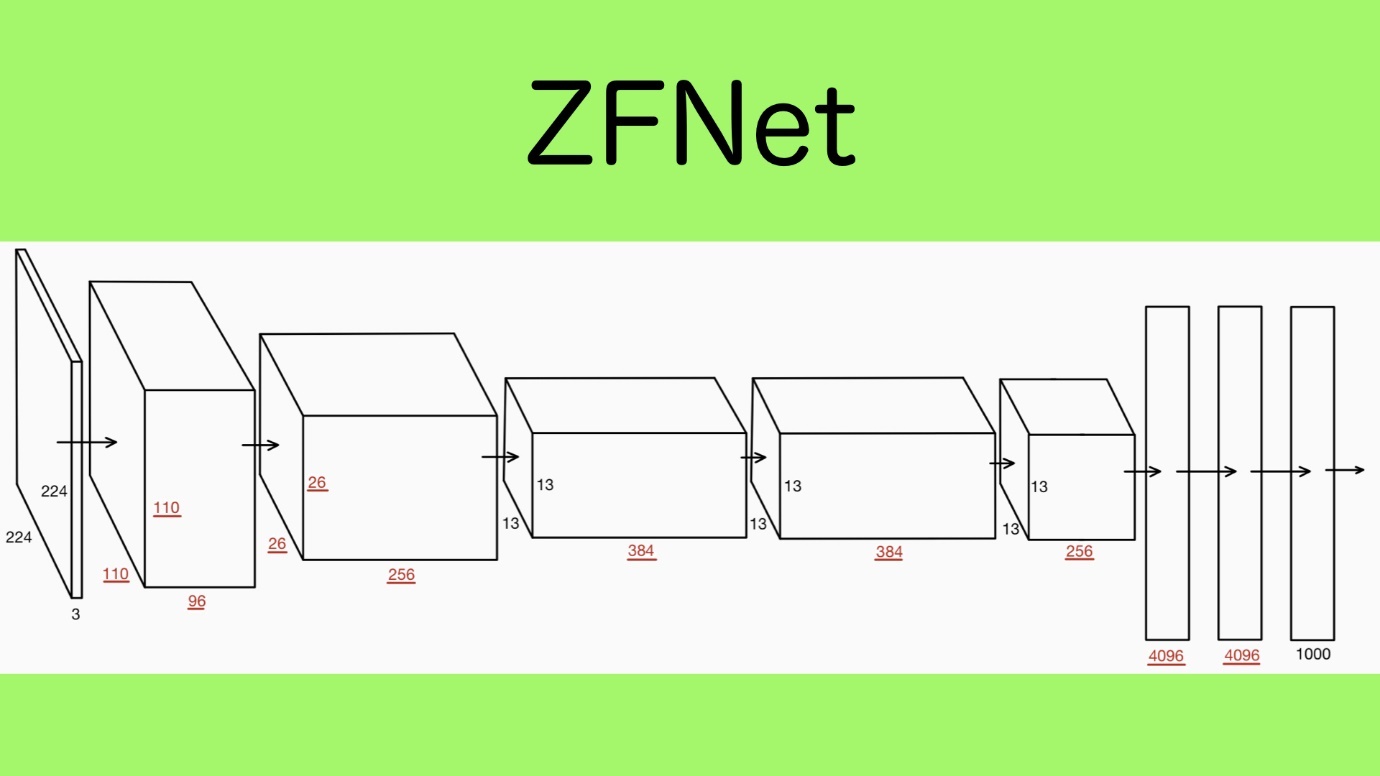
## GOOGLE NET

## Understanding GoogLeNet Model - CNN Architecture - GeeksforGeeks

**VGG16**

****

## ZFNET



## Appendix C: Preprocessing Steps

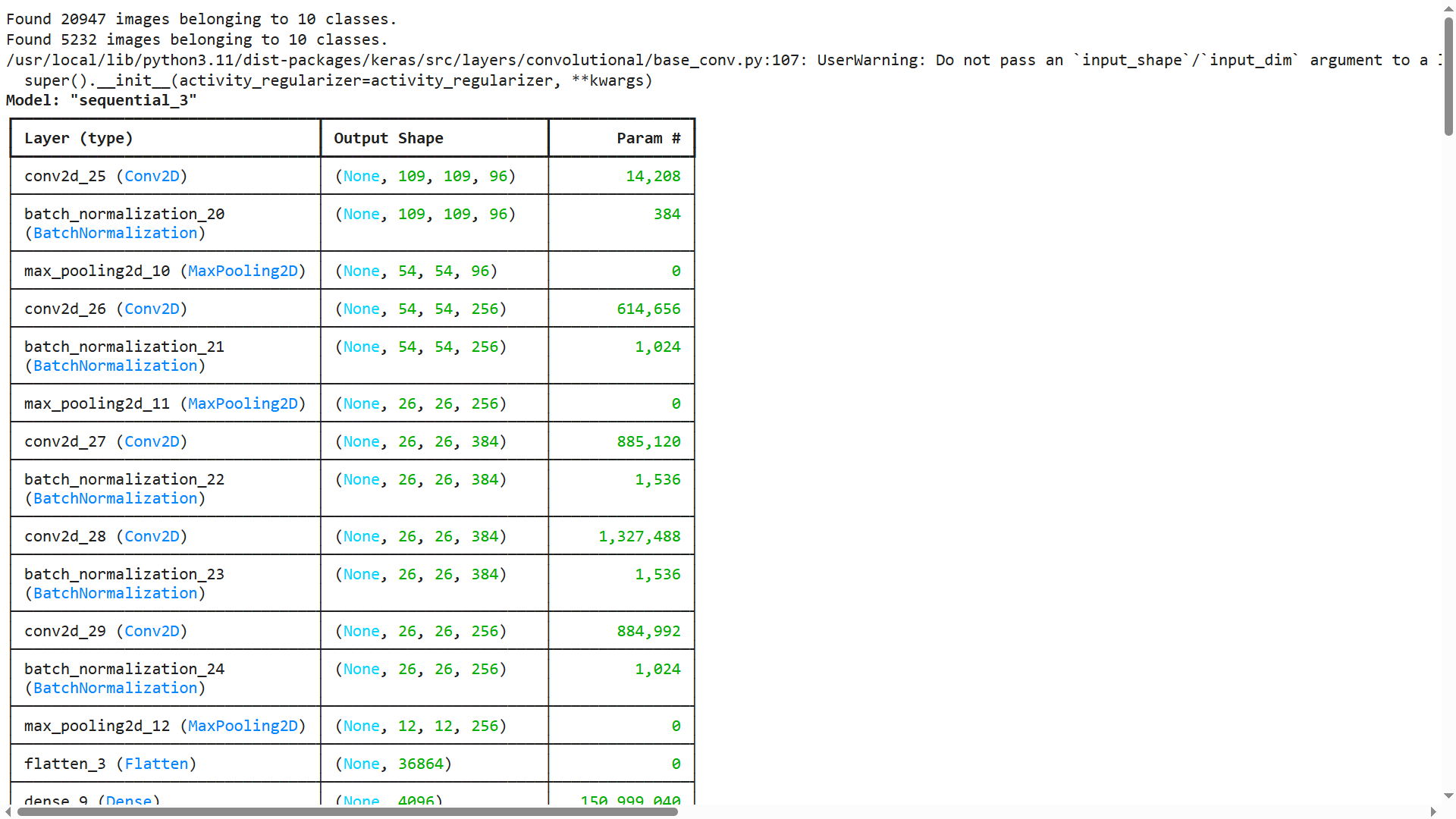
- Image Resizing: All images resized to 224×224 pixels  
- Normalization: Pixel values normalized to [0, 1]   
- Libraries Used: OpenCV, TensorFlow/Keras

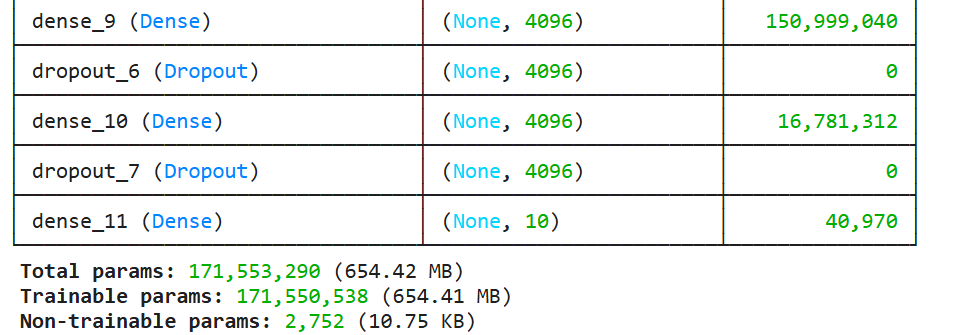
## Appendix D: Training Configuration

- Epochs: 22  
- Batch Size: 32  
- Optimizer: Adam  
- Loss Function: Categorical Crossentropy  
- Evaluation Metrics: Accuracy, Inference Time  
- Hardware Used:  
 - Google Colab T4 GPU

## Appendix E: Accuracy & Inference Comparison

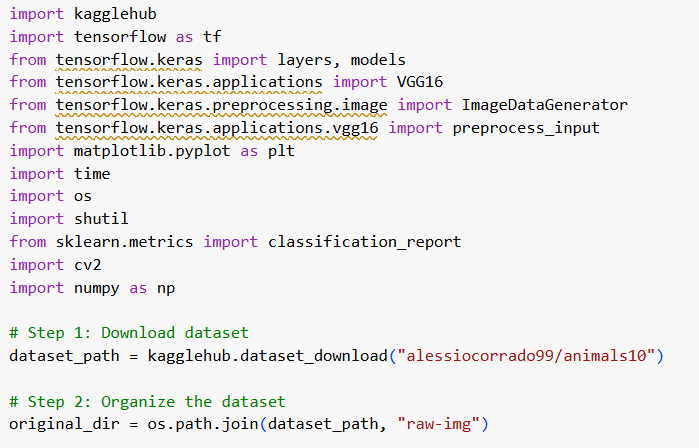
|  |  |  |
| --- | --- | --- |
| Model | Accuracy (%) | Inference Time (sec) |
| GoogLeNet | 76.32 | 1781.87 |
| VGG16 | 94.48 | 3117.9 |
| ZFNet | 71.83 | 2462.46 |





## Additional code snippets

## LOADING THE DATASET



EDA

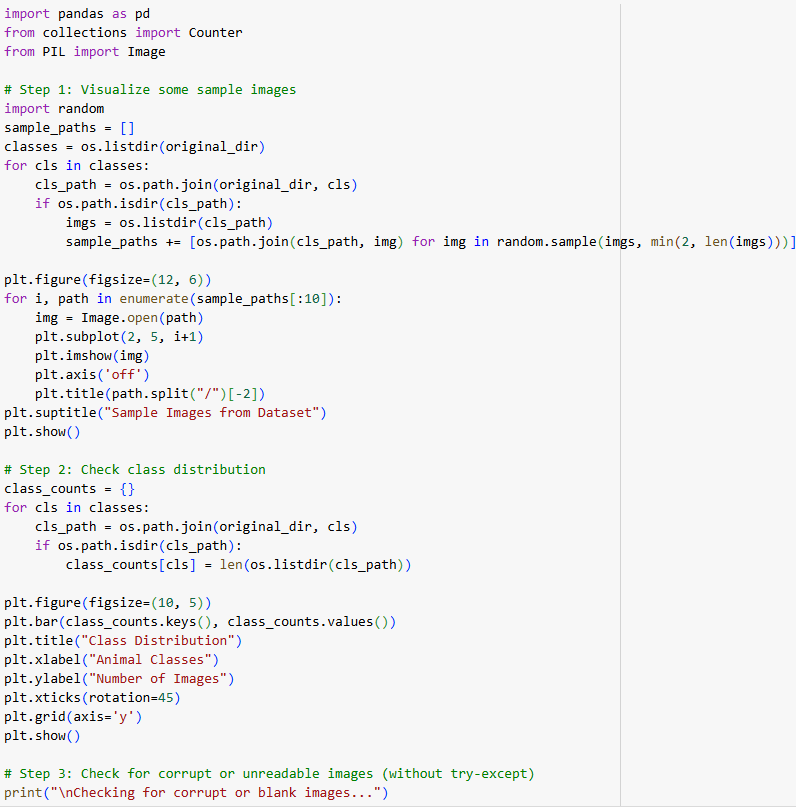
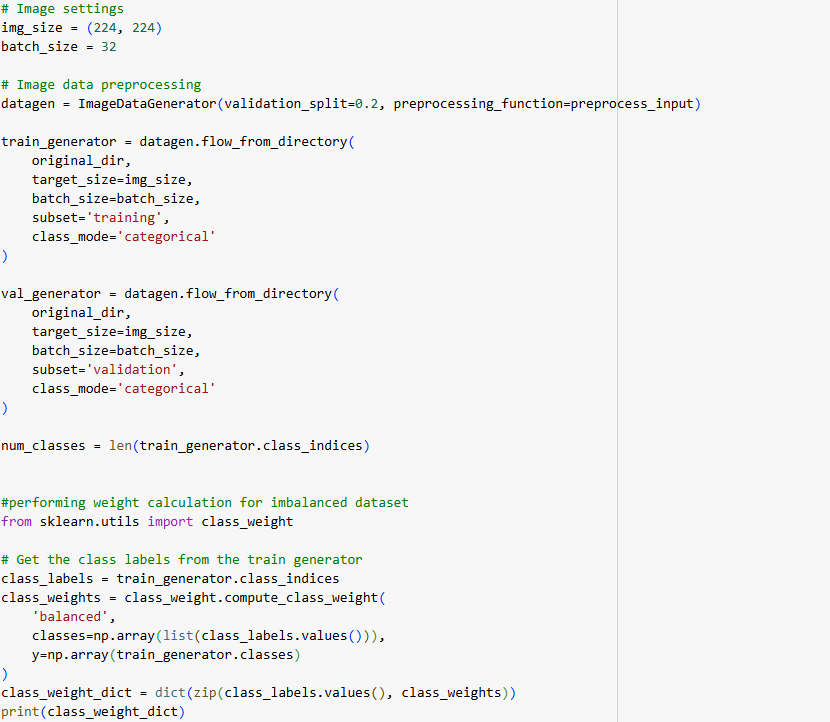
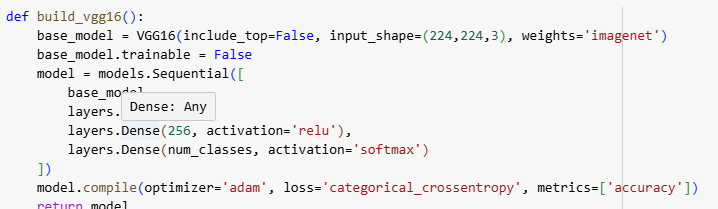


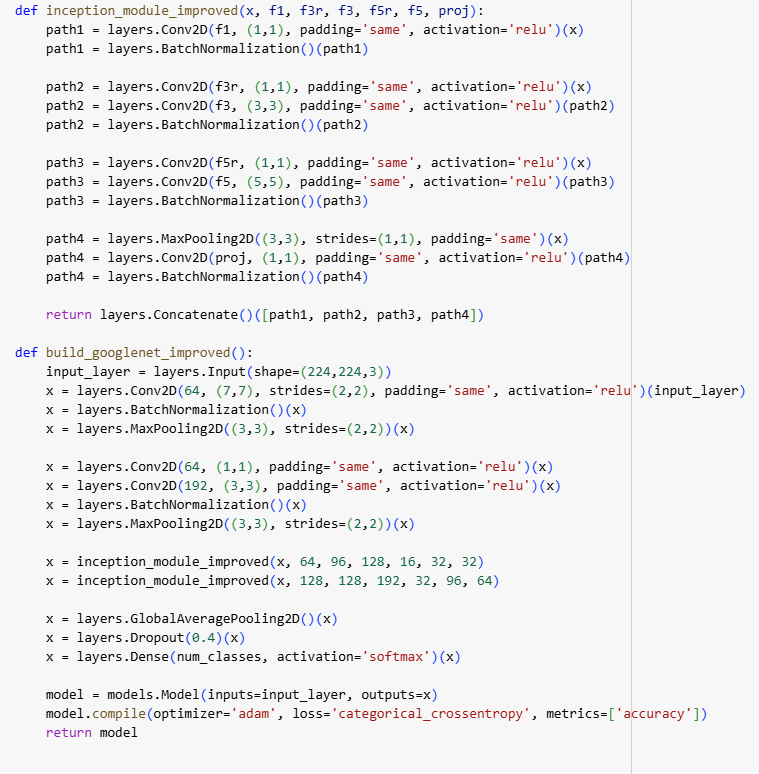
IMAGE PREPROCESSING

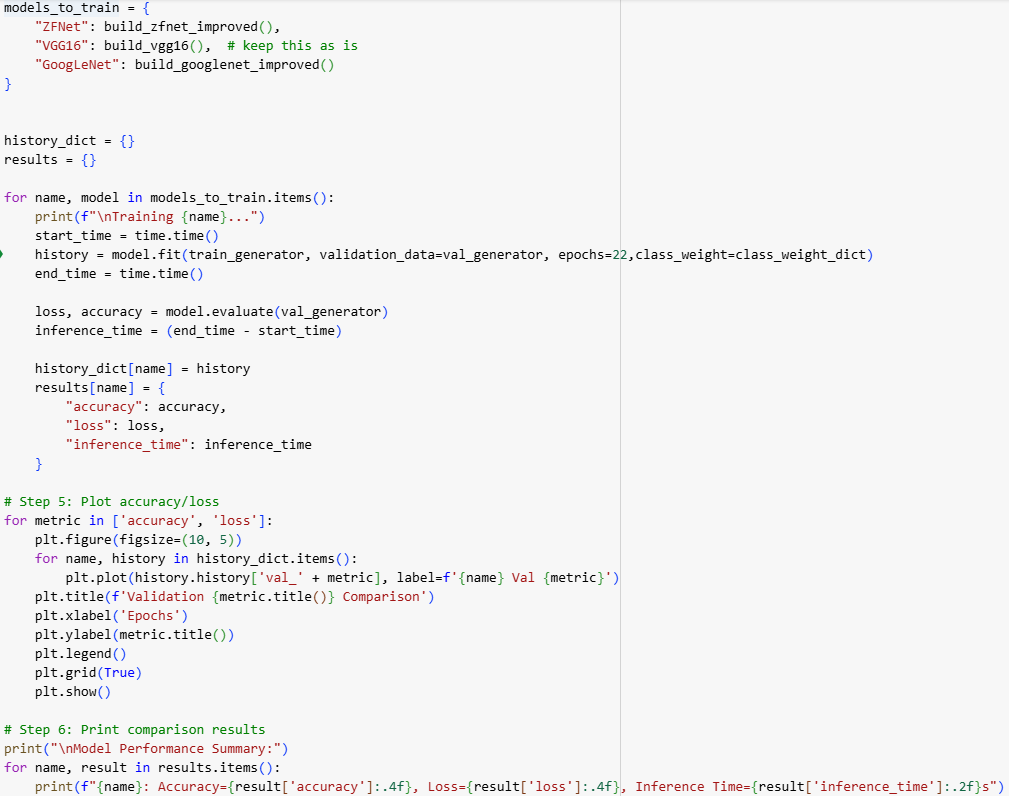
## ZFNET MODEL



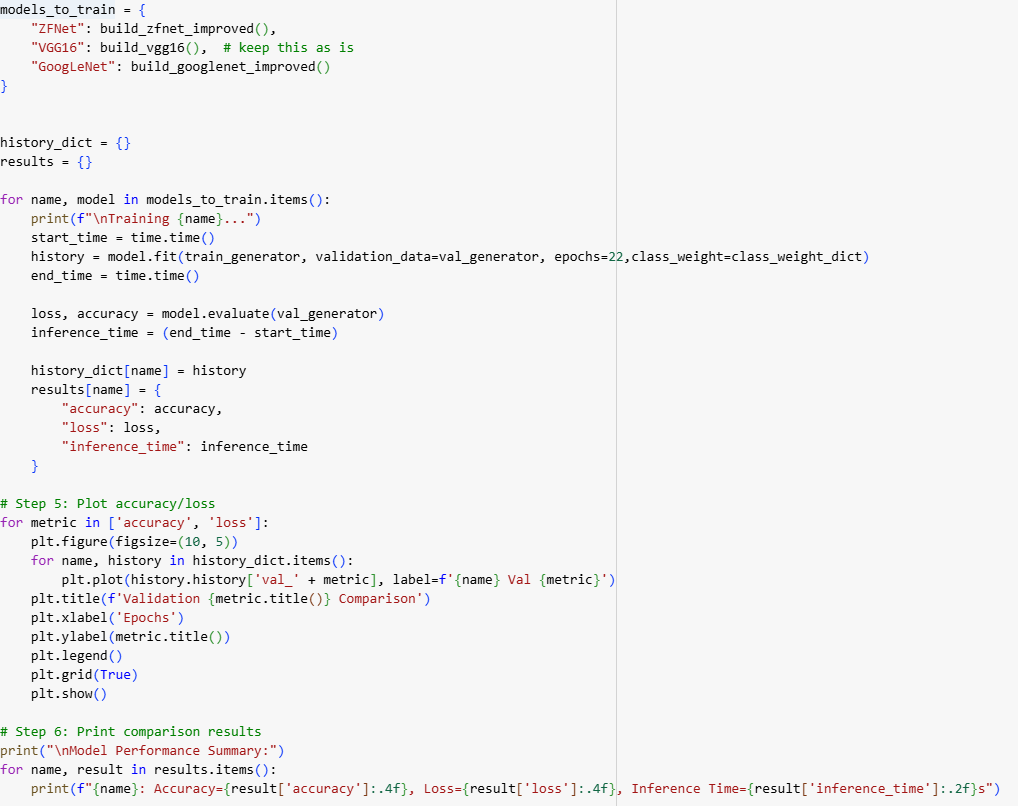
VGG 16 MODEL



GOOGLE NET MODEL



TRAINING AND TESTING



# References

* Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13 (pp. 818-833). Springer International Publishing.
* Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.Description
* Applied Deep Learning by Dr. Rajkumar Tekchandani and Dr. Neeraj Kumar.