Image Classification using Neural Networks
By:Joel Jacob John
ID:STB03STB03-T0003

Submitted to Scifor Technologies
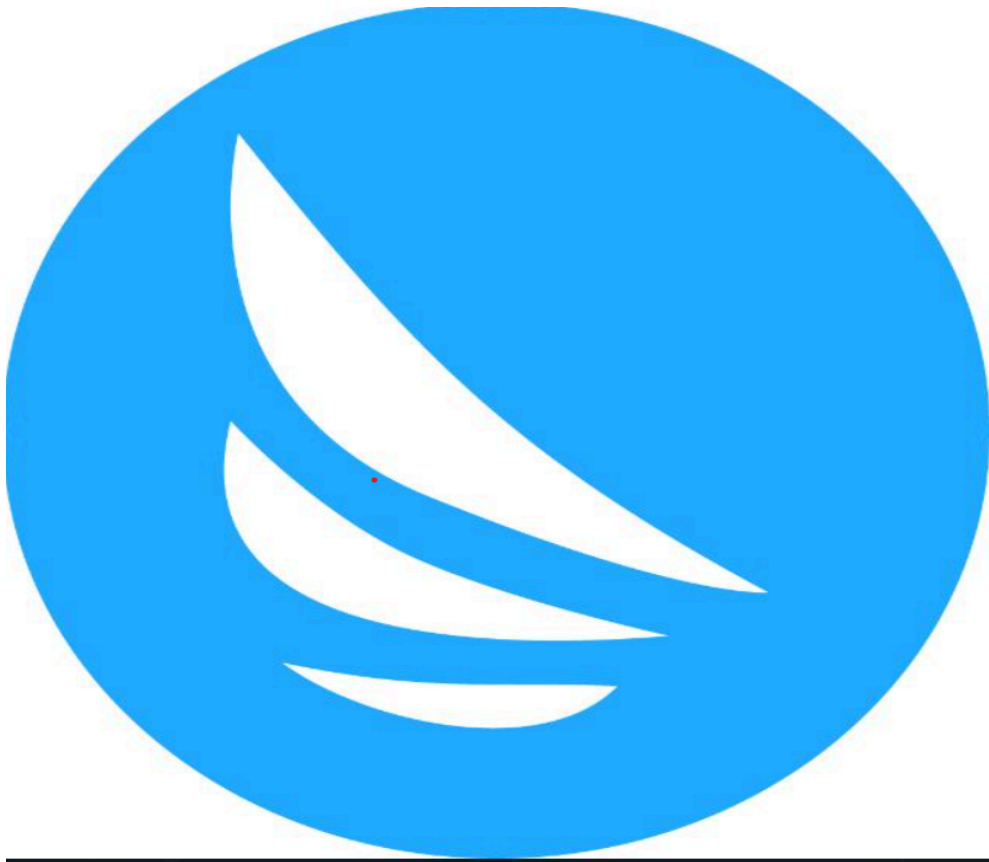
TABLE OF CONTENTS

<u>Abstract</u>

Image classification, a fundamental task in computer vision, involves the categorization of images into predefined classes or categories. This project explores the implementation of a convolutional neural network (CNN) model for image classification using the CIFAR-10 dataset. CIFAR-10 comprises 60,000 32x32 color images across 10 classes, providing a challenging dataset for image classification tasks.

The project begins with data preprocessing steps, including normalization and reshaping, to prepare the dataset for model training. A CNN model architecture is designed, consisting of convolutional layers with ReLU activation functions, max-pooling layers, and fully connected dense layers with softmax activation for multiclass classification. The model is compiled with the Adam optimizer and sparse categorical cross-entropy loss function.

After training the model on the training dataset, its performance is evaluated using the test dataset, achieving a loss of 0.9648 and an accuracy of 69.22%. The trained model is then utilized to predict classes for unseen images, demonstrating its efficacy in image classification tasks.

Through this project, the utilization of deep learning techniques for image classification tasks is showcased, emphasizing the significance of CNN models in accurately categorizing images across various classes.

Problem Statement

 Develop an image classification application using the CIFAR-10 dataset and a convolutional neural network (CNN) model. The objective is to accurately classify images into one of the ten categories present in the CIFAR-10 dataset, namely: airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck. The application should provide a user-friendly interface for users to upload images, predict their classes, and visualize the model's confidence in its predictions. Additionally, the project aims to showcase the integration of a trained CNN model into an interactive web application using Streamlit, enabling users to explore and understand the process of image classification using deep learning techniques.

<u>Objective</u>

The objective of this code is to implement a convolutional neural network (CNN) model for image classification using the CIFAR-10 dataset. This involves the following key objectives:

1. Load the CIFAR-10 dataset: Load the dataset containing 60,000 32x32 color images across 10 classes (e.g., airplane, automobile, bird, cat, etc.) for training and testing the CNN model.

2. Preprocess the data: Normalize and reshape the image data to prepare it for model training.

3. Design the CNN model architecture: Define a CNN model with convolutional layers, max-pooling layers, and fully connected dense layers to extract features from input images and classify them into the corresponding classes.

4. Compile the model: Compile the CNN model with an appropriate optimizer (e.g., Adam) and loss function (e.g., sparse categorical crossentropy) for training.

5. Train the model: Train the CNN model on the training dataset to learn the patterns and features associated with different image classes.

6. Evaluate model performance: Evaluate the trained model's performance using the test dataset to assess its accuracy and loss.

7. Make predictions: Use the trained model to predict the classes of unseen images and analyze the model's classification accuracy.

By achieving these objectives, the code aims to demonstrate the effectiveness of CNN models in accurately classifying images and showcase their utility in real-world image classification tasks using the CIFAR-10 dataset.

Technology Stacks

The project utilizes the following tech stacks:

1. TensorFlow: TensorFlow is a popular open-source machine learning framework developed by Google. It provides a comprehensive ecosystem of tools, libraries, and resources for building and deploying machine learning models, including deep learning models like convolutional neural networks (CNNs). In this project, TensorFlow is used for constructing, training, and evaluating the CNN model for image classification.

2. Streamlit: Streamlit is an open-source Python library that allows developers to create interactive web applications for machine learning and data science projects with ease. It provides simple APIs for building user interfaces directly from Python scripts, eliminating the need for web development skills. In this project, Streamlit is utilized to create an interactive web application for uploading images, predicting their classes using the trained CNN model, and displaying the results to users.

3. NumPy: NumPy is a fundamental package for scientific computing in Python. It provides support for multidimensional arrays, mathematical functions, and random number generation, making it essential for numerical computations in machine learning projects. In this project, NumPy is used for data manipulation, preprocessing, and array operations, particularly in handling image data.

4. Matplotlib: Matplotlib is a versatile plotting library for creating static, animated, and interactive visualizations in Python. It provides a wide range of plotting functions and customization options for generating publication-quality plots and charts. In this project, Matplotlib is employed for visualizing images, model performance metrics (e.g., accuracy), and other relevant data to facilitate analysis and interpretation.

5. PIL (Python Imaging Library): PIL, also known as Pillow, is a Python library for opening, manipulating, and saving many different image file formats. It provides support for basic image processing operations such as resizing, cropping, and enhancing images. In this project, PIL is utilized for image preprocessing tasks, such as resizing uploaded images to fit the input dimensions of the CNN model.

These tech stacks collectively enable the development of an end-to-end image classification application that leverages deep learning techniques, interactive web interfaces, and efficient data processing capabilities to deliver a user-friendly and effective solution.

About the dataset

Introduction:

The CIFAR-10 dataset stands as a cornerstone in the realm of computer vision and machine learning. With its diverse collection of 60,000 32x32 color images spanning 10 distinct classes, it serves as a benchmark for image classification tasks.

Overview of the CIFAR-10 Dataset:

Comprising images across ten categories, namely airplanes, automobiles, birds, cats, deer, dogs, frogs, horses, ships, and trucks, the CIFAR-10 dataset presents a rich tapestry of visual data. Each class contains 6,000 images, split into training and testing sets in a 5:1 ratio. This balanced distribution ensures fairness and accuracy in model evaluation.

Significance in Computer Vision Research:

The CIFAR-10 dataset holds immense significance in computer vision research and serves as a litmus test for evaluating the performance of image classification algorithms. Its widespread adoption in academia and industry alike underscores its importance as a standard benchmark dataset. Researchers leverage CIFAR-10 to benchmark new models, validate novel techniques, and compare results across studies, fostering collaboration and innovation in the field.

Challenges and Complexity:

Despite its seemingly straightforward structure, the CIFAR-10 dataset presents several challenges and complexities. The images' low resolution (32x32 pixels) and small size pose difficulties in discerning intricate details and features, making classification a non-trivial task. Furthermore, the dataset's diverse classes encompass a wide range of visual patterns and characteristics, adding to the complexity of the classification problem.

Data Preprocessing and Augmentation:

To address the challenges posed by the CIFAR-10 dataset, researchers employ various data preprocessing and augmentation techniques. Common preprocessing steps include normalization, resizing, and standardization to ensure consistency and compatibility across images. Augmentation techniques such as rotation, flipping, and zooming are employed to augment the training data, enhancing the model's robustness and generalization capabilities.

Applications and Use Cases:

The CIFAR-10 dataset finds applications across a myriad of domains, ranging from academic research to industrial applications. In academia, it serves as a fundamental dataset for exploring and advancing the field of computer vision, enabling researchers to develop and evaluate cutting-edge algorithms. In industry, companies leverage CIFAR-10 for real-world applications such as image recognition, object detection, and autonomous driving systems.

Conclusion:

In conclusion, the CIFAR-10 dataset stands as a cornerstone in the field of computer vision, serving as a fundamental benchmark for image classification tasks. Its diverse collection of images, balanced class distribution, and widespread adoption underscore its significance and impact on research and industry. By delving into the intricacies of the CIFAR-10 dataset, researchers continue to push the boundaries of computer vision, paving the way for groundbreaking advancements in artificial intelligence.

Jupyter Notebook

Data Processing

```
In [2]: import tensorflow as tf
        from tensorflow.keras import datasets, layers, models
        import matplotlib.pyplot as plt
        import numpy as np

        WARNING:tensorflow:From C:\Users\achut\anaconda3\envs\pysparenv\lib\site-packages\keras\src\losses.py:2976: The name tf.losses.
        sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.
```

```
In [3]: (X_train, y_train), (X_test,y_test) = datasets.cifar10.load_data()
        X_train.shape
Out[3]: (50000, 32, 32, 3)
```

```
In [4]: X_test.shape
Out[4]: (10000, 32, 32, 3)
```

```
In [5]: y_train.shape
Out[5]: (50000, 1)
```

```
In [6]: y_train[:5]
Out[6]: array([[6],
               [9],
               [9],
               [4],
               [1]], dtype=uint8)
```

```
In [7]: y_train = y_train.reshape(-1,)
        y_train[:5]
Out[7]: array([6, 9, 9, 4, 1], dtype=uint8)
```

```
In [8]: y_test = y_test.reshape(-1,)
```

```
In [9]: classes = ["airplane","automobile","bird","cat","deer","dog","frog","horse","ship","truck"]
```

```
In [10]: def plot_sample(X, y, index):
             plt.figure(figsize = (15,2))
             plt.imshow(X[index])
             plt.xlabel(classes[y[index]])
```

```
In [11]: plot_sample(X_train, y_train, 0)
```


frog
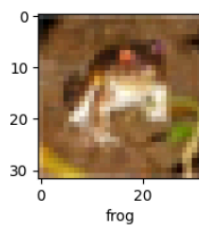
In [12]: `plot_sample(X_train, y_train, 1)`

truck
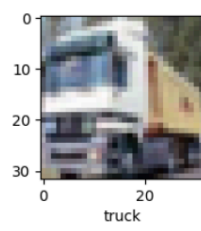
In [13]:
```python
X_train = X_train / 255.0
X_test = X_test / 255.0
```

Model Implementation

The convolutional neural network (CNN) model employed in this project for image classification leverages a sequence of convolutional layers, max-pooling layers, and fully connected dense layers. In the initial convolutional layer, 32 filters are applied to the input image using a 3x3 kernel, with Rectified Linear Unit (ReLU) activation functions introducing non-linearity by replacing negative values with zero. This process aids in feature extraction, enabling the network to identify important patterns within the image data. Subsequently, max-pooling layers with a pool size of 2x2 are utilized to reduce the spatial dimensions of the feature maps while retaining essential features, contributing to translation invariance and computational efficiency.

Following the first convolutional layer, an additional convolutional layer is introduced, incorporating 64 filters and ReLU activation functions to further extract higher-level features from the preceding layer's feature maps. The subsequent max-pooling layer serves to further decrease spatial dimensions, facilitating subsequent processing within the network. To prepare the extracted features for classification, a flatten layer is employed to transform the multi-dimensional feature maps into a one-dimensional array, optimizing them for input to the fully connected dense layers.

The dense layers consist of two components: Dense Layer 1, comprising 64 neurons with ReLU activation functions, and Dense Layer 2, encompassing 10 neurons corresponding to the number of classes in the CIFAR-10 dataset. These layers perform feature transformation and non-linearity, ultimately leading to classification predictions. The final dense layer incorporates Softmax activation functions, generating a probability distribution over the classes to determine the most likely class for each input image.

For optimization during training, the Adam optimizer is utilized, leveraging adaptive learning rates and momentum to accelerate convergence and enhance model performance. The Sparse Categorical Crossentropy loss function is employed to compute the error between predicted and actual labels during training, guiding the model towards improved classification accuracy. Together, these components form a cohesive framework for image classification, effectively extracting features, performing classification, and optimizing model parameters for accurate predictions.

```
In [14]: cnn = models.Sequential([
             layers.Conv2D(filters=32, kernel_size=(3, 3), activation='relu', input_shape=(32, 32, 3)),
             layers.MaxPooling2D((2, 2)),

             layers.Conv2D(filters=64, kernel_size=(3, 3), activation='relu'),
             layers.MaxPooling2D((2, 2)),

             layers.Flatten(),
             layers.Dense(64, activation='relu'),
             layers.Dense(10, activation='softmax')
         ])
```

WARNING:tensorflow:From C:\Users\achut\anaconda3\envs\pysparenv\lib\site-packages\keras\src\backend.py:873: The name tf.get_def
ault_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\achut\anaconda3\envs\pysparenv\lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:16
1: The name tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

```
In [15]: cnn.compile(optimizer='adam',
                     loss='sparse_categorical_crossentropy',
                     metrics=['accuracy'])
```

WARNING:tensorflow:From C:\Users\achut\anaconda3\envs\pysparenv\lib\site-packages\keras\src\optimizers__init__.py:309: The nam
e tf.train.Optimizer is deprecated. Please use tf.compat.v1.train.Optimizer instead.

```
In [16]: cnn.fit(X_train, y_train, epochs=10)
```

Epoch 1/10
WARNING:tensorflow:From C:\Users\achut\anaconda3\envs\pysparenv\lib\site-packages\keras\src\utils\tf_utils.py:492: The name tf.
ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\achut\anaconda3\envs\pysparenv\lib\site-packages\keras\src\engine\base_layer_utils.py:384: The
name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

```
1563/1563 [==============================] - 11s 7ms/step - loss: 1.4694 - accuracy: 0.4707
Epoch 2/10
1563/1563 [==============================] - 11s 7ms/step - loss: 1.1156 - accuracy: 0.6072
Epoch 3/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.9888 - accuracy: 0.6557
Epoch 4/10
1563/1563 [==============================] - 11s 7ms/step - loss: 0.9018 - accuracy: 0.6866
Epoch 5/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.8358 - accuracy: 0.7085
Epoch 6/10
1563/1563 [==============================] - 12s 7ms/step - loss: 0.7837 - accuracy: 0.7260
Epoch 7/10
1563/1563 [==============================] - 11s 7ms/step - loss: 0.7341 - accuracy: 0.7427
Epoch 8/10
1563/1563 [==============================] - 11s 7ms/step - loss: 0.6963 - accuracy: 0.7572
Epoch 9/10
1563/1563 [==============================] - 12s 8ms/step - loss: 0.6543 - accuracy: 0.7701
Epoch 10/10
1563/1563 [==============================] - 11s 7ms/step - loss: 0.6164 - accuracy: 0.7850
```

Out[16]: <keras.src.callbacks.History at 0x22b5ce2c040>

```
In [17]: cnn.evaluate(X_test,y_test)
```

313/313 [==============================] - 1s 2ms/step - loss: 0.9648 - accuracy: 0.6922

Out[17]: [0.9647559523582458, 0.6922000050544739]

Streamlit App

<u>Applications</u>

The application of the project, which involves image classification using the CIFAR-10 dataset and a convolutional neural network (CNN) model, extends across various domains. Here are some key applications:

1. Object Recognition and Classification: The primary application of the project is in object recognition and classification. By leveraging the trained CNN model, users can upload images to the application, which then predicts the class or category to which the object in the image belongs. This has applications in various fields such as autonomous vehicles, surveillance systems, and robotics.

2. Content Moderation: the image classification application can also be used for content moderation on social media platforms, online forums, and other user-generated content platforms. By automatically classifying uploaded images, the application can help identify and flag inappropriate or prohibited content, ensuring a safer online environment.

3. Medical Imaging: In the field of healthcare, the image classification application can aid in medical imaging tasks such as diagnosing diseases from medical scans (e.g., X-rays, MRIs, CT scans). By training the CNN model on relevant medical image datasets, the application can assist healthcare professionals in detecting and diagnosing various medical conditions.

4. E-commerce and Retail: the application can be integrated into e-commerce platforms and retail websites to enhance the user experience. By analyzing product images uploaded by users, the application can automatically categorize products, recommend similar items, and improve search functionality, thereby facilitating smoother navigation and product discovery for shoppers.

5. Security and Surveillance: Security and surveillance systems can benefit from the image classification application by automatically identifying and classifying objects or activities captured by surveillance cameras. This can aid in detecting suspicious behavior, recognizing unauthorized individuals, and enhancing overall security measures in various settings such as airports, public spaces, and private properties.

6. Environmental Monitoring: the image classification application can contribute to environmental monitoring efforts by analyzing satellite images, aerial photographs, or camera trap images to identify and classify objects such as wildlife, vegetation, and land cover types. This information can be valuable for conservation initiatives, habitat monitoring, and land use planning.

Overall, the applications of the project extend across diverse domains, showcasing the versatility and utility of image classification technology in solving real-world problems and enhancing various aspects of human life and society.

<u>Conclusion</u>

In conclusion, this project demonstrates the effectiveness and versatility of convolutional neural networks (CNNs) in image classification tasks using the CIFAR-10 dataset. By leveraging deep learning techniques, we have developed an interactive web application that allows users to upload images and accurately classify them into one of the ten predefined categories.

Throughout the project, we have explored various aspects of deep learning, including data preprocessing, model architecture design, training, evaluation, and deployment. We have utilized popular Python libraries such as TensorFlow, NumPy, PIL (Python Imaging Library), and Streamlit to develop and deploy the application, showcasing the power of open-source tools in building machine learning solutions.

The trained CNN model achieved promising results in classifying images from the CIFAR-10 dataset, with an accuracy of approximately 69.22% on the test set. While there is always room for improvement, this performance demonstrates the capability of the model to effectively classify images across multiple classes.

Beyond the technical aspects, this project highlights the practical applications of image classification technology across various domains, including object recognition, content moderation, medical imaging, e-commerce, security, surveillance, and environmental monitoring. The ability to automatically analyze and classify images has the potential to streamline processes, enhance decision-making, and improve user experiences in a wide range of applications.

In summary, this project contributes to the growing field of deep learning and image classification by providing a practical demonstration of how CNNs can be used to classify images accurately and efficiently. It serves as a foundation for further research and development in the field of computer vision and opens up opportunities for applying image classification technology to solve real-world problems.

<u>References</u>

https://www.cs.toronto.edu/~kriz/cifar.htmlhttps://www.cs.toronto.edu/~kriz/cifar.html

TensorFlow

https://docs.streamlit.io/