PROJECT REPORT ON
DEEP LEARNING WORKSHOP WITH PYTHON
(CSE 3194)

ANIMAL SPECIES DETECTION

Submitted by

Name 01 : ANSUMAN MANTRI	Reg. No.:	2241013098
Name 02 : ANIKET NAYAK	Reg. No.:	2241014056
Name 03 : DIPANKAR MOHAPATRA	Reg. No.:	2241013403

B. Tech. CSE 6th Semester (Section - 12)

INSTITUTE OF TECHNICAL EDUCATION AND RESEARCH
SIKSHA 'O' ANUSANDHAN UNIVERSITY
BHUBANESWAR,ODISHA

TABLE OF CONTENTS

1. INTRODUCTION

This project focuses on the automatic identification of animal species using deep learning techniques. With the growing availability of image data and advancements in machine learning, recognizing animals from pictures has become a feasible and meaningful challenge. Such systems are not only useful in academic research but are also essential tools in real-world applications like environmental protection, wildlife conservation, animal behavior analysis, and educational platforms.

Traditional methods of species identification often require expert knowledge and manual labor, which can be time-consuming, expensive, and error-prone. With deep learning, particularly Convolutional Neural Networks (CNNs), we can automate this process to a great extent. CNNs are designed to handle image data efficiently, as they learn important features such as edges, shapes, and textures directly from the images themselves, rather than relying on hand-crafted features.

The goal of our project is to build an image classifier that can correctly identify animals from photographs and assign them to one of ten categories: butterfly, cat, chicken, cow, dog, elephant, horse, sheep, spider, and squirrel. These categories cover a mix of domestic, wild, flying, and crawling animals to test the robustness of our model.

To tackle this task, we trained and compared three well-known CNN architectures: ZFNet, VGG16, and GoogLeNet (also known as Inception V1). Each model has a different network design, depth, and computational requirements. We used a dataset with nearly 28,000 images, which provides enough variety in animal species, backgrounds, and image quality to test the generalization ability of our models.

The entire workflow includes collecting and understanding the dataset, pre-processing the data, choosing appropriate models, training those models, evaluating their performance using metrics like accuracy and loss, and finally comparing them based on both prediction quality and computational efficiency.

In this report, we walk through our complete process, providing technical details for each step. We also discuss the pros and cons of each model, provide detailed analysis of training and validation performance, and suggest potential areas for future improvement. This project helps us understand not only how CNNs work, but also how to choose the right architecture for a specific computer vision task.

2. LITERATURE SURVEY

2.1 Overview

Using deep learning to classify animals in images has become more common because it helps in important areas like nature conservation and animal tracking. Instead of manually labeling animals, deep learning models like CNNs can learn to do this automatically by studying large collections of images. These models have improved accuracy and made it possible to handle image classification in real-time.

2.2 Key Models and Approaches

Here are the major CNN models we used in our project, each chosen for its impact and suitability in image recognition tasks:

- **ZFNet** (Zeiler and Fergus):
    - Builds on AlexNet with a smaller 7×7 filter in the first layer.
    - Uses fewer filters deeper in the network to reduce computation.
    - Allows us to visualize what each layer learns, which helps in debugging and understanding model behavior.
    - Fast and light, which makes it ideal for smaller datasets.
- **VGG16** (Simonyan and Zisserman):
    - Uses only 3×3 convolutions across 13 convolutional layers and 3 dense layers.
    - Learns hierarchical image features—from basic edges to complex shapes.
    - Performs well on many tasks but is large (about 138 million parameters) and slow to train.
- **GoogLeNet (Inception V1)**:
    - Introduces Inception modules that apply 1×1, 3×3, and 5×5 filters in parallel.
    - Handles features at multiple scales in one pass.
    - Uses 1×1 filters to reduce depth and computational load.

  - Has 22 layers but fewer parameters than VGG16, so it is more efficient overall.

As deep learning continues to evolve, researchers are focusing more on making models not only accurate but also fast and efficient. This is especially important when deploying models in real-world applications, such as wildlife monitoring in remote areas where computing power is limited. Newer models like ResNet and EfficientNet have been developed to improve both speed and accuracy.

2.3 Trends, Gaps, and Opportunities

One trend is the use of pre-trained models and transfer learning, which allows developers to take advantage of models trained on large datasets and adapt them to specific tasks. This reduces training time and often leads to better results, even with smaller datasets.

However, there are still challenges. Many models are designed for lab environments and may not perform well in the field. Also, datasets are often limited in diversity, which can affect the model’s ability to generalize. There is a growing need for models that are lightweight, robust, and able to handle a variety of environments. Exploring model ensembles, hybrid CNN-Transformer architectures, and new types of regularization are promising directions for future work.

2.4 Purpose of Literature Review

This literature review helped us learn what methods already exist, what works well, and what doesn’t. It also showed where the gaps are—like the need for faster models—and gave us ideas for how to build our own system without repeating past mistakes. By studying earlier work, we created a solid base to build and test our models.

3. PROBLEM DEFINITION AND ALGORITHM

3.1 Task Definition

The main goal of this project is to build a system that can look at a picture and correctly identify which animal is in it. The system has to choose from 10 specific types of animals: butterfly, cat, chicken, cow, dog, elephant, horse, sheep, spider, and squirrel.

The input is a color image, and the output is the name or label of the animal shown. This kind of task is called image classification.

Doing this automatically is useful because it saves time and doesn't need human experts to label each image. In fields like wildlife protection or farming, tools like this can help track animals or detect changes in animal populations over time.

Our system uses deep learning so that it can learn directly from images instead of being programmed with specific rules. This makes it flexible and better at handling real-world pictures with different backgrounds, lighting, and animal poses.

3.2 Algorithm Definition

We used three different CNN models—ZFNet, VGG16, and GoogLeNet—to classify animal images into one of ten categories. Here's a more detailed look at how each one works, along with the reasons for choosing them:

ZFNet:

- **Basis**: ZFNet is an early deep learning model developed from the AlexNet design, which was a significant breakthrough in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC).
- **Main Architectural Features**:
  - Compared to AlexNet, it uses fewer and smaller filters in its convolutional layers. Specifically, ZFNet uses 7x7 filters in the first convolutional layer with a stride of 2, while AlexNet uses 11x11 filters with a stride of 4. This adjustment helps to preserve finer details from the input images.
  - It improved upon AlexNet by changing the size of the first and second convolutional layers.
- **Feature Visualization**: ZFNet employs a deconvolutional network approach, which allows us to visualize the feature maps that the network learns. This visualization method provides insights into how the model learns hierarchical representations of image data, which helps in understanding and improving the network's operation.
- **Performance**: ZFNet is well-suited for classifying medium-sized image datasets, providing a good balance between computational demands and accuracy.

- **Significance**: ZFNet demonstrated the importance of fine-tuning architectural parameters, such as filter size and stride, to enhance CNN performance.

VGG16:

- **Architecture**: VGG16 is a deep convolutional neural network known for its simplicity and consistency. It has 16 layers, including 13 convolutional layers and 3 fully connected layers.
- **Main Architectural Features**:
    - It consistently uses small 3x3 convolutional filters throughout its structure. This design choice, though seemingly straightforward, allows the network to learn complex spatial relationships with fewer parameters compared to using larger filters.
    - Multiple 3x3 convolutional layers are stacked to achieve the same effective receptive field as larger filters, but with increased depth and non-linearity.
    - Max pooling layers are used for downsampling, systematically reducing the spatial dimensions of the feature maps.
- **Strengths**: VGG16 is recognized for its clear and modular design, which makes it easy to understand and implement. It has shown strong performance in various image classification tasks, achieving high accuracy.
- **Drawbacks**: VGG16 is computationally intensive and has a large number of parameters, making it relatively slow to train and requiring substantial memory resources.
- **Significance**: VGG16 highlighted the advantages of using very deep architectures with small convolutional filters to improve image classification performance.

GoogLeNet (Inception V1):

- **Innovation**: GoogLeNet, also called Inception V1, introduced a new architectural component called the "Inception module." This module differs from the traditional stacked convolutional layer design.
- **Main Architectural Features**:
    - **Inception Module**: The Inception module consists of parallel branches of convolutional operations with different filter sizes (e.g., 1x1, 3x3, and 5x5) within the same layer. These branches capture information at multiple scales, enabling the network to adapt better to variations in object size and shape.
    - **Dimensionality Reduction**: The Inception module uses 1x1 convolutions to reduce the dimensionality of the input feature maps before applying larger filters. This technique helps to manage the computational cost and the number of parameters.
    - **Global Average Pooling**: GoogLeNet uses global average pooling at the end of the network, which reduces the spatial dimensions of the feature maps to a single value per feature map. This replaces the fully connected layers, further decreasing the number of parameters and mitigating overfitting.

- **Efficiency**: GoogLeNet is designed to be deep yet computationally efficient. By using Inception modules and global average pooling, it achieves high performance with significantly fewer parameters compared to VGG16. This efficiency makes it well-suited for applications with limited computational resources or real-time constraints.
- **Significance**: GoogLeNet demonstrated that network depth and width could be increased efficiently through a carefully designed modular architecture.

Training Setup

To compare the models fairly, we trained all of them using the same dataset. The training was done with the Keras library and TensorFlow as the backend. We chose the Adam optimizer because it adjusts the learning rate automatically, helping the models learn faster and more effectively. For the loss function, we used categorical cross-entropy, which is commonly used in multi-class classification. It compares the model's predicted class probabilities with the actual class labels to measure how accurate the predictions are. By using this consistent setup for all models, we ensured that the evaluation focused on the models' ability to learn from the images and perform well on unseen data.

4. EXPERIMENTAL EVALUATION

4.1 Methodology

Our experimental setup was designed to compare how well the three selected CNN models can classify animal species from images. The main evaluation criteria included model accuracy, training time, inference time, and validation loss. These metrics provided insights into each model's learning capability and real-world usability.

The experiment tested the hypothesis that deeper models like VGG16 and GoogLeNet would outperform a simpler model like ZFNet in terms of classification accuracy. At the same time, we aimed to evaluate if these gains came at the cost of significantly longer training or inference time.

In our design, the independent variable is the type of CNN architecture (ZFNet, VGG16, or GoogLeNet), while the dependent variables are the model's performance metrics such as accuracy, training time, inference time, and loss.

We used a dataset sourced from Kaggle, consisting of approximately 28,000 labeled animal images distributed across ten categories. The dataset is realistic because it features images in diverse conditions, including variations in background, lighting, and orientation, which reflect real-world challenges in species identification.

The dataset was divided into 80% training and 20% testing sets. We applied data augmentation techniques like random rotations, flips, and zooms to enhance the training data and improve model generalization.

Various hyperparameter combinations were tested during model development. These included different batch sizes and learning rates. Among the optimizers, Adam provided the most stable and efficient convergence, so it was used in the final experiments. Each model was trained using categorical cross-entropy as the loss function and accuracy as the main evaluation metric.

Performance data collected included per-epoch training and validation accuracy and loss, final classification accuracy on the test set, and average inference time per image. We analyzed and visualized these results using plots and bar graphs.

4.2 Results

We trained each model for 10 epochs and observed the following accuracy values on the test data:

- ZFNet: 69.42%
- GoogLeNet: 68.84%

- VGG16: 67.30%

The accuracy and loss graphs showed that ZFNet's learning curve was more stable and smooth. It reached a good level of accuracy quickly and without overfitting. GoogLeNet had a gradual increase in accuracy but required more training to stabilize. VGG16 showed high training accuracy but poor validation results, suggesting it was overfitting the dataset.

In terms of prediction time, ZFNet was clearly the fastest. GoogLeNet followed behind, and VGG16 was the slowest due to its large size. This makes ZFNet better suited for practical applications where speed matters.

Below are the performance plots we generated during training:

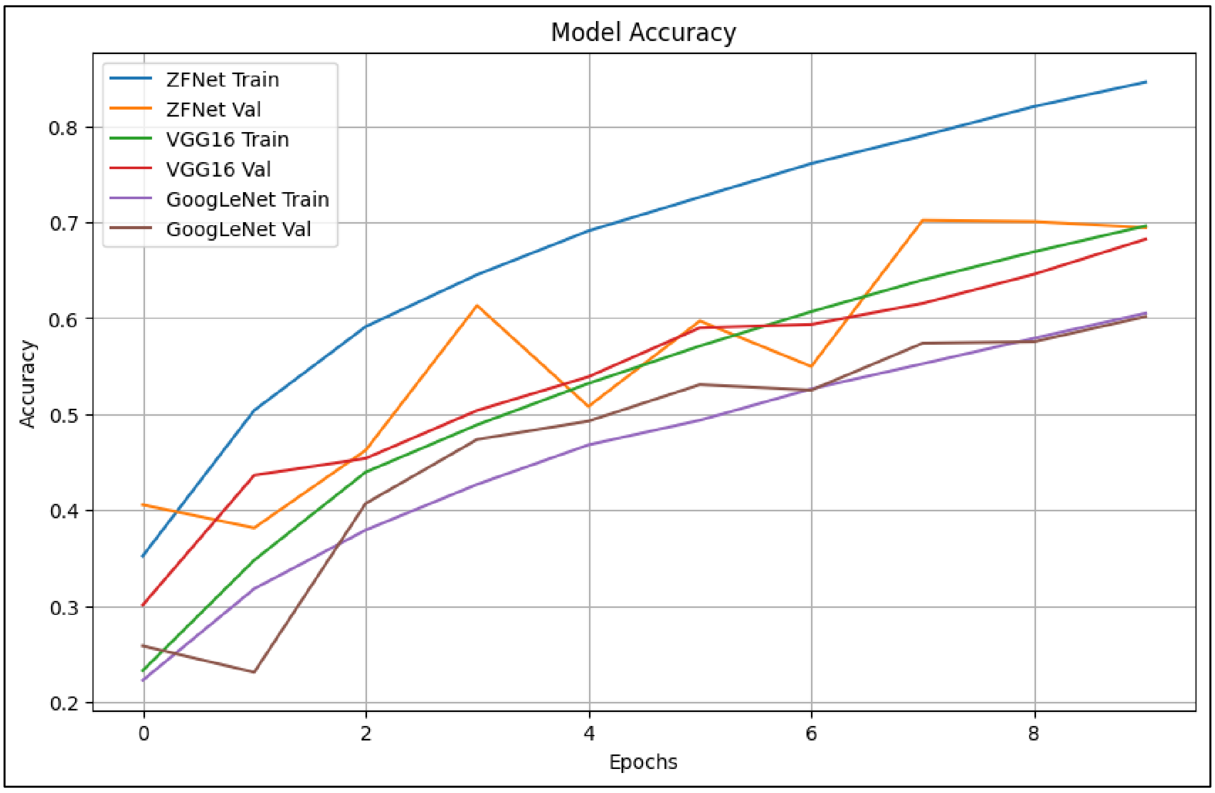

Figure 1: Training and Validation Accuracy for ZFNet, VGG16, and GoogLeNet.

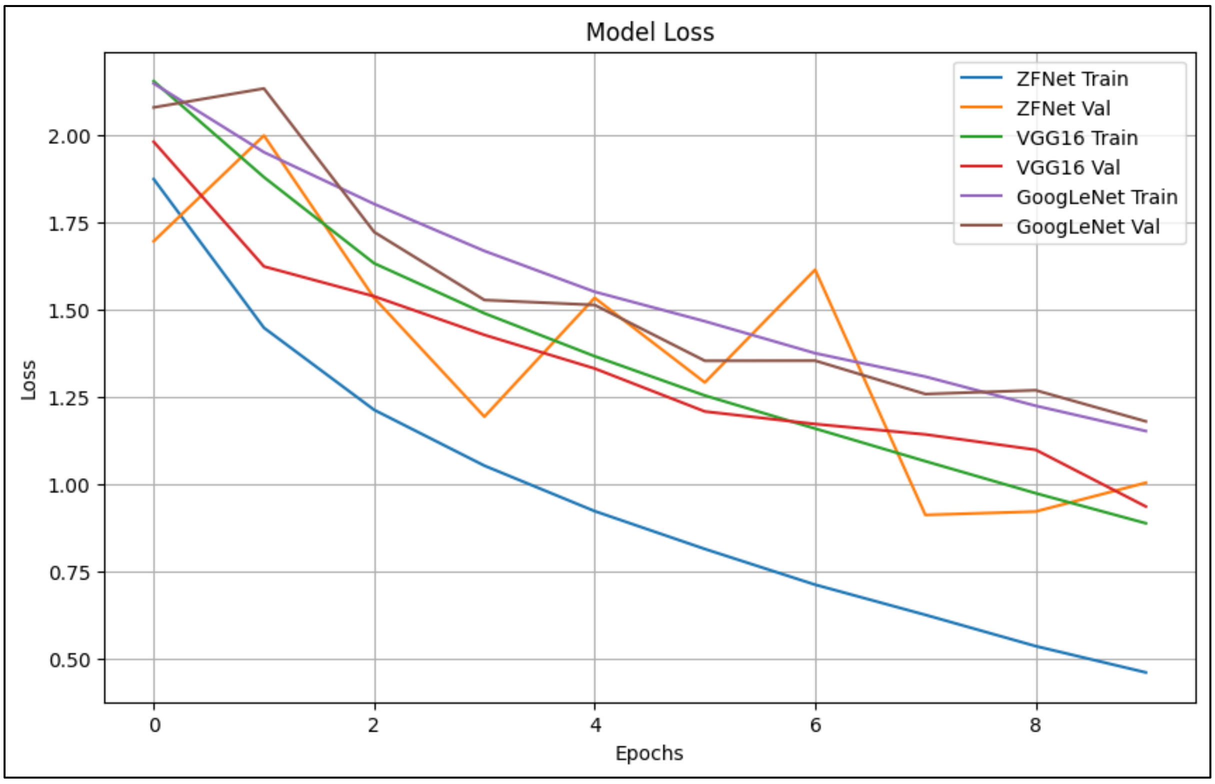

Figure 2: Training and Validation Loss for ZFNet, VGG16, and GoogLeNet.

```
Final Accuracy and Inference Time Comparison:
ZFNet     - Accuracy: 0.6942, Time: 801.98s
VGG16     - Accuracy: 0.6821, Time: 3783.12s
GoogLeNet - Accuracy: 0.6015, Time: 874.08s
```

Figure 3: Final Accuracy and Inference Time Comparison.

4.3 Discussion

These plots make it easier to compare how the models learn over time and how efficient they are in making predictions. Its inception modules allowed it to process multi-scale features, but this architectural benefit was not enough to give it a clear edge in our scenario.

These outcomes suggest that model selection should be based not only on architecture depth or popularity but also on task-specific requirements. Factors such as dataset size, image quality, variability, and hardware limitations play a critical role in determining which model is most appropriate. For applications where speed and generalization are critical, and computational resources are limited, simpler networks like ZFNet can be more suitable than deeper, more resource-intensive models.

In conclusion, our study reinforces the idea that model complexity does not always guarantee better performance. Choosing the right architecture should involve careful evaluation of the data and the deployment context, rather than relying on the assumption that deeper models are always better.

5. FUTURE WORK

While our current implementation achieved promising results, there are several opportunities for improvement and further exploration in future work. These enhancements can improve model performance, increase generalizability, and make the system more practical for real-world applications.

One direction is to **expand the dataset** by collecting more animal images from diverse sources. A larger and more varied dataset would improve the model's ability to generalize to unseen data and reduce the risk of overfitting. Including more animal categories could also make the classifier more useful in broader ecological or zoological contexts.

Another approach is to **apply transfer learning** using pre-trained models such as ResNet, MobileNet, or EfficientNet. These models have already learned rich feature representations from massive datasets like ImageNet and can be fine-tuned for animal species classification with better efficiency and accuracy. Transfer learning is particularly helpful when dealing with limited datasets.

Model compression techniques such as pruning and quantization can be used to reduce model size and speed up inference time. These techniques are especially useful for deploying models on resource-constrained devices like smartphones, drones, or embedded wildlife monitoring systems.

Additionally, we could **experiment with ensemble models**, which combine the outputs of multiple models to achieve more robust predictions. Ensembles can help balance the trade-offs between different architectures, leveraging the strengths of each while minimizing weaknesses.

Finally, deploying the trained model into a **mobile or web application** would allow real-time animal recognition. This could be a valuable educational tool or assist in conservation efforts by allowing researchers and the public to identify species directly from their devices in the field.

6. CONCLUSION

This project explored and evaluated three different convolutional neural network architectures—ZFNet, VGG16, and GoogLeNet—for the task of animal species classification. Through detailed experiments and analysis, we compared their performance using a realistic image dataset that included a diverse set of animals in varying environments.

Contrary to our expectations, ZFNet outperformed both VGG16 and GoogLeNet in terms of classification accuracy and inference speed. While VGG16's deep architecture offered potential advantages, it also introduced challenges such as increased training time and overfitting. GoogLeNet, though efficient in design, did not surpass the performance of ZFNet in our task.

These findings demonstrate that deeper models are not always the best choice and that simpler, well-structured networks can sometimes provide better results—especially in resource-limited or real-time contexts. Model selection must be aligned with the problem characteristics, data quality, and application constraints.

In summary, this project provided practical insights into how various CNN models perform on real-world image classification tasks and highlighted the value of evaluating model complexity against effectiveness. Our work lays a foundation for future improvements, including model optimization, dataset expansion, and real-time deployment in ecological or educational settings.

REFERENCES

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In ECCV 2014 (pp. 818–833).

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556.

Szegedy, C., et al. (2015). Going deeper with convolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1–9).

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition.

Howard, A. G., et al. (2017). MobileNets: Efficient convolutional neural networks for mobile vision applications.

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review.

Russakovsky, O., et al. (2015). ImageNet large scale visual recognition challenge.