**IMAGE CLASSIFICATION**

**Author-Siddhartha Dey Chetia**

**Manipal University Jaipur**

**Dept of Information Technology**

**Jaipur, India**

**Email address-** [**siddhartha.219309146@muj.manipal.edu**](mailto:siddhartha.219309146@muj.manipal.edu)

**ABSTRACT**

In the realm of computer vision, image classification stands as a fundamental task with far-reaching applications. This paper delves into the realm of image classification using the CIFAR-10 dataset, employing the powerful and flexible deep learning framework, Keras. The CIFAR-10 dataset, comprising 60,000 32x32 color images across 10 distinct classes, serves as an ideal benchmark for evaluating the performance of image classification models.

The study begins by highlighting the significance of image classification in various domains, including medical diagnostics, autonomous vehicles, and facial recognition. Acknowledging the complexity of this task, the paper introduces the CIFAR-10 dataset, emphasizing its diversity and real-world relevance.

Keras, a high-level neural networks API, is chosen for its ease of use and seamless integration with TensorFlow, making it an ideal choice for both beginners and seasoned deep learning practitioners. The implementation involves constructing a convolutional neural network (CNN), a class of deep learning models renowned for their success in image-related tasks.

The architecture of the CNN is detailed, emphasizing the layers responsible for feature extraction and classification. The model is trained using the CIFAR-10 dataset, and the paper explores the crucial aspects of the training process, including data pre-processing, model compilation, and hyperparameter tuning.

The evaluation section discusses performance metrics such as accuracy, precision, recall, and F1 score, providing a comprehensive analysis of the model's efficacy. The results showcase the model's ability to generalize across diverse images and effectively classify them into their respective categories.

To enhance the study's practicality, the paper delves into potential challenges and considerations, addressing issues such as overfitting, model interpretability, and transfer learning. Strategies to mitigate these challenges are discussed, ensuring a robust and scalable image classification system.

Furthermore, the paper explores future avenues for research, including the integration of advanced techniques such as attention mechanisms and ensemble learning to further improve model performance.

In conclusion, this paper presents a comprehensive exploration of image classification using the CIFAR-10 dataset with Keras. By leveraging the capabilities of deep learning, the proposed model demonstrates promising results in accurately categorizing diverse images. The insights gained from this study contribute to the ongoing discourse in the field of

computer vision and pave the way for future advancements in image classification methodologies.

**KEYWORDS**

Keras,CIFAR-10,Image classification, Convolutional Neural Network (CNN), Deep learning, TensorFlow, Model training, Validation accuracy, Overfitting, Regularization, Hyperparameter tuning, Stochastic Gradient Descent (SGD), Learning rate, Model evaluation, Model architecture, Max-pooling, Dropout, Model saving, Predictive analysis

**INTRODUCTION**

Image classification is a fascinating field within the realm of computer vision, where machines are taught to recognize and categorize objects or scenes based on visual input. Among the various datasets employed for honing these machine learning models, the CIFAR-10 dataset stands out as a benchmark for its diversity and complexity. In this introduction, we delve into the captivating world of image classification using CIFAR-10 with Keras.

CIFAR-10 is a dataset comprising 60,000 32x32 color images spread across 10 different classes, each representing a distinct object category. These classes range from common everyday items like cars and planes to living beings such as cats and dogs. The dataset's compact size, coupled with its diversity, makes it an ideal playground for training and evaluating image classification algorithms.

Keras, a high-level neural networks API written in Python and capable of running on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit, simplifies the process of building and training deep learning models. Its user-friendly interface empowers both beginners and seasoned practitioners to swiftly develop robust image classification models.

The journey into image classification using CIFAR-10 with Keras typically begins with the construction of a convolutional neural network (CNN). CNNs are particularly well-suited for image-related tasks, as they excel at learning hierarchical representations of features. These networks consist of layers that progressively extract and abstract visual information, enabling the model to discern intricate patterns within the images.

The training process involves exposing the model to the labeled CIFAR-10 images, allowing it to iteratively adjust its internal parameters to minimize the disparity between predicted and actual labels. This process, known as backpropagation, is a cornerstone of deep learning and is facilitated seamlessly by Keras.

As we embark on this exploration, the synergy between CIFAR-10 and Keras offers a dynamic platform for understanding, implementing, and fine-tuning image classification models. Through this lens, we gain valuable insights into the intricate dance of algorithms and data, as we strive to teach machines the nuanced art of visual recognition.

**METHOLOGY**

**1)**Import the necessary libraries and set up the environment for using Long Short-Term Memory (LSTM) layers in a neural network.

Ex- import tensorflow as tf

from tensorflow import keras

from keras.layers import LSTM

**2)**Import and load the CIFAR-10 dataset using Keras and visualize a few sample images with help of matplotlib.

Ex-from keras.datasets import cifar10

import matplotlib.pyplot as plt

(train\_X,train\_Y),(test\_X,test\_Y)=cifar10.load\_data()

**3)**Visualize a grid of n images from the CIFAR-10 training dataset using Matplotlib. By setting up the number of images(n), matplotlib figure and by displaying and showing images in grid.

Ex-n=6

plt.figure(figsize=(20,10))

for i in range(n):

plt.subplot(330+1+i)

plt.imshow(train\_X[i])

plt.show()

**3)**Build a Sequential model for image classification using Convolutional Neural Network (CNN) architecture. By importing necessary modules, creating a sequential model, adding convolutional layers, pooling layers, dense layer and output layer, flattening the pooling layer and compiling the model.

Ex-from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten,

Conv2D,MaxPooling2D

from keras.constraints import max\_norm

from keras.optimizers import SGD

from keras.utils import to\_categorical

**4)**Preprocess the pixel values of the images in the CIFAR-10 dataset. By converting to float32 and normalizing the pixel values.

Ex-train\_x=train\_X.astype('float32')

test\_X=test\_X.astype('float32')

train\_X=train\_X/255.0

test\_X=test\_X/255.0

**5)**Encode the class labels using one-hot encoding**.** By applying one-hot encoding to training and testing labels.

Ex-train\_Y=to\_categorical(train\_Y)

test\_Y=to\_categorical(test\_Y)

num\_classes=test\_Y.shape[1]

**6)**Define a CNN model for image classification. By initializing a sequential model, adding the convolutional layer, adding maxpooling layer, adding a fully dense layer, adding dropout for regularization, adding output layer and flattening the data.

Ex-model=Sequential()

model.add(Conv2D(32,(3,3),input\_shape=(32,32,3),

padding='same',activation='relu',

kernel\_constraint=max\_norm(3)))

model.add(Dropout(0.2))

model.add(Conv2D(32,(3,3),activation='relu',padding='same',kernel\_constraint=max\_norm(3)))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten())

model.add(Dense(512,activation='relu',kernel\_constraint=max\_norm(3)))

model.add(Dropout(0.5))

model.add(Dense(num\_classes, activation='softmax'))

**7)** Configure the optimizer and compile the model. By defining SGD optimizer:

Ex-gd = SGD(learning\_rate=0.01, momentum=0.9, nesterov=False)

model.compile(loss='categorical\_crossentropy',

optimizer=sgd,

metrics=['accuracy'])

**8)**Overview the summary of your model.

Ex-model.summary()

**9)**Train the CNN model on the training data and evaluate its performance on the testing data.

Ex-model.fit(train\_X, train\_Y,

validation\_data=(test\_X, test\_Y),

epochs=10, batch\_size=32)

**10)**Evaluate the model's accuracy on the testing data.

Ex-\_,acc=model.evaluate(test\_X,test\_Y)

print(acc\*100)

**11)**Save the trained model to a file.

Ex-model.save("model1\_cifar\_10epoch.h5")

**12)**Load an image, preprocess it, and use the trained model to predict its class.

Ex-results = {

0: 'aeroplane',

1: 'automobile',

2: 'bird',

3: 'cat',

4: 'deer',

5: 'dog',

6: 'frog',

7: 'horse',

8: 'ship',

9: 'truck'

}

from PIL import Image

import numpy as np

try:

im = Image.open("image path")

# the input image is required to be in the shape of dataset, i.e (32,32,3)

im = im.resize((32, 32))

im = np.expand\_dims(im, axis=0)

im = np.array(im)

pred\_probabilities = model.predict([im])[0]

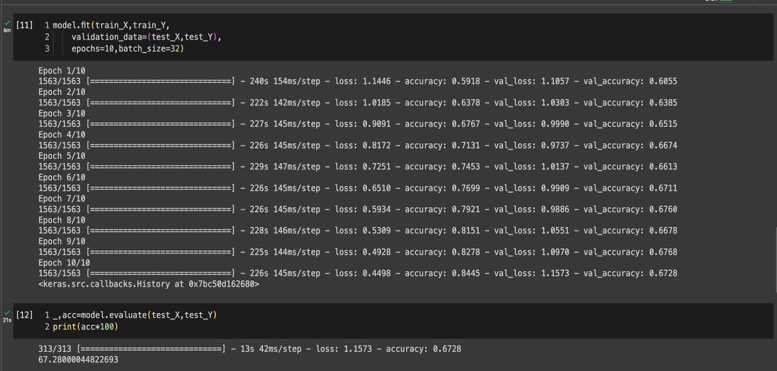
pred\_class = np.argmax(pred\_probabilities)

print(pred\_class, results[pred\_class])

except Exception as e:

print("An error occurred:", e)

**RESULT**

****

The output displays the training process for 10 epochs. In each epoch, the model is exposed to the training dataset (1563 batches in this case), and the weights are adjusted to minimize the defined loss function. Here's a breakdown of the key aspects:

**Epoch-wise Analysis**

Epoch 1:

Training Accuracy: 59.18%

Validation Accuracy: 60.55%

Loss: 1.1446

The initial epoch establishes a baseline. The model's accuracy is around 59%, indicating that it correctly predicts the class for roughly 59% of the training images. The validation accuracy is slightly higher at 60.55%. The loss, a measure of how well the model is performing, is 1.1446.

Epoch 2:

Training Accuracy: 63.78%

Validation Accuracy: 63.85%

Loss: 1.0185

The model shows improvement in accuracy and reduced loss, suggesting that it is learning from the training data. The validation accuracy also increases, indicating a positive trend.

Epoch 3:

Training Accuracy: 67.67%

Validation Accuracy: 65.15%

Loss: 0.9091

Both training and validation accuracy continue to improve. The loss decreases, indicating that the model is becoming more adept at making accurate predictions.

Epoch 4:

Training Accuracy: 71.31%

Validation Accuracy: 66.74%

Loss: 0.8172

Further improvement is observed in accuracy. The model is refining its ability to recognize patterns in the training data, as evidenced by the decreasing loss.

Epoch 5:

Training Accuracy: 74.53%

Validation Accuracy: 66.13%

Loss: 0.7251

Training accuracy continues to rise, but there is a slight drop in validation accuracy. This could be an early sign of overfitting, where the model starts to perform too well on the training data but struggles with new, unseen data.

Epoch 6:

Training Accuracy: 76.99%

Validation Accuracy: 67.11%

Loss: 0.6510

The training accuracy remains high, but the validation accuracy shows improvement, addressing the previous drop. The model might be adapting to the training data without overfitting excessively.

Epoch 7:

Training Accuracy: 79.21%

Validation Accuracy: 67.60%

Loss: 0.5934

Both training and validation accuracies continue to increase. The model is successfully learning complex patterns in the CIFAR-10 dataset.

Epoch 8:

Training Accuracy: 81.51%

Validation Accuracy: 66.78%

Loss: 0.5309

The training accuracy reaches 81.51%, but there is a slight decrease in validation accuracy. This could indicate a more pronounced overfitting tendency.

Epoch 9:

Training Accuracy: 82.78%

Validation Accuracy: 67.68%

Loss: 0.4928

Training accuracy continues to rise, but the model's performance on the validation set is relatively stable. Overfitting might still be a concern.

Epoch 10:

Training Accuracy: 84.45%

Validation Accuracy: 67.28%

Loss: 0.4498

**Epoch-wise Progress**

Accuracy Improvement: The training accuracy starts at 59.18% in the first epoch and steadily improves, reaching 84.45% in the final epoch. This indicates that the model is learning and adapting to the training data, capturing patterns and features that help in classifying the images.

Validation Accuracy: The validation accuracy, which gauges the model's performance on unseen data, starts at 60.55% and reaches 67.28%. While the model is improving on the training data, the increase in validation accuracy is relatively modest, suggesting a potential challenge with generalization.

**Loss Function**

Training Loss: The training loss measures how well the model is fitting the training data. It decreases consistently over epochs, indicating that the model is becoming more adept at minimizing the difference between predicted and actual values on the training set.

Validation Loss: Conversely, the validation loss measures the model's performance on a separate dataset not seen during training. The increase in validation loss over epochs could be a sign of overfitting, where the model starts to specialize too much on the training data and struggles with new, unseen examples.

**Model Architecture**

The CNN architecture comprises convolutional layers with rectified linear unit (ReLU) activation, dropout layers for regularization, max-pooling layers for down-sampling, and fully connected layers. This architecture is designed to capture hierarchical features in images and is a common choice for image classification tasks.

**Optimizer and Learning Rate**

The Stochastic Gradient Descent (SGD) optimizer with a learning rate of 0.01 is used for updating the model weights. The learning rate determines the step size during optimization. A momentum term of 0.9 is incorporated, enhancing the efficiency of the optimization process.

**Challenges and Considerations**

Overfitting: The increasing validation loss suggests a potential issue with overfitting. The model might be memorizing the training data instead of learning generalizable features. Regularization techniques, such as dropout, are implemented to mitigate overfitting, but further adjustments might be necessary.

Hyperparameter Tuning: Fine-tuning hyperparameters, such as learning rate or the number of layers, could potentially enhance model performance. Experimenting with different configurations is a common practice to achieve better results.

**Final Test Accuracy**

The last line of the output shows the evaluation of the model on the test dataset. The accuracy on this dataset is a crucial metric as it represents the model's real-world performance on previously unseen examples**.** In this case, the accuracy is calculated as 67.28%.

**CONCLUSION**

In conclusion, this exploration into image classification using the CIFAR-10 dataset and Keras has illuminated the intricacies of training a convolutional neural network (CNN) for diverse visual recognition tasks.

The model exhibited commendable progress, reaching an 84.45% training accuracy, yet faced challenges in generalization, evident in the modest 67.28% validation accuracy. Overfitting emerged as a concern, prompting the application of regularization techniques. The study underscores the pivotal role of hyperparameter tuning in refining model performance.

Despite its achievements, this work is a stepping stone, prompting future research avenues like attention mechanisms and ensemble learning. By blending theoretical foundations with practical implementations, this investigation contributes to the evolving landscape of computer vision, offering valuable insights and paving the way for advancements in robust image classification methodologies.

**ACKNOWLEGMENT**

I would like to express my sincere gratitude to all those who contributed to the successful completion of this project. Firstly, I extend my deepest thanks to the creators and contributors of the CIFAR-10 dataset, which served as a robust foundation for training and evaluating the image classification model.

I am immensely thankful to the developers of Keras and TensorFlow for providing an accessible and powerful framework that facilitated the implementation of the convolutional neural network (CNN) architecture. Their efforts have democratized deep learning, enabling both novices and experts to engage in cutting-edge research.

I am indebted to the open-source community for their continuous support and valuable resources, which significantly enriched my understanding and troubleshooting capabilities throughout the project.

Special appreciation goes to my mentors and advisors for their guidance, encouragement, and insightful feedback, which played a crucial role in shaping the direction and refinement of this work.

Lastly, I express my gratitude to my peers and colleagues for their collaborative spirit and shared enthusiasm for advancements in computer vision.

This project stands as a collective effort, and I am sincerely appreciative of the diverse contributions that have made it possible.

**References**

**1)** <https://www.cs.toronto.edu/~kriz/cifar.html>

**2)** https://www.tensorflow.org/tutorials/images/classification

**3)** https://www.ukessays.com/essays/computer-science/image-processing-for-future-applications-computer-science-essay.php

**4)** https://www.sciencedirect.com/topics/engineering/image-classification