Animal image classification using transfer learning with ResNet50

Álvaro García Vásquez
agarciav2102@alumno.ipn.mx

***Abstract*—This paper presents an animal classification system developed using transfer learning with the ResNet50 convolutional neural network pre-trained on ImageNet. Designed to distinguish between three classes of animals—cats, dogs, and snakes—the system demonstrates a high accuracy of approximately 98.67% on a balanced dataset comprising 3,000 images. The methodology includes data preprocessing steps, model fine-tuning, and the implementation of data augmentation techniques to enhance generalization. Additionally, a custom user interface developed with Tkinter facilitates user interaction, allowing real-time classification with immediate predictions and confidence scores. This study underscores the efficacy of transfer learning in image classification tasks and highlights the potential of such systems in various application settings.**

Keywords—transfer learning, ResNet50, image classification, computer vision, deep learning, Tkinter, user interface, animal recognition.

I. Introduction

The field of computer vision has experienced a quantum leap in progress with the advent of deep learning, particularly in tasks such as image classification. The ability to accurately classify images into predefined categories is crucial in various applications, ranging from biological studies to the enhancement of social media experiences.

This report details the development of an image classification system utilizing a deep neural network, namely the ResNet50 model pre-trained on the ImageNet dataset. The model is fine-tuned to classify images into three distinct categories: cats, dogs, and snakes. The selection of these specific classes stems from the goal to not only capture the versatility in domestic and wild animals but also to demonstrate the model's capability to discern fine-grained features and generalize across varied datasets.

Emphasis is placed on the dataset's construction, ensuring equal representation and diversity to promote unbiased learning. The incorporation of transfer learning accelerates the training process and capitalizes on the pre-existing high-level feature recognition learned by ResNet50 on ImageNet. To counteract overfitting and enhance the model's ability to generalize to new images, data augmentation techniques such as rotation, scaling, and flipping are employed.

Furthermore, the project underscores the importance of user experience by integrating a Graphical User Interface (GUI) developed with Tkinter. This interface bridges the gap between the theoretical aspects of deep learning models and practical, user-friendly applications. The GUI invites users to upload images and receive real-time classifications, making the system accessible to a broader audience.

II. Related Work

In the evolution of image classification, the development of Convolutional Neural Networks (CNNs) marked a significant milestone. The introduction of the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) led to the development of AlexNet, which significantly outperformed existing methods in image classification tasks. This success ushered in a wave of advancements in deep learning architectures, with ResNet being one of the most notable. ResNet's innovative use of residual connections allowed the training of very deep networks, significantly improving performance on challenging image recognition tasks.

Parallel to these advancements in deep learning, there has been a growing interest in applying these technologies to the field of wildlife conservation. Automated animal classification systems have been developed using camera trap images to monitor and study biodiversity, which is crucial for conservation efforts.

Furthermore, the integration of machine learning models with user interfaces has become a focus area, aiming to make these powerful tools more accessible and user-friendly. Tools like TensorFlow.js have been developed to run machine learning models directly in the browser, allowing users to interact with these models in real-time.

This project draws on these developments, applying the ResNet50 model, pre-trained on ImageNet, to the task of animal classification. The goal is to differentiate between various common animals, demonstrating the model's practical application through a user-friendly graphical interface developed in Tkinter. This approach not only showcases the capabilities of modern machine learning models in image classification but also emphasizes the importance of user accessibility in the application of such technologies.

III. Methodology

This section elaborates on the systematic approach adopted in the development of the animal classification system, reflecting the procedural steps executed in the project's Python notebook.

A. Importing Libraries

The project commenced with importing essential Python libraries and modules. Key imports included:

- TensorFlow and Keras for leveraging the ResNet50 model, facilitating neural network operations and model architecture customization.

- ImageDataGenerator from Keras for advanced image preprocessing, enabling real-time data augmentation and normalization, crucial for model training efficiency and effectiveness.
- NumPy for array manipulations, supporting efficient handling and transformation of image data.
- Matplotlib and Seaborn for visualizing training progress, model accuracy, and generating confusion matrices, offering insights into the model's performance.
- Sklearn for model evaluation metrics, aiding in a comprehensive analysis of the model's classification abilities.
- Tkinter for developing the graphical user interface (GUI), making the model accessible for real-time classification by end-users.

B. Data Preprocessing

In this phase, the dataset comprising 3,000 images was meticulously prepared to meet the model's input specifications. Key steps included:

- Resizing images to 256x256 pixels to streamline the process.
- Categorizing images into three distinct classes (cats, dogs, and snakes) to facilitate targeted training and accurate classification.
- Splitting the dataset into training (70%), validation (10%), and testing (20%) sets, ensuring a balanced distribution for comprehensive model learning and evaluation.

C. Loading the ResNet50 Model

The ResNet50 architecture, renowned for its efficacy in image recognition tasks, was loaded with ImageNet pre-trained weights, excluding the top layer to accommodate the classification of three unique animal categories. A Global Average Pooling (GAP) layer followed by a Dense layer with a softmax activation function were appended to the pre-trained base, crafting a model tailored for the specific classification task.

D. Data Augmentation

To enhance the model's ability to generalize and reduce overfitting, extensive data augmentation was employed using ImageDataGenerator. Techniques included:

- Random rotations, width and height shifts to simulate different orientations and perspectives.
- Shear transformations and zooms to mimic varied image scales and angles.
- Horizontal flips to represent mirror images, increasing the dataset's diversity.

E. Model Training

The model training encompassed the following critical steps:

- Configuring the Adam optimizer for efficient backpropagation and adjusting the learning rate to optimize network weights effectively.
- Employing categorical cross-entropy as the loss function to quantify the discrepancy between actual and predicted outputs.
- Executing training over 10 epochs with a batch size of 32, while monitoring validation loss for early stopping to prevent overfitting and optimize model performance.

F. Model Evaluation

Post-training, the model underwent thorough evaluation on the test dataset, deriving key performance metrics such as accuracy, precision, recall, F1-score, and loss. These metrics illuminated the model's classification prowess and its predictive reliability.

G. Saving the Model

Upon achieving satisfactory performance metrics, the model was saved to disk. This allows for persistent future use in application deployment or further refinement and testing.

H. User Interface

A user-friendly GUI was developed using Tkinter, offering users the ability to upload and classify images seamlessly. The interface includes features like image selection, real-time classification display, and probability scoring, enhancing the interactive experience and demonstrating the practical utility of the classification model.

IV. Model Evaluation

The model's generalization was rigorously tested using a separate dataset reserved for testing. The evaluation metrics obtained were test accuracy and test loss, which reflect the model's predictive performance and its confidence in the accuracy of those predictions, respectively.

The test accuracy achieved attests to the model's capability to correctly classify images it has not previously encountered, confirming the effectiveness of the training regimen. Concurrently, the test loss provides insight into the model's precision in predicting new data, with a lower value indicating a higher confidence in the model's classifications. These metrics serve as the cornerstone for gauging the model's potential for deployment in real-world scenarios.

The model's performance was quantitatively evaluated using several metrics: accuracy, precision, recall, and the F1-score. Each metric provides insight into different aspects of the model's predictive capabilities.

Accuracy is the most intuitive performance measure and it is simply a ratio of correctly predicted observation to the total observations. Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. High precision relates to the low false positive rate. Recall (Sensitivity) - the ratio of correctly predicted positive observations to the all observations in actual class - is particularly important when the costs of false negatives are high. The F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. It is a good way to show that a classifer has a good value for both recall and precision.

And finally, the loss metric was tracked during the training and validation phases, providing an indication of the model's prediction error. A lower loss value indicates a more accurate model, which is less confident in incorrect predictions.

The confusion matrix and other detailed metrics such as the classification report further reveal the model's performance across individual classes, highlighting any specific strengths or weaknesses in classifying certain animals.

V. User Interface

The user interface (UI), developed using the Tkinter library in Python, serves as an interactive platform for real-time classification with the trained model. The UI is designed to be user-friendly and intuitive, allowing users to upload images for classification effortlessly.

Upon launching the application, users are presented with a clean and straightforward layout, featuring a central display area for image preview, a classification result section, and a probability score that indicates the model's confidence in its prediction. The main functionalities are:

- **Image Selection:** Users can select an image from their file system through the 'Select Animal' button, which opens a file dialog window.
- **Image Display:** The selected image is displayed on the screen, automatically resized to fit the model's input requirements of 256x256 pixels.
- **Classification and Probability Display:** Upon image selection, the model classifies the image, and the result, along with the probability score, is displayed below the image. The model distinguishes between the three classes - cats, dogs, and snakes, offering insights into its decision-making process.
- **Exit Function:** A dedicated 'Exit' button allows users to close the application at any time, ensuring a smooth and user-controlled interaction experience.

This UI not only enhances the applicability of the animal classifier model but also provides an educational tool for users to explore and understand machine learning concepts in image classification.

VI. Results

The testing and validation of the animal classifier reveal its high degree of accuracy in correctly identifying images of cats, dogs, and snakes. The following segments offer a comprehensive overview of the model's performance metrics, present a nuanced view of its predictive accuracy, and visually highlight its classification prowess through selected prediction examples.

A. Training History

The figure below illustrates the training and validation accuracy and loss over each epoch, providing insight into the learning progress of the model and its convergence behavior over time.

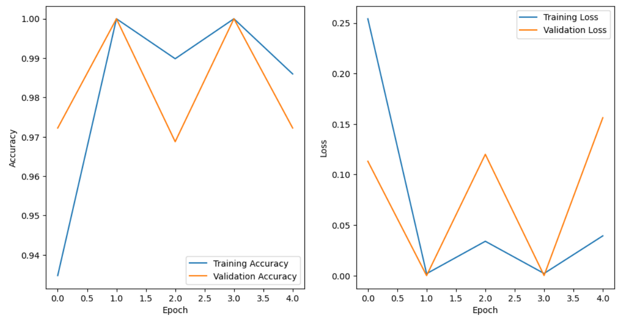

Fig. 1. Training and validation accuracy and loss across epochs.

B. Model Accuracy

The table below summarizes the performance of the animal classifier model in terms of accuracy.

Metric	Value (%)
Accuracy	0.9866666793823242
Loss	0.05052186921238899

TABLE I
Model evaluation.

C. Model's Performance

A detailed assessment of the model's performance on the test dataset is summarized in the table below, highlighting key metrics that demonstrate the model's classification effectiveness.

Class	Precision	Recall	F1-score	Support
Cats	0.98	0.97	0.98	200
Dogs	0.97	0.98	0.98	200
Snakes	1.00	1.00	1.00	200
Accuracy			0.98	600
Macro avg	0.99	0.98	0.98	600
Weighted avg	0.99	0.98	0.98	600

TABLE II
Classification report for the animal classifier model.

D. Confusion Matrix

The confusion matrix provides a visual representation of the model's performance across the different classes. The matrix illustrates the number of true positive, false positive, true negative, and false negative predictions for each class, offering insight into the model's classification accuracy and potential areas of misclassification.

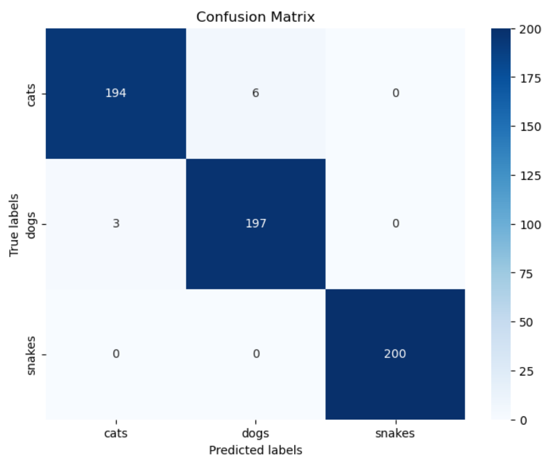

Fig. 2. Confusion matrix of the animal classification model.

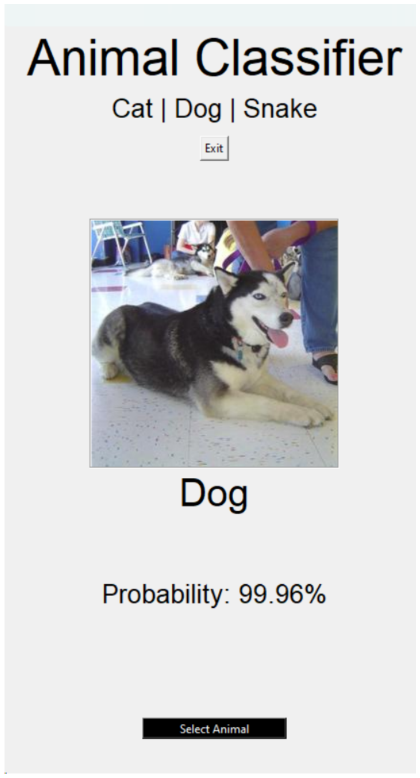

Fig. 4. Example of a dog correctly classified.

E. Example Predictions

Five examples of model predictions are provided, showcasing its ability to accurately classify different animal images.

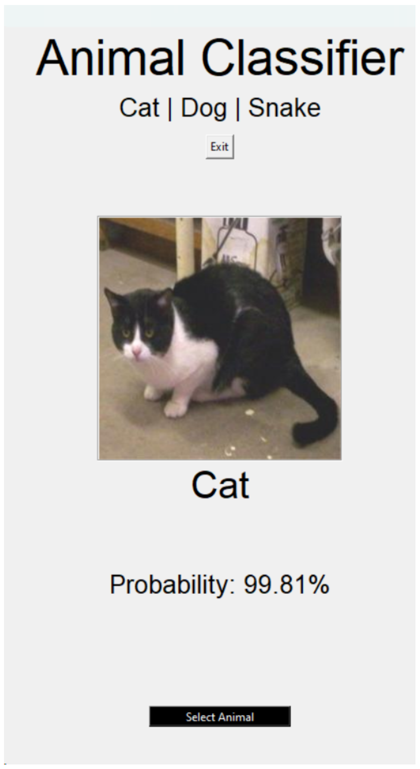

Fig. 3. Example of a cat correctly classified.

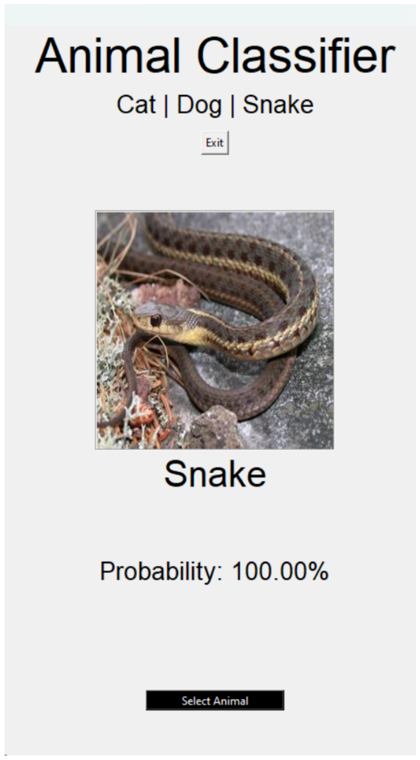

Fig. 5. Example of a snake correctly classified.

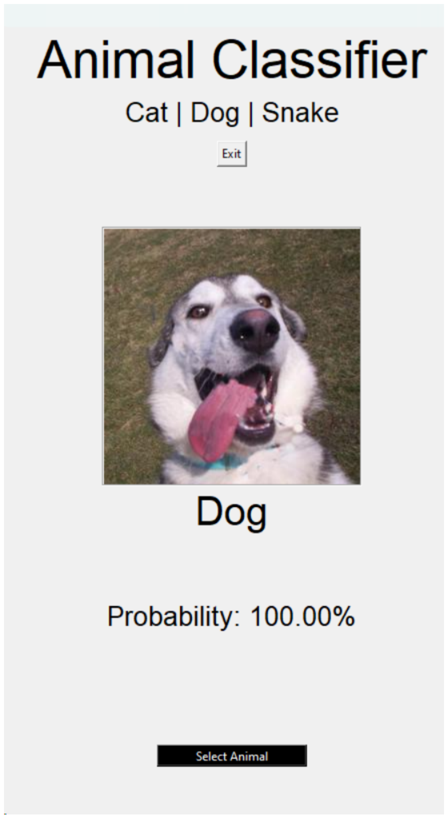

Fig. 6. Example of a dog correctly classified.

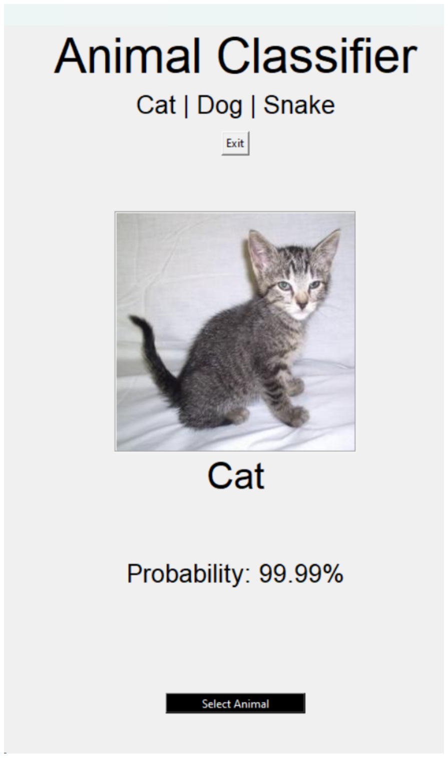

Fig. 7. Example of a cat correctly classified.

VII. Conclusion

In this project, I successfully developed and evaluated an animal image classification system leveraging the power of transfer learning with the ResNet50 model, pre-trained on the ImageNet dataset. The system demonstrated high accuracy in classifying images of cats, dogs, and snakes, which underscores the effectiveness of transfer learning in leveraging pre-trained models for specific tasks with limited datasets.

The integration of a user-friendly interface using Tkinter facilitated real-time interaction with the model, making advanced machine learning models more accessible to non-experts. This project not only showcased the potential of deep learning in image classification but also highlighted the importance of user experience in the development of machine learning applications.

Future work could focus on expanding the dataset to include more animal classes and exploring more advanced deep learning models to improve classification accuracy. Additionally, deploying the model as a web or mobile application could further enhance its accessibility and practical utility.

In conclusion, this project exemplifies how transfer learning can be harnessed to create effective and user-friendly machine learning solutions, paving the way for broader applications in educational tools, wildlife conservation, and beyond.

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep Learning*, MIT Press, 2016, http://www.deeplearningbook.org.
[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep Residual Learning for Image Recognition*, 2015, https://arxiv.org/abs/1512.03385.
[3] Martín Abadi, Ashish Agarwal, Paul Barham, et al., *TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems*, 2015, https://www.tensorflow.org/.
[4] François Chollet, *Deep Learning with Python*, Manning Publications, 2017.
[5] Guido van Rossum and Fred L. Drake Jr., *Python 3 Reference Manual*, CreateSpace, Scotts Valley, CA, 2009.
[6] Mark Roseman, *Modern Tkinter for Busy Python Developers*, 2012.