Final Project

Convolutional Neural Network for Animal ID Classification

Prepared by:
Utsav Lamichhane

Introduction

In this project, I tried on a Convolutional Neural Network (CNN) to classify the identity of animals based on top-view images. The motivation for this endeavor stems from the pivotal role that precise animal identification plays in both behavioral studies and health monitoring in a controlled environment. The application of a deep learning approach to this task aimed to leverage the robust feature extraction capabilities of CNNs to automate and potentially improve the accuracy of animal identification over traditional methods.

Data Description

The dataset provided for this project contained folders for each of twelve animals, with each folder holding images named according to a "YYYYMMDD_HHMMSS_i.png" pattern. This naming scheme was instrumental in the preprocessing stage, as it contained the date and time information that I used to stratify the images into training and validation datasets. Specifically, I used all images collected prior to October 11, 2024, as the training set, and images from this date forward as the validation set. This temporal split ensured that the model was validated on unseen data from the same animals, simulating a real-world application where a model needs to generalize well to new data from known categories.

Methods

Model Architecture

I designed the CNN with several layers to effectively learn from the image data:

- **Input Layer**: Images were resized to 150x150 pixels as uniform input dimensions are crucial for CNNs.
- **Convolutional Layers**: I employed multiple convolutional layers with ReLU activation functions. These layers are the building blocks of CNNs, designed to capture spatial hierarchies in images through the application of learnable filters.
- **Pooling Layers**: Following each convolutional layer, I added MaxPooling layers to reduce the dimensionality of the data, thus decreasing computational complexity and overfitting.
- **Regularization**: L2 regularization was used within convolutional layers to mitigate the risk of overfitting by penalizing large weights.
- **Flattening Layer**: The Flatten layer transformed the 2D matrix data to a vector that can be fed into the fully connected dense layers.
- **Dense Layers**: I included dense layers with a large number of neurons to learn non-linear combinations of the high-level features extracted by the convolutional layers.
- **Output Layer**: The final layer used a softmax activation function to provide a probability distribution over the 12 animal classes.

Data Augmentation

To address the issue of limited data and improve the model's ability to generalize, I implemented several data augmentation techniques:

- **Rotations, translations, and flips** increase the diversity of the training data by simulating variations in animal positioning and orientation that might occur in new images.
- **Zooms** help the model learn to identify animals from varying distances.

Hyperparameter Tuning

Using the Keras Tuner, I conducted a random search to optimize the model's architecture and training parameters:

- I experimented with different numbers of convolutional layers, units, and dropout rates.
- The learning rate and L2 regularization values were varied in a logarithmic scale to find the best combination for minimizing the validation loss while maximizing accuracy.

Results

Model Training

The training process was visualized through plots showing the accuracy and loss for both the training and validation sets across epochs. As seen in the attached plots, the model accuracy on the training set consistently improved, reaching above **95%**, while the validation accuracy stabilized around **77%**. This indicates a good fit to the training data, **albeit with some signs of overfitting as evidenced by the higher validation loss** compared to the training loss.

Model Evaluation

The best model achieved a validation accuracy of 77.1%. A detailed classification report and confusion matrix provided insights into the model's performance across individual classes:

- **Precision, Recall, and F1 Score**: These metrics highlighted the strengths and weaknesses in identifying specific animals. While some classes achieved high scores, indicating reliable performance, others showed room for improvement.

Class	Precision	Recall	F1-Score
1015	0.75	1	0.86
1048	0	0	0
2003	1	1	1
2013	1	1	1
28	1	1	1
3	0.5	0.5	0.5
32	0.67	0.67	0.67
6005	1	0.5	0.67
7	0.33	0.5	0.4
7050	0.75	1	0.86
8020	1	1	1
9010	0.75	0.75	0.75
Overall	0.78	0.77	0.76

- **Confusion Matrix**: The matrix revealed specific instances of misclassifications, where the model confused between certain animals, suggesting areas for potential refinement in the model or data.

	1015	1048	2003	2013	28	3	32	6005	7	7050	8020	9010
1015	3	0	0	0	0	0	0	0	0	0	0	0
1048	1	0	0	0	0	1	0	0	0	0	0	0
2003	0	0	4	0	0	0	0	0	0	0	0	0
2013	0	0	0	2	0	0	0	0	0	0	0	0
28	0	0	0	0	2	0	0	0	0	0	0	0
3	0	0	0	0	0	1	1	0	0	0	0	0
32	0	1	0	0	0	0	2	0	0	0	0	0
6005	0	0	0	0	0	0	0	2	2	0	0	0
7	0	0	0	0	0	0	0	0	1	0	0	1
7050	0	0	0	0	0	0	0	0	0	3	0	0
8020	0	0	0	0	0	0	0	0	0	0	4	0
9010	0	0	0	0	0	0	0	0	0	1	0	3

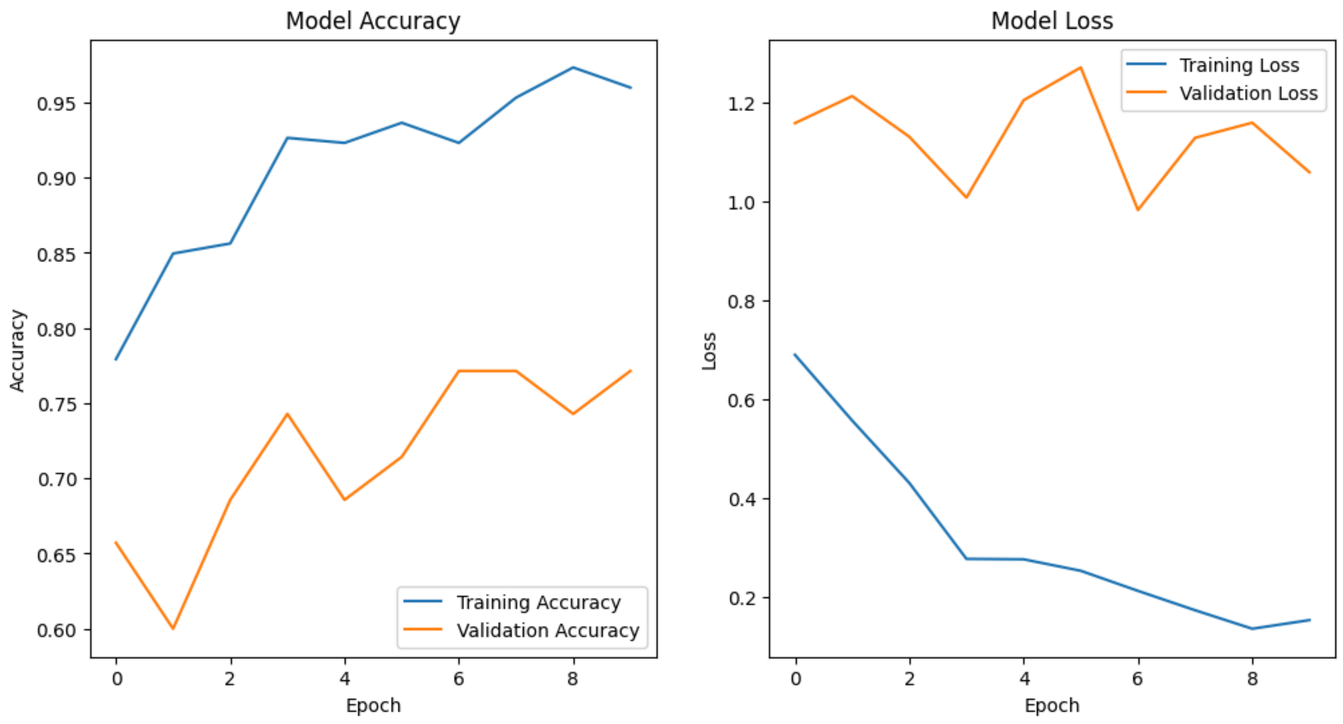
Discussion

The CNN demonstrated a promising ability to classify animal identities from images. The use of data augmentation and L2 regularization effectively contributed to the model's performance. However, the disparity between training and validation metrics suggested that further tuning of the model's architecture and training process could be beneficial. Additional approaches like ensemble methods, advanced regularization techniques, or even larger, more diverse datasets could potentially enhance the model’s robustness and accuracy.

Conclusion

This project underscored the effectiveness of convolutional neural networks in image classification tasks, particularly in the domain of animal identification. The insights gained from the methodology, challenges encountered, and the outcomes achieved offer valuable directions for future research and practical applications in automated animal monitoring systems.

Plot



Appendices

- **Code**: Detailed Python scripts and model configurations used throughout the project.