An Enhanced Convolution Neural Network Approach with Higher Classification rate for Images

Dr. Y Padma[#1], Pavan Kumar Kolluru[*2], Dr. N Srinivasa Rao[#3], Dr. Sk Nagul[*4]
[#1]Assistant Professor, Department of Information Technology,
P.V.P Siddhartha Institute of Technology, Vijayawada
[*2]Assistant Professor,Department of CSE, VFSTR deemed to be University,A.P
[#3]Associate Professor, Narasimhareddy College of Engineering, Hyderabad
[*4]Associate Professor, Department of CSE, Lendi Institute of Engineering and Technology, Vizianagaram.
[#1] padmayenuga@pvpsiddhartha.ac.in
[*2] pavanwithu@gmail.com
[#3]rao75nidamanuru@gmail.com
[*4] nagulcse@gmail.com

Article Info

Page Number:* *1671 - 1678

Publication Issue:

Vol 72 No. 1 (2023)

Article History

***Article Received:** 15 October 2022*

***Revised:** 24 November 2022*

***Accepted:** 18 December 2022*

Abstract

Image categorization necessitates the development of features that can recognize image patterns that reveal group identity. The goal of this research is to classify photos from the public domain. Deep learning algorithms wereused to combine diverse picture feature sources from the CIFAR-10 image dataset. The majority of typical convolutional neural networks (CNN) have the same structure: convolutional layer modification and a max-pooling layer procedure coupled by a number of completely linked layers. The primary goal of this paper is to increase the efficiency of simple convolution neural network models. On a dataset from the Canadian Institute for Advanced Research (CIFAR-10), the Artificial Neural Network (ANN) technique isused with two distinct CNN topologies. After 10 hours of running, the updated model achieves an 88 percent classification accuracy rate. The Keras library, which is available for Python programming, is used to implement the deep learning

Keywords :-Machine Learning; Artificial Neural Networks; Convolutional Neural Networks; Keras; CIFAR-10

1. INTRODUCTION

Many hours of training are required to appropriately categorize items. People make numerous mistakes before finally getting it perfect. Machine learning follows the same structure. Deep learning can categorize objects as well as or better than humans when given a high-quality set of data. Some of the repetitive tasks could be replaced by robots if a completely accurate picture classifier is developed, allowing humanity to focus on the most fun pursuits.

Since some of the characteristics that distinguish different classes are hardly perceptible to the human eye, it is difficult to get a high classification rate on a collection of small pictures. To be the most efficient at analyzing and correctly detecting all kinds of objects, computing vision is continually developing. This type of research could help increase the effectiveness of autonomous vehicles, which are sometimes ineffective in specific scenarios of object detection, resulting in considerable harm. The majority of classic neural network methods do not produce satisfactory outcomes for the majority of potential workloads. The aforementioned fact disqualifies machines from taking over monotonous human tasks.

This study classifies 10 classes of many, evenly distributed images from the CIFAR-10 dataset using a novel type of CNN structure. The enhanced model, which is based on the model, substitutes the max-pooling and dense function with two-dimensional convolution layers, resulting in a higher classification rate.

2. LITERATURE REVIEW

The typical convolutional neural network usually initializes the weights of all network layers at once before network training, and then updates the weights of the network using the back-propagation algorithm to enhance the network's accuracy during network training, according to [1]. However, as the depth of the network grows, the computing cost of this method rises considerably, affecting the test accuracy. To solve this problem, a method of gradually reinitializing the weights of each layer is proposed, in which the weight of the previous layer is determined and remains unchanged after a certain training period, then the weights of all subsequent layers are initialized, and so on until the weights of all layers are determined. A series of experiments were conducted out using the CIFAR-10 dataset to verify the method's performance. The results reveal that the network's accuracy has increased by 9%, while training time has decreased by 29%. It demonstrates that the technique can improve network accuracy while also reducing training time.In [2,] learning the parameters to meet the goal function is part of training the deep learning models. Typically, the goal is to reduce the amount of money lost during the learning process. In the case of amodel is given data samples in the supervised method of learning and the results of their efforts.

When a model produces a result, it is called a model. It compares the output to the desired output and then takes the better of the twoattempts to reconcile the generated and desired outputs close the gap between the generated and intended outputs This is Optimization algorithms were used to achieve this. To increase the model's accuracy, an optimization method cycles through multiple cycles until convergence. To overcome the obstacles involved with the learning process, a variety of optimization strategies have been developed. Six of these have been chosen to be investigated in this study in order to learn more about their complexities. Stochastic Gradient Descent, Nesterov momentum, RMSProp, Adam, Adagrad, and Adadeltaare among the approaches examined.The trials were carried out using four datasets: MNIST, FashionMNIST, CIFAR-10, and CIFAR-100.At epoch 200, the ideal training results for MNIST are 1.00 with RMSProp and Adam, FashionMNIST is 1.00 with RMProp and Adam, CIFAR-10 is 1.00 with RMSProp at epoch 200, and CIFAR-100 is 1.00 with Adam at epoch 100.

For MNIST, FashionMNIST, CIFAR-10, and CIFAR-100, the top testing results are 0.9826, 0.9853, 0.9855, and 0.9842, respectively.

The results demonstrate that the Adam optimization algorithm outperforms the others in the testing phase, while RMSProp and Adam outperform the others in the training phase.In [3], for example Training neural networks is a computationally difficult process that takes a long time to complete. We present two ways in this paper that improve job efficiency by actively selecting the most relevant points from a training data set.

The first method creates a batch that maximizes the estimator's entropy reduction, whereas the second method only trains on datapoints with a predicted probability less than a predefined threshold. To speed up, both strategies rely on data metrics while keeping the neural network training architecture based on epochs.

The findings show that the offered methods work provide for a large reduction in the amount of time spent training in studies without affecting the accuracy of the CIFAR-10 dataset.

Proposed Work

The issue raised here affects the entire field of autonomous devices, which employ a variety of classifiers to function without human supervision. The most well-known autonomous applications can be found in the scientific and automotive fields, where researchers are working to automate everything from medical scan interpretation to vehicle operation and diagnostics. Both of the aforementioned sectors are vital to people because their lives are dependent on a single decision.To produce the most accurate projections in the shortest period of time, followed by an immediate response, equipment is automated and given the ability to do repetitive jobs. Multiple sectors could benefit from the use of the most appropriate algorithms if they knew which picture classification model was the fastest and most accurate. Because each neural network method operates differently with various datasets, the no-free-lunch theorem asserts that no single algorithm can solve all issues.Forinstance, the CNN model utilized in this research work should be most helpful in classifying a few classes made up of tiny picture data.

The presented topic is attempted to be solved using the widely used CIFAR-10 dataset, which is routinely used to evaluate image classification methods. The dataset contains 60000 32x32 colour photos divided into ten groups, each with 6000 images [3]. The complete set of photos is separated into two sets: a training set of 50000 images and a testing set of 10,000 images.Each class is represented by 5000 images in the training batch, whereas each class is represented by 1000 images in the testing batch. The CIFAR-10 dataset consists of ten classes separated into six animal categories (bird, cat, deer, dog, frog, horse) and four vehicle types (aeroplane, automobile, ship, and truck). The classes are mutually exclusive, according to the publisher, with no overlaps of likeness, such as between trucks and vehicles.

CIFAR-10 datasets are available for download on the official website [5]. The files are accessible in a variety of forms, including Python. The CIFAR-10-python.tar.gzfile needs to be extract to disclose the following files:

- batches.metaexplains the labels for the 10 different classes that are included.

- 50000 picture training data split into five 30 MBbatch files.
- readme.html — HTML page with a link to the dataset's official website.
- test batch - a 30 MB file containing 10000 photos for testing.

The primary goal of this study is to evaluate and compare the performance of several CNN models developed using Python scripts. The image classification pipeline utilised in this study reinvents the state of the art by omitting certain components seen in a traditional CNN model. The new model should enhance the classification rate if it is successfully implemented. The deep neural network models are tested on the CIFAR-10 dataset, which is a well-known image classification dataset, and the results are then compared to the rest of the published solutions.

Step 1: From Keras library CIFAR-10 dataset [6] can be downloaded.
Step 2: The files must be decrypted using a Python script before we can create a model and utilize the dataset to see the actual images.
Step 3: We can build two CNN model and trained that models on training dataset.
Step 4: We can test the model on testing dataset and evaluate the performance

Experimental & Result Analysis

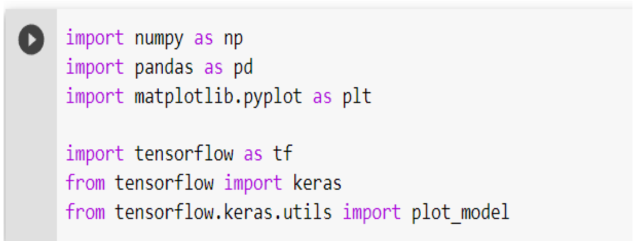
Deep neural network models take a long time to execute. All of the software is installed on top of Python, allowing for the use of Google Collab to run the full code. The decoder file uses the Python programming language to import the CIFAR-10 dataset into a Google Collab in a suitable format.

Python requires the following packages to run CNN models:

• Keras 2.1.5
• Tensorflow 1.7.0
• NumPy 1.14.2
•Matplolib 2.2.2

The Keras package, which initialises CNN models, is the key component of the overall setup [7]. The package's backend might be based on a CPU or GPU component, which can affect the experiment's execution time.

Each Python project begins with the import of commonly used packages, as illustrated in Figure 2, which make typing code easier. The Keras package, which runs on top of Tensorflow — a Google-developed open source machine learning framework that allows for speedier development of CNN networks in Python code – is the core component of the following configuration.



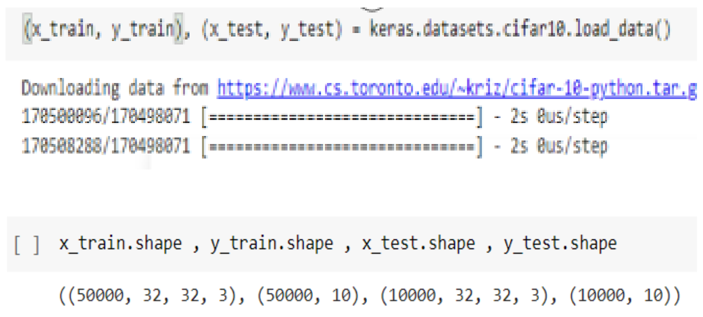
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.utils import plot_model
```

Fig. 1. Import of packages

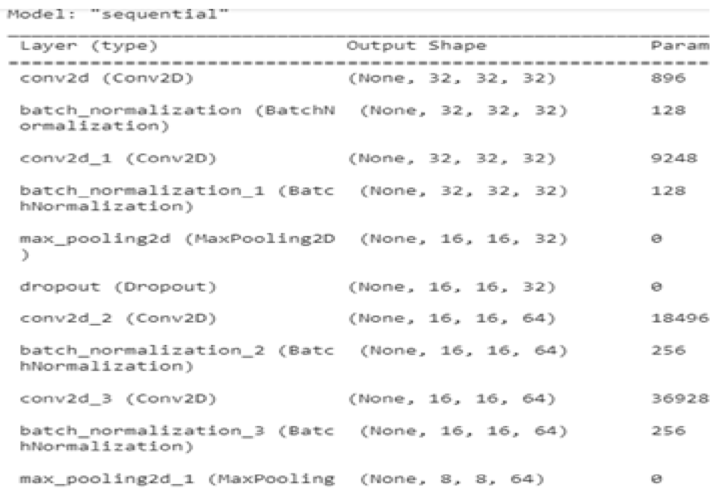
Second, the work necessitates the inclusion of the CIFAR-10_models.ipynb fileprovides a set of decoding functions for the CIFAR-10visualising the labels and using the Pickle package to analyse the data. The NumPy library is used to convert data into arrays.

The next step is to import the functions from the CIFAR-10models.ipynb file and loadingthe class names and then checkingclasses count and specify the input image's dimensions as 32 by 32 pixels with 3 channels RGB. The pixel intensity at that point is indicated by the three arrays of integers, which have values ranging from 0 to 255. The CNN can use this information to describe the likelihood of an object belonging to a given class.

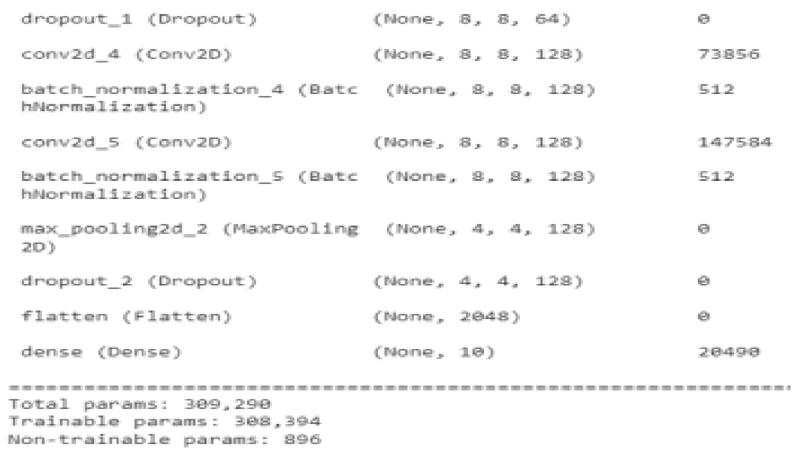
The images must also be decoded and retrieved, which is a major effort.Integer data types are used for class labels, while one-hot encoded vectors are used for classes. The CIFAR-10 data is divided into ten categories83 percent (50000) of the photos were used for training, while the rest were used for other purposes.For testing, 17 percent (10000) is used, which is validation.



```
(x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.load_data()

Downloading data from https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.g
170500096/170498071 [==============================] - 2s 0us/step
170508288/170498071 [==============================] - 2s 0us/step

[ ]  x_train.shape , y_train.shape , x_test.shape , y_test.shape

     ((50000, 32, 32, 3), (50000, 10), (10000, 32, 32, 3), (10000, 10))
```



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param
=================================================================
 conv2d (Conv2D)             (None, 32, 32, 32)        896

 batch_normalization (BatchN  (None, 32, 32, 32)       128
 ormalization)

 conv2d_1 (Conv2D)           (None, 32, 32, 32)        9248

 batch_normalization_1 (Batc  (None, 32, 32, 32)       128
 hNormalization)

 max_pooling2d (MaxPooling2D  (None, 16, 16, 32)       0
 )

 dropout (Dropout)           (None, 16, 16, 32)        0

 conv2d_2 (Conv2D)           (None, 16, 16, 64)        18496

 batch_normalization_2 (Batc  (None, 16, 16, 64)       256
 hNormalization)

 conv2d_3 (Conv2D)           (None, 16, 16, 64)        36928

 batch_normalization_3 (Batc  (None, 16, 16, 64)       256
 hNormalization)

 max_pooling2d_1 (MaxPooling  (None, 8, 8, 64)         0
```

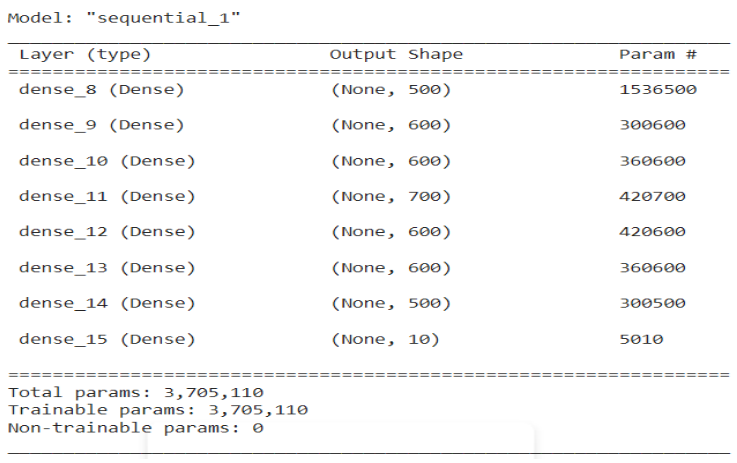


```
dropout_1 (Dropout)          (None, 8, 8, 64)          0

conv2d_4 (Conv2D)            (None, 8, 8, 128)         73856

batch_normalization_4 (Batc  (None, 8, 8, 128)         512
hNormalization)

conv2d_5 (Conv2D)            (None, 8, 8, 128)         147584

batch_normalization_5 (Batc  (None, 8, 8, 128)         512
hNormalization)

max_pooling2d_2 (MaxPooling  (None, 4, 4, 128)         0
2D)

dropout_2 (Dropout)          (None, 4, 4, 128)         0

flatten (Flatten)            (None, 2048)              0

dense (Dense)                (None, 10)                20490

=================================================================
Total params: 309,290
Trainable params: 308,394
Non-trainable params: 896
```

Fig. 2.Process of Fetching and compiling the data of Deep CNN

Since using the model, each layer can be created in a single line of code. The Keras package's add() method makes creating models pretty simple. After executing the function indicated in Figure 4, The self-explanatory code for the entire process summarizes the CNN model using four 2D layers

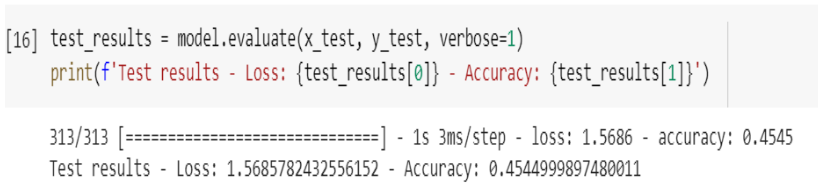
The model is first configured with a sequential function, allowing for the development of a stack of objects that is a linear stack of layers, with each layer transmitting data to the next. The first two Convolution2D(32x(3x3) scripts create 32 3x3 convolution filters, followed by two more Convolution2D(64x(3x3) scripts that use a larger number of filters.



```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_8 (Dense)             (None, 500)               1536500

 dense_9 (Dense)             (None, 600)               300600

 dense_10 (Dense)            (None, 600)               360600

 dense_11 (Dense)            (None, 700)               420700

 dense_12 (Dense)            (None, 600)               420600

 dense_13 (Dense)            (None, 600)               360600

 dense_14 (Dense)            (None, 500)               300500

 dense_15 (Dense)            (None, 10)                5010

=================================================================
Total params: 3,705,110
Trainable params: 3,705,110
Non-trainable params: 0
_________________________________________________________________
```

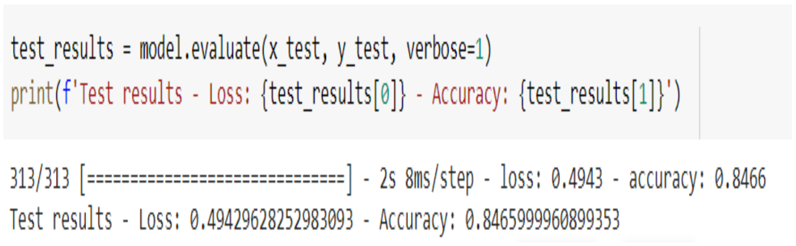
Fig. 3.Multilayer Perceptron

There is no change in training the model because same parameters are used, with the addition of 30 more training epochs being the only changeand the prediction result are shown in figure



```
[16] test_results = model.evaluate(x_test, y_test, verbose=1)
     print(f'Test results - Loss: {test_results[0]} - Accuracy: {test_results[1]}')

     313/313 [==============================] - 1s 3ms/step - loss: 1.5686 - accuracy: 0.4545
     Test results - Loss: 1.5685782432556152 - Accuracy: 0.4544999897480011
```

Fig. 4.MultiLayer Perceptron Accuracy



```
test_results = model.evaluate(x_test, y_test, verbose=1)
print(f'Test results - Loss: {test_results[0]} - Accuracy: {test_results[1]}')

313/313 [==============================] - 2s 8ms/step - loss: 0.4943 - accuracy: 0.8466
Test results - Loss: 0.49429628252983093 - Accuracy: 0.8465999960899353
```

Fig. 5. Deep Convolution Neural Network Accuracy

The accuracy of all the tested models, are shown in the Table 1.

Table 1. Comparison of Accuracy

Classification Model	**Accuracy**
Multi Layer Perceptron	45.49%
Deep CNN	84.65%

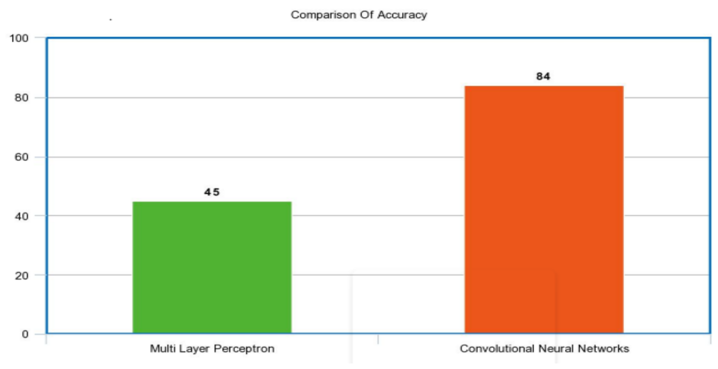


Fig. 6.Comparison of models

Between the simplest and most complicated models utilized in the experiment, the results show a 10% improvement. The accuracy of the CNN model improved to 84.94% after deleting the max-pooling and dense functions; however, because the upgraded CNN had to run three times more epochs, the running time increased.

Conclusion

The significance of the study's findings lies in the demonstration that CNN models' accuracy may be increased by simply executing and modifying its conventional structure using a programming language One of the most interesting discoveries was the accuracy to running time ratio, which showed that while the final model needed the most processing power to achieve the best accuracy, the most conventional CNN structure had the best accuracy to running time ratio. Based on the components they have, the AI implementers would need to decide whether it is worthwhile to use the more robust models.

References

[1] A. Krizhevsky, I. Sutskever and G. E. Hinton, "Imagenet classification with deep convolutional neural networks" in Advances in Neural Information Processing Systems, Curran Associates, Inc., vol. 25, pp. 1097-1105, 2012

[2] Zaheer, Raniah, and HumeraShaziya. "A study of the optimization algorithms in deep learning." 2019 third international conference on inventive systems and control (ICISC). IEEE, 2019.

[3] Mourad, Sara, HarisVikalo, and Ahmed Tewfik. "Online selective training for faster neural network learning." 2019 IEEE Data Science Workshop (DSW). IEEE, 2019.

[4] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[5] S. Liu and W. Deng, "Very deep convolutional neural network based image categorization with short training sample size," in 2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR), pp. 730–734, Nov. 2015.

[6] A. Krizhevsky, V. Nair and G. Hinton, Cifar-10 (cana-dian institute for advanced research), [online] Available: http://www.cs.toronto.edu/kriz/cifar.htm/.

[7] Planche, Benjamin, and Eliot Andres. Hands-On Computer Vision with TensorFlow 2: Leverage deep learning to create powerful image processing apps with TensorFlow 2.0 and Keras. Packt Publishing Ltd, 2019.