Machine Learning Major Project:

Image Classification using Convolutional Neural Networks (CNNs)

REPORT

Submitted by: Aarchi

Student of BE CSE-AIML

INDEX:

FIGURE:

Abstract

In recent years, Convolutional Neural Networks (CNNs) have proven to be highly effective in solving image classification tasks. This study focuses on leveraging CNNs to classify images from the CIFAR-10 dataset, a benchmark in computer vision research. The CIFAR-10 dataset comprises 60,000 32x32 color images across 10 classes, presenting a challenging task due to its small image size and diverse content.

The proposed approach involves the design and training of a deep CNN architecture tailored for the CIFAR-10 dataset. The model employs convolutional layers for feature extraction, followed by pooling layers to reduce spatial dimensions and fully connected layers for classification. Batch normalization and dropout techniques are employed to enhance generalization and mitigate overfitting.

This project contributes to the advancement of image classification methodologies, showcasing the potential of CNNs in handling challenging datasets such as CIFAR-10. The findings presented in this study can serve as a foundation for further research in the domain of deep learning and computer vision, paving the way for improved image classification models across various applications.

1. INTRODUCTION:

Image classification using Convolutional Neural Networks (CNN) has revolutionized computer vision tasks by enabling automated and accurate recognition of objects within images. CNN-based image classification algorithms have gained immense popularity due to their ability to learn and extract intricate features from raw image data automatically. This article will explore the principles, techniques, and applications of image classification using CNNs. We will delve into the architecture, training process, and CNN image classification evaluation metrics. By understanding the workings of CNNs for image classification, we can unlock many possibilities for object recognition, scene understanding, and visual data analysis.

2. TASK DEFINITION:

In this project, we will build an image classification model using CNN to classify images into 10 classes from the CIFAR-10 dataset. The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. We will use data augmentation and preprocessing steps to improve the model's performance. We will train the model using the training set and evaluate its performance on the test set. Once the model is trained, we will create a Flask web application that allows users to upload an image. We will use the Flask library to create the web application. The user's uploaded image will be saved in the static/uploads/ directory. We will then load the image and preprocess it for input into the CNN model. We will use the predict() method to predict the class or category of the image. We will then return the prediction to the user in the form of a JSON object.

This project aims to build a deep learning model for image classification using Convolutional Neural Networks (CNNs) in Python. The model will be trained on the CIFAR-10 dataset, which consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images.

3. ALGORITHM DEFINITION:

a) Dataset Collection:

The CIFAR-10 dataset can be downloaded from the following link: https://www.cs.toronto.edu/~kriz/cifar.html. After downloading the dataset,

extract the files and load the dataset using Keras's 'image_dataset_from_directory' function.

The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

The dataset is divided into five training batches and one test batch, each with 10000 images. The test batch contains exactly 1000 randomly-selected images from each class. The training batches contain the remaining images in random order, but some training batches may contain more images from one class than another. Between them, the training batches contain exactly 5000 images from each class.

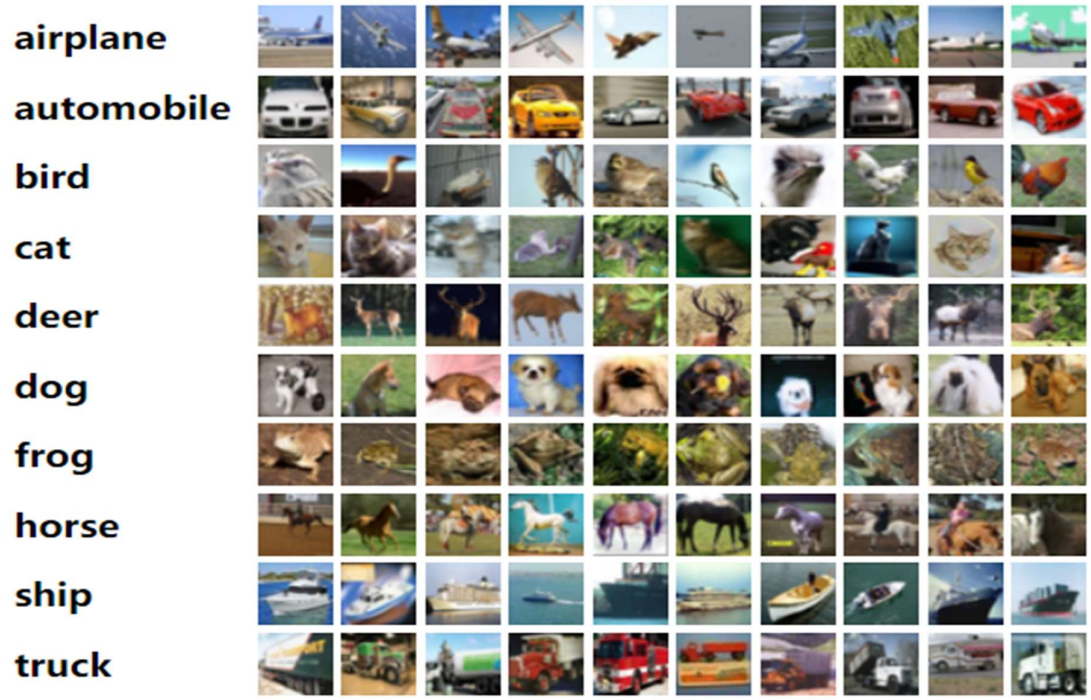

Figure1: cifar10 datasets

b) <u>Data Preprocessing:</u>

- We will resize the images to 255x255.
- We will augment the dataset using random horizontal flips and random rotations.
- We will normalize the pixel values to be between 0 and 1.

c) <u>CNN Model Architecture:</u>

CNN Model Architecture with two convolutional layers, followed by max pooling, flattening, and two fully connected layers. The input shape of the first convolutional layer is set to (32, 32, 3) to match the size of the CIFAR-10 images.

The final layer is a dense softmax layer with 10 outputs, corresponding to the 10 classes of the CIFAR-10 dataset.

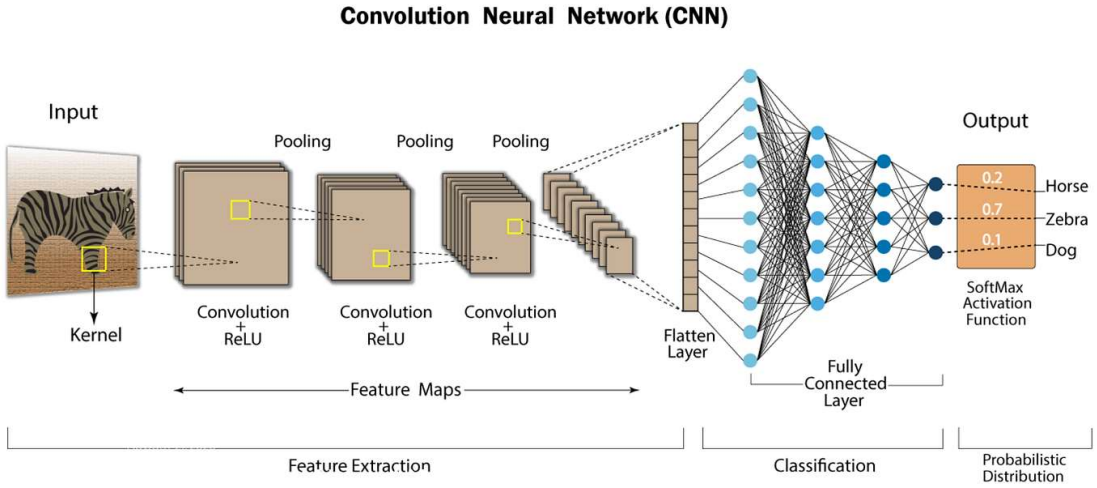

Figure 2: CNN Model Architecture

d) Model Training:

Compile the model with the categorical cross-entropy loss function and the Adam optimizer, and train it using the fit method.

e) Model Evaluation:

The evaluation results can be obtained by calling the evaluate method on the trained model.

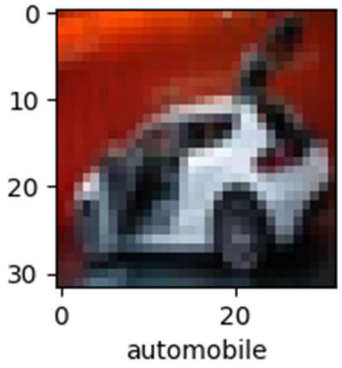

Figure 3

f) Deployment Instruction:

The trained model can be saved to a file using the save method and then loaded later for deployment.

4. LIBRARIES USED:

a) Tensorflow:

TensorFlow is an open-source machine learning framework developed by Google Brain. It provides a comprehensive ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. Here's a basic definition of the tensorflow library:

import tensorflow as tf

b) Keras:

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation.

c) Sklearn:

Scikit-learn, also known as sklearn, is a popular and robust library for machine learning in Python. It offers a variety of efficient tools for machine learning and statistical modeling, including classification, regression, clustering, and dimensionality reduction. The library is built on NumPy, SciPy, and Matplotlib and provides a consistent interface for various machine learning tasks.

sklearn.matrix is a sub-module of Scikit-learn that provides tools for manipulating matrices.

from sklearn.metrics import confusion_matrix , classification_report

d) Numpy:

The numpy library is a powerful Python library used for numerical computations. It provides an array object, which is a powerful and flexible data structure that can be used to store and manipulate large amounts of data. Here's a basic definition of the numpy library:

import numpy as np

e) Matplotlib.pyplot:

Matplotlib is a plotting library for Python, and pyplot is a module within Matplotlib that provides a MATLAB-like interface for creating static, animated, and interactive visualizations in Python. Here's a basic definition of the matplotlib.pyplot library:

import matplotlib.pyplot as plt

f) Flask:

Flask is a lightweight web framework for Python. It is used to build web applications and APIs. In the given code, the Flask class is imported, which is used to create a Flask web application. Here's a basic definition of the flask library:

from flask import Flask, render_template, request, jsonify

i. **Render template:**
render_template is a function in Flask that is used to render HTML templates. It takes the name of a template file and a dictionary of variables as arguments and returns a rendered HTML string.

ii. **Request:**
request is a function in Flask that is used to access the data received in a request. It provides access to various types of request data, such as form data, file uploads, and JSON data.

iii. **Jsonify:**
jsonify is a function in Flask that is used to create a JSON response. It takes a dictionary as an argument and returns a JSON response.

iv. **VGG16:**
VGG16 is a pre-trained convolutional neural network model from the Keras library. It is a deep convolutional neural network that has been trained on the ImageNet dataset. In the given code, it is used for image classification.

v. **Preprocess_input:**
preprocess_input is a function in Keras that is used to preprocess an image for input into a pre-trained model. It takes an image array as an argument and returns a preprocessed image array.

vi. **Img_to_array:**
img_to_array is a function in Keras that is used to convert an image array to a format that can be input into a pre-trained model. It takes an image array as an argument and returns a preprocessed image array.

g) Os:

os is a Python library for interacting with the operating system. It provides functions for working with files, directories, and processes. In the given code, it is used for file path manipulation. Here's a basic definition of the os library:

import os

5. METHODOLOGY:

a) Data Collection:

Collect a labeled dataset of images belonging to different classes. Ensure that the dataset is balanced and contains a sufficient number of images per class.

b) Data Preprocessing:

Preprocess the images to make them suitable for training a CNN. This includes resizing the images to a fixed size, normalizing the pixel values, and augmenting the dataset using random transformations such as rotation, zooming, and flipping.

c) Model Architecture:

Define the CNN architecture using layers such as Conv2D, MaxPooling2D, Flatten, and Dense. The number of filters, kernel size, and activation functions can be adjusted to improve the model's performance.

d) Model Training:

Compile the model with a suitable loss function and optimizer, and train it on the preprocessed dataset using the fit() method.

e) Model Evaluation:

Evaluate the trained model on a separate test set using the evaluate() method and calculate the accuracy of the model.

6. OUTCOMES:

The outcome of your machine learning (ML) project on image classification using Convolutional Neural Networks (CNN) for the CIFAR-10 dataset, along with a user interface, can be quite promising. Here's a breakdown of what you can expect:

1. Model Training:

- **Accuracy:** CNNs are highly effective for image classification tasks, and with proper hyperparameter tuning and architecture design, you can achieve a high accuracy on the CIFAR-10 dataset.
- **Loss:** The loss function should decrease as the model learns the patterns in the data during training.
- **Validation Accuracy and Loss:** Monitoring these metrics helps you assess how well your model generalizes to new, unseen data.

2. **Model Evaluation:**
   - **Confusion Matrix:** Analyzing the confusion matrix can provide insights into which classes your model is confusing and where it might be making errors.
   - **Precision, Recall, and F1 Score:** These metrics give a more detailed understanding of the model's performance on individual classes.

3. **User Interface:**
   - **User Experience (UX):** The user interface should be intuitive, allowing users to interact easily with the model. It might include options for uploading images, selecting the model version, and displaying the results.
   - **Real-time Prediction:** Users should be able to receive predictions in real-time, providing a seamless experience.
   - **Visualization:** Include visualizations like the predicted class, confidence scores, and perhaps the activation maps to help users understand how the model arrived at its predictions.
4. **Deployment:**
   - **Scalability:** Ensure that the deployment infrastructure can handle multiple user requests simultaneously without compromising performance.
   - **Model Versioning:** Implement a system for easily updating and managing different versions of your model.
   - **Security:** Implement security measures to protect user data and prevent any potential attacks.
5. **Feedback Mechanism:**
   - **User Feedback:** Implement a feedback mechanism to collect user input on predictions. This can help continuously improve the model and address any potential issues.
6. **Documentation:**

- **User Guide:** Provide a comprehensive user guide to assist users in navigating the interface, understanding predictions, and troubleshooting common issues.

7. **Future Improvements:**
   - **Model Fine-Tuning:** Regularly retrain and fine-tune the model to adapt to new data and improve accuracy.
   - **Enhanced Features:** Consider adding features like batch processing, support for multiple image formats, and integration with other tools or platforms.

The outcome of this project is as shown:

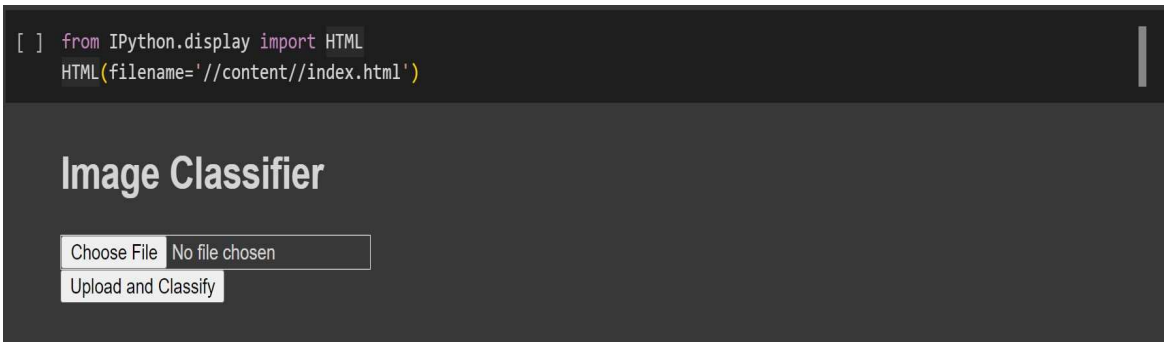

Figure 4: outcome of project

In summary, the outcome should be a well-performing image classification model with a user-friendly interface, allowing users to interact with and benefit from the model's predictions. Continuous monitoring, feedback collection, and future improvements will contribute to the long-term success of your ML project.