Report on Convolutional Neural Network for Cat and Dog Classification

Introduction

The objective of this project is to create a Convolutional Neural Network (CNN) to classify images of cats and dogs. The dataset consists of images from two categories: cats and dogs. The implementation uses Python with the NumPy, OpenCV, Matplotlib, and TensorFlow libraries.

Data Preprocessing

Image Conversion

Images are loaded using OpenCV and converted to grayscale, as color is not considered a differentiating factor between cats and dogs in this context. The images are then resized to a specified dimension (dim).

Training Data Creation

The training data is created by iterating through the categories (cats and dogs) and loading each image, converting it to grayscale and resizing it. The training data is a list where each element is a tuple containing the resized image array and the corresponding label (0 for cats, 1 for dogs).

Data Visualization

Matplotlib is used to visualize two images from the training data, one representing a cat and the other a dog. The images are displayed in grayscale for better clarity.

Data shuffling

To prevent the model from learning any order or sequence in the data, the training data is randomly shuffled.

Data Normalization

The pixel values of the images are normalized by dividing them by 255.0 to ensure that the model converges faster during training.

Model Architecture

The CNN model is built using TensorFlow's Keras API. The architecture consists of three convolutional layers with ReLU activation functions, two max-pooling layers to reduce image size, and a dropout layer to prevent overfitting. The model's input shape is determined based on the dimensions of the training data. The output layer consists of one neuron, representing the binary classification of cats and dogs.

Convolutional Layers

Conv2D layer with 64 filters and a 3x3 kernel.

MaxPooling2D layer with a 2x2 pool size.

Dropout layer with a dropout rate of 0.3.

Conv2D layer with 128 filters and a 3x3 kernel.

MaxPooling2D layer with a 2x2 pool size.

Fully Connected Layers

The model is followed by two dense (fully connected) layers. The final layer uses the softmax activation function for binary classification.

Model Compilation

The model is compiled using the Adam optimizer, sparse categorical crossentropy loss function, and accuracy as the metric. This configuration is suitable for binary classification problems.

Conclusion

This report provides an overview of the implementation of a CNN for classifying cat and dog images. The model architecture is designed to capture relevant features from the images, and the training data is appropriately preprocessed for effective learning. The next steps would involve training the model, evaluating its performance on a test set, and fine-tuning hyperparameters for optimal results.`