Cat-vs-Dog Classifier

A Deep Learning Approach for Binary Image Classification

Abstract

This project implements a Convolutional Neural Network (CNN) model to classify images as either a cat or a dog. Leveraging deep learning frameworks such as TensorFlow and Keras, the model achieves high accuracy and provides actionable insights. This report details the methodologies, results, and potential future enhancements for the classifier.

Introduction

Image classification is a critical task in computer vision. The Cat-vs-Dog Classifier aims to solve the binary classification problem by using CNNs to distinguish between images of cats and dogs. This classifier can serve as a foundation for real-world applications such as animal monitoring systems, pet identification apps, and educational tools.

Objectives

1. Develop a robust CNN model for binary classification.
2. Use data augmentation techniques to improve model generalization.
3. Evaluate model performance using metrics such as accuracy, precision, and recall.
4. Provide insights through visualizations like confusion matrices and classification reports.

Technologies and Tools

- **Languages**: Python
- **Libraries/Frameworks**: TensorFlow, Keras, NumPy, scikit-learn, Matplotlib, Seaborn
- **Hardware**: Nvidia GPU (optional, for faster training)

Dataset

- **Source**: Kaggle Dogs vs Cats Dataset
- **Structure**:
    - **Training Set**: 10,000 labeled images (5,000 each of cats and dogs).
    - **Testing Set**: 2,000 labeled images.
- **Preprocessing**:
    - Images resized to 256x256 pixels.
    - Normalized pixel values between 0 and 1.

Methodology

Model Architecture

1. **Convolutional Layers**:
   Extract spatial features using filters.
2. **Max Pooling Layers**:
   Downsample feature maps to reduce spatial dimensions and computational cost.
3. **Dropout Layers**:
   Prevent overfitting by randomly deactivating neurons.
4. **Fully Connected Layers**:
   Map extracted features to output probabilities.
5. **Activation Functions**:
    - ReLU for intermediate layers.
    - Sigmoid for the output layer (binary classification).

Code Implementation

- The model is implemented using Python and TensorFlow/Keras.
- Data augmentation enhances diversity by applying transformations like rotation, zoom, and horizontal flipping.
- A binary cross-entropy loss function is used, with the Adam optimizer for faster convergence.

Training

- **Epochs**: 10
- **Batch Size**: 32
- **Training/Validation Split**: 80/20
- **Data Augmentation**: Applied to the training set to improve generalization.

Results and Evaluation

Challenges Faced

1. Class imbalance: Addressed through data augmentation.
2. Overfitting: Resolved using dropout layers and early stopping techniques.
3. High computational cost: Optimized using a GPU.

Applications

1. **Animal Monitoring Systems**: Identifying animals in surveillance systems.
2. **Pet Identification Apps**: Helping pet owners recognize their pets in large datasets.
3. **Educational Purposes**: Demonstrating CNN techniques for image classification.

Future Enhancements

1. **Transfer Learning**: Incorporate pre-trained models like VGG16 or ResNet to improve accuracy.
2. **Multi-Class Classification**: Expand to classify more animal categories.
3. **Model Deployment**: Deploy the model as a web or mobile app for real-time usage.
4. **Explainability**: Use Grad-CAM to visualize model decision-making.

Conclusion

The Cat-vs-Dog Classifier demonstrates the effective use of CNNs for binary image classification. With an accuracy of 92%, the model provides reliable results, paving the way for future enhancements and practical applications in image classification.

References

1. TensorFlow Documentation: https://www.tensorflow.org
2. Kaggle Dogs vs Cats Dataset: https://www.kaggle.com/c/dogs-vs-cats
3. Deep Learning with Python by François Chollet

Appendices

- **Code**: All code is available on the GitHub repository.
- **Repository Link**: Cat-vs-Dog Classifier