Github Link : https://github.com/nayoo1119/CECS_456_Final_Project

1. Introduction

Image classification has become one of the most impactful applications of deep learning, finding use in domains such as medical imaging, autonomous driving, and content moderation. Convolutional Neural Networks (CNNs) stand out as one of the most effective tools for image classification due to their ability to capture spatial hierarchies in data. This project focuses on classifying images into eight distinct categories using CNN. The dataset, sourced from Kaggle, comprises diverse images representing objects like airplanes, cars, animals, and more. The primary objective is to design and train a CNN model that achieves high accuracy while demonstrating robustness across diverse image categories.

2. Dataset and Related Work

Dataset : The Natural Images Dataset consists of 13,798 images categorized into 8 classes:

- Categories: Airplane, Car, Cat, Dog, Flower, Fruit, Motorbike, Person.

Preprocessing:

- All images were resized to 128x128 pixels for uniformity. Pixel values were normalized to the [0, 1] range to accelerate training.

Related Work : Deep learning models like AlexNet, VGG, and ResNet have become benchmarks in image classification by demonstrating the power of convolutional layers to learn patterns in images.

- AlexNet was one of the first models to show the potential of deep learning, winning a major image recognition competition in 2012 with its ability to extract meaningful features.
- VGG followed with an even deeper network, using smaller filters to improve accuracy while maintaining simplicity.
- ResNet introduced shortcut connections, which made it possible to train very deep networks without performance issues like vanishing gradients.

These models have shown that convolutional layers are excellent at capturing spatial and hierarchical features in images. Inspired by these advanced architectures, this project uses a simpler CNN design to achieve effective classification while remaining efficient. The goal is to balance good performance with manageable computational requirements, making it practical for smaller-scale projects.

3. Methodology

The Convolutional Neural Network (CNN) architecture designed for this project is structured to classify images into 8 categories effectively. The following components and configurations were used:

- Input Layer: The input layer accepts RGB images resized to dimensions of (128, 128, 3), ensuring uniformity across the dataset and compatibility with the model.

- Convolutional Layers:

(1) Three convolutional layers were employed to extract features from the images.

(2) 1st Layer: This layer uses 64 filters with a kernel size of 7x7 and applies the ReLU (Rectified Linear Unit) activation function. The larger kernel size helps capture more spatial features in the initial layers.
(3) 2nd and 3rd Layers: These layers use 128 filters with a kernel size of 3x3, along with the ReLU activation function. The increased filter count helps the model learn more complex features.

- Pooling Layers: MaxPooling with a 2x2 kernel was applied after each convolutional layer. Pooling reduces the spatial dimensions of the feature maps, which helps decrease computational requirements while retaining the most important features. This also prevents overfitting by acting as a form of down-sampling.
- Flatten Layer: After the convolutional and pooling layers, the 2D feature maps are flattened into a 1D vector. This transformation prepares the data for input into the fully connected layers.
- Fully Connected Layers:
  (1) A dense layer with 128 neurons and ReLU activation is included to learn the relationships between high-level features.
  (2) The output layer contains 8 neurons with a softmax activation function, which is designed for multi-class classification. Each neuron corresponds to one class and outputs a probability for that class.

The model was compiled with the following settings:

(1) Loss Function: Categorical Crossentropy was used, which is well-suited for multi-class classification problems as it measures the difference between predicted probabilities and actual labels.
(2) Optimizer: The Adam optimizer was chosen for its ability to adapt the learning rate dynamically during training, making it effective for both large and small datasets.
(3) Metric: Accuracy was used as the primary metric to track the model's classification performance during training and validation.

This architecture was carefully designed to balance simplicity and performance, ensuring that the model can effectively learn from the dataset while being computationally efficient.

4. Experimental Setup

The project was implemented in Google Colab, leveraging GPU acceleration for efficient model training. The dataset was loaded and processed using TensorFlow's ImageDataGenerator, which handled data augmentation and rescaling. The training data was split into:

- 80% Training Set and 20% Validation Set

Training configurations:

- Batch Size: 32 , Epochs: 10 and Validation Split: 20%

Each training epoch took approximately 2 minutes, with consistent performance improvements observed.

5. Measurement

Metrics : The model's performance was evaluated using:

- Accuracy: Reflects the proportion of correctly classified images.
- Loss: Measures the deviation of predictions from actual labels.

Observations

During training and validation, both accuracy and loss metrics were tracked. The final results demonstrated minimal overfitting, indicating the model's ability to generalize across unseen data.

6. Results Analysis, Intuitions, and Comparison

Results

- Validation Accuracy: 99.29% and Validation Loss: 0.0218

Analysis

The high validation accuracy highlights the effectiveness of the CNN in distinguishing features across classes. The low validation loss signifies robust learning and minimal errors during prediction.

Intuitions

- The model's performance can be attributed to the combination of convolutional and pooling layers, which efficiently captured image features.
- The balanced dataset with diverse images contributed to effective learning.

Comparison

While the architecture is simpler than models like VGG and ResNet, it achieves comparable results for this dataset. The success demonstrates that with a well-prepared dataset, even basic architectures can achieve outstanding performance.

7. Conclusion

This project successfully implemented a CNN for multi-class image classification, achieving a validation accuracy of 99.29%. The results underline the importance of dataset preprocessing, architectural design, and appropriate hyperparameter tuning. Future work could explore deeper architectures or data augmentation techniques to enhance performance.

8. Contributions

This project was completed as an individual effort by Nay Thuya Oo.