IJARSCT

ISSN (Online) 2581-9429

International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)

Impact Factor: 6.252

Volume 2, Issue 4, May 2022

CIFAR-10 Image Classification with Convolutional Neural Networks

Praneeta Handral, Ritika Kulkarni, Swapna MD, Nikhil Kumar
Students, Department of Computer Science and Engineering,
Alva's Institute of Engineering and Technology, Mangalore, India

Abstract: *In this project, we work on image classification of the CIFAR-10 dataset using supervised machine learning techniques. The dataset consists of 60,000 32x32RGB images containing one of 10 object classes, with 6000 images per class. We experiment with various learning algorithms including nearest neighbor classifier, one-vs-all classification, Softmax classifier, two-layer fully connected artificial neural network (ANN), deep convolutional neural network (CNN), and deep residual networks (ResNet). We use cross validation by splitting the 50,000training data into49,000 training samples and 1,000 validation samples to select the optimized hyper parameters for each parametric classifier. Among all methods, the 56-layer deep residual network yields the best performance with a training accuracy above 99% and validation accuracy of 93.6%.*

Keywords: Image Classification; CIFAR-10; Supervised Machine Learning Algorithm; Deep Convolutional Neural Network (CNN); Deep Residual Network (ResNet).

I. Introduction

Image classification is an active area of research and has been studied in common applications such as unmanned vehicles and emergency robotics. In this paper, an integrated neural network (CNN)-based architecture is proposed using the Cifar 10 dataset, which has a total of 60,000 images [1]. These images are divided into training and testing sections, each with 50,000 and 10,000 images respectively. Although the CNN-based image classification methods presented herein are very efficient, they require a large amount of memory. The purpose of our article is to perform image classification under limited memory conditions. Such cases often occur in embedded systems. The proposed method achieves an accuracy of 85.9 while requiring only 2 GB of GPU memory.

II. Problem Definition

The problem described is of importance throughout the field of automated equipment, which implements several diverse classifiers to operate independently of human control. The most common automated applications can be seen in the automotive and scientific industries, where scientists trying to automate vehicle operations or analyze medical scans, will also come up with accurate diagnosis. Both the mentioned areas are very important for human because life depends on a decision. The goal of automating devices and allowing them to take control of monotonous tasks is to make the most accurate predictions.

Respond immediately. Knowledge of the fastest and most accurate image classification model can help multiple industries implement the most appropriate algorithms. However, according to the no-free lunch theorem, each neural network algorithm behaves differently on different datasets, so there is no single algorithm that solves all sorts of problems. For example, the CNN model examined in this study should help classify a small number of classes that consist primarily of small images. Attempts to resolve the issues presented are performed on the widely used CIFAR10 dataset. It is often used to evaluate image classification algorithms. The dataset contains "60,000 32x32 color images in 10 layers, 6,000 images in each layer" [3]. The full set of images is split into 2 sets of 50,000 images for the training set and the remaining 10,000 images for the test set. The training pack contains 5000 images for each class and the test pack contains 1000 representative images for each class. The CIFAR10 dataset is a set of 10 classes divided into 6 animal categories (birds, cats, deer, dogs, frogs, horses) and 4 vehicle types (airplanes, cars, boats, trucks). As the editors pointed out, the classes are mutually exclusive and do not have the following overlapping similarities: B. Between the truck and the car. The CIFAR10 dataset can be downloaded from the official website [5]. These files are available for download in a variety of Python, Matlab, and binary versions suitable for C programs. After downloading the Python version (cifar10 python.tar.gz), the data will be extracted and you will see the following file:

- batches.meta - includes a Python dictionary object that identifies the tag names of the 10 included classes.
- data_batch_1, data_batch_2,…, data_batch_5 - training data of 50,000 images divided into five files, 30 MB each.
- readme.html - HTML document that links to the official website of the dataset.
- test_batch - test data of 10,000 images in a 30MB file.

To view the actual images and use them, the files must decoded with a Python script.

III. Proposed Work

The main object of the present study is to test and compare the performance of different CNN models performed with the use of scripts written in the Python programming language. The small image classification pipeline used in this project is a modern reconstruction, removing some of the components used in the standard CNN model. If successfully implemented, the reinvented model will improve the classification rate. Test the deep neural network model applied on one of the famous image classification datasets, that is the CIFAR10 dataset, because the results can then be compared with the rest of the solutions announced.

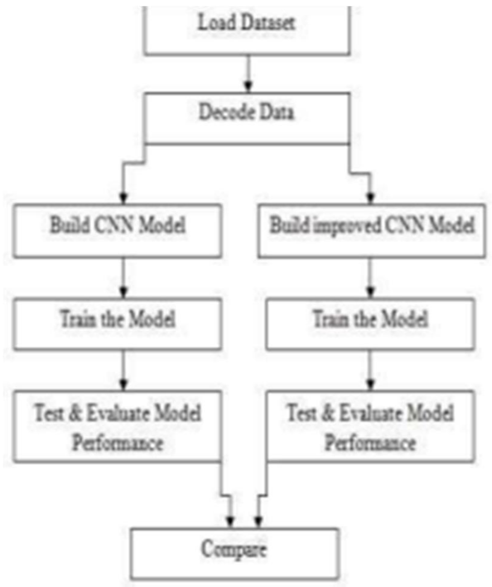


Figure 1: Proposed block diagram

Step 1: We can download the CIFAR-10 dataset from reputable website.

Step 2: Before constructing a version we are able to decode the dataset to peer the real pictures and use them, the documents must be decoded with the usage of a Python script.

Step 3: We can construct CNN version and educated that fashions on education dataset.

Step 4: We can take a look at the version on checking out dataset and examine the overall performance end result of each the version and evaluate the overall performance.

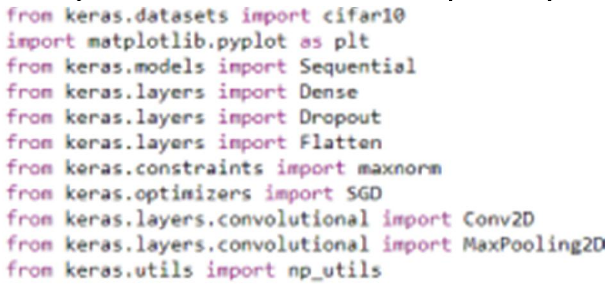
IV. Experimental And Result Analysis

Running deep neural network models is very time consuming. All software is installed on Python, allowing Jupyter Notebook to be used to run all the code. The Python programming language is implemented in a decoder file to import the CIFAR10 dataset into a Jupyter notebook in a compatible format.

To run CNN models, Python requires the following packages:

- Tensorflow 1.7.0
- Keras 2.1.5
- Pickle (Python's built-in package)
- NumPy 1.14.2
- Matplolib 2.2.2

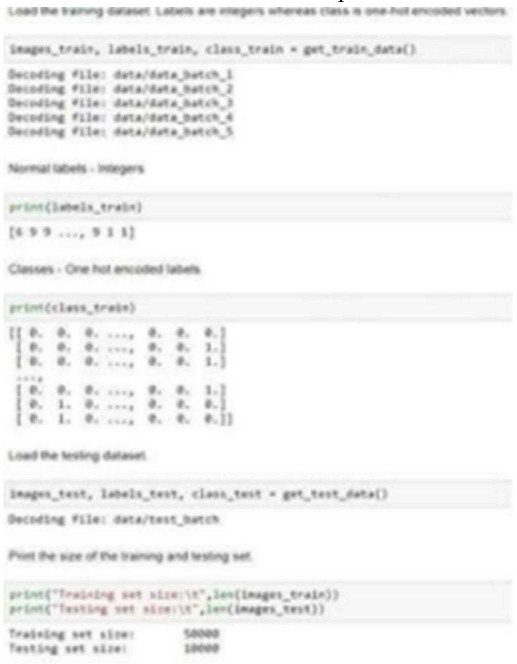
The Keras library is a key component of the entire setup that initializes the CNN model [14]. This package allows you to create a backend based on CPU or GPU components. This can change the execution time of the experiment. First, all Python projects require you to import used packages, as shown in Figure 2. This makes it easier to enter the code. The main component of the next setup is the Keras package running in Tensorflow-Open source machine learning framework developed by Google that enables faster implementations of CNN networks in Python script.



```
from keras.datasets import cifar10
import matplotlib.pyplot as plt
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import Dropout
from keras.layers import Flatten
from keras.constraints import maxnorm
from keras.optimizers import SGD
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils import np_utils
```

Figure 2: Imports of packages

Second, the project requires importing the file "decoder.py" containing a set of functions written to decode the CIFAR10 dataset using the Pickle package and plot the labels and image data into the tables using the NumPy library. The code for the entire decoder can be accessed from the link at the end of this chapter for further study.



Load the training dataset. Labels are integers whereas class is one-hot encoded vectors.

```
images_train, labels_train, class_train = get_train_data()

Decoding file: data/data_batch_1
Decoding file: data/data_batch_2
Decoding file: data/data_batch_3
Decoding file: data/data_batch_4
Decoding file: data/data_batch_5
```

Normal labels - Integers

```
print(labels_train)

[6 9 9 ..., 9 1 1]
```

Classes - One hot encoded labels

```
print(class_train)

[[ 0.  0.  0. ...,  0.  0.  0.]
 [ 0.  0.  0. ...,  0.  0.  1.]
 [ 0.  0.  0. ...,  0.  0.  1.]
 ...,
 [ 0.  0.  0. ...,  0.  0.  1.]
 [ 0.  1.  0. ...,  0.  0.  0.]
 [ 0.  1.  0. ...,  0.  0.  0.]]
```

Load the testing dataset

```
images_test, labels_test, class_test = get_test_data()

Decoding file: data/test_batch
```

Print the size of the training and testing set

```
print("Training set size:\t",len(images_train))
print("Testing set size:\t",len(images_test))

Training set size:      50000
Testing set size:       10000
```

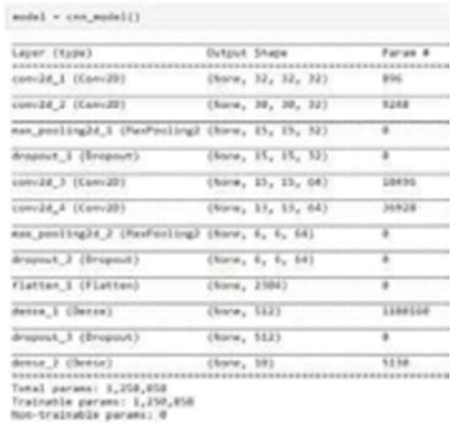
Figure 3: Process of decoding and fetching data

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Impact Factor: 6.252
Volume 2, Issue 4, May 2022

The next step requires using the functions imported from the "decoder.py" file to load the layer name, check the number of layers, and set the size of the input image, which is 32 x 32 pixels, and specify the channel number as three (red, green and blue). The three arrays of numbers are made up of values from 0 to 255, indicating the intensity of the pixels at that point. With this information, the CNN can describe the probability that an object belongs to a particular class.

Another important operation is decryption and image recovery. Class labels use integer data type, while class use onehot encoded vector. The CIFAR10 dataset was divided into two sets: 83% (50,000) images for training and the remaining 17% (10,000) for testing, tested using “print()” shown in Figure 3.

The Keras library makes it very intuitive to create a model because you can use the model.add () function to define each layer with a single line of code. The code used for the entire operation is self-explanatory, and after executing the function shown in Figure 4, we use four 2D layers to summarize the CNN model. First, the model is initialized with a sequential function that can build a linear stack of layers that are treated as a stack of objects, and each layer passes data to the next layer. The first two "Conv2D (32, (3, 3)" scripts each initialize 32 convolution filters of 3x3 size, and then two more "Conv2D (64, (3, 3)" scripts are more Use many filters.

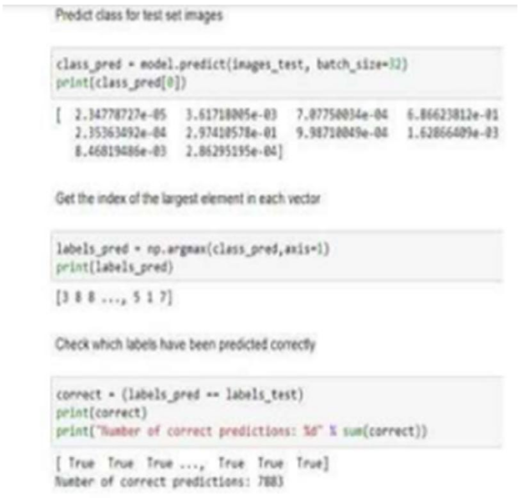


```
model = cnn_model()

Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 32, 32, 32)        896
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 30, 30, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 15, 15, 32)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 64)        18496
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 13, 13, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 64)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_1 (Dense)              (None, 512)               1180160
_________________________________________________________________
dropout_3 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                5130
=================================================================
Total params: 1,250,858
Trainable params: 1,250,858
Non-trainable params: 0
```

Figure 4: Simple CNN model building

Afterwards, the model is trained on the training data. The ModelCheckpoint () method saves the subsequent optimal model All ages. The compilation method described in the previous chapter specifies the loss function, optimizer, and model evaluation metrics. Finally, the model fits the provided data to a batch size of 128. This is the number of samples per gradient update. The model uses 100 iterations (epochs). After successful training, the model is evaluated and the accuracy and loss are plotted on the Matplotlib graph. The IPython notebook contains some scripts for predicting the test image classes shown in Figure 5.



Predict class for test set images

```
class_pred = model.predict(images_test, batch_size=32)
print(class_pred[0])
```

```
[ 2.34778727e-05  3.61718005e-03  7.07750034e-04  6.86623812e-01
  2.35363492e-04  2.97410578e-01  9.98710049e-04  1.62866409e-03
  8.46819486e-03  2.86295195e-04]
```

Get the index of the largest element in each vector

```
labels_pred = np.argmax(class_pred,axis=1)
print(labels_pred)
```

```
[3 8 8 ..., 5 1 7]
```

Check which labels have been predicted correctly

```
correct = (labels_pred == labels_test)
print(correct)
print("Number of correct predictions: %d" % sum(correct))
```

```
[ True  True  True ...,  True  True  True]
Number of correct predictions: 7883
```

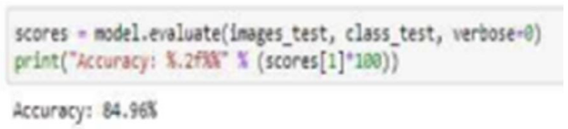
Figure 5: Sample predictions

4.1 Improved CNN Model

Firstly, the pocket book calls for uploading some extra features from Keras and decoder packages. The manner of uploading magnificence names in addition to fetching and deciphering the facts stays unchanged. Definition of the stepped forward CNN version includes the maximum vital modifications proven in figure 6. As referred to withinside the introduction, the stepped forward version replaces the max-pooling and dense feature with two- dimensional convolution layers. The structure makes use of 9 layers with a specific quantity of convolution filters. In the end, the version makes use of the operation of two-dimensional worldwide common pooling. After the shape definition, the version is constructed and summarized.

The model is trained using the same parameters, where the only difference is the increased number of 350 epochs. and the prediction result are shown in figure 6.

Evaluate the model



```
scores = model.evaluate(images_test, class_test, verbose=0)
print("Accuracy: %.2f%%" % (scores[1]*100))

Accuracy: 84.96%
```

Figure 6: Improved Model prediction

4.2 Comparison

The accuracy, as well as running time of all the tested models, are presented in the following table

Table 1. Accuracy Of The CNN Models

Classification Model	Accuracy
Simple CNN	78.83%
Modified CNN	84.96%

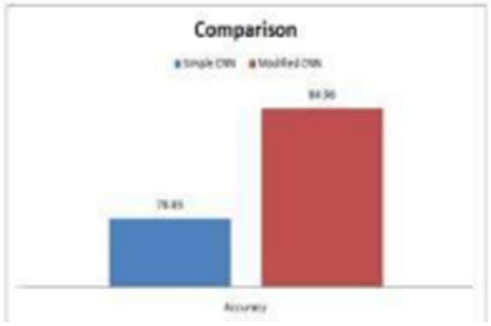


Figure 7: Comparison of models

The results show a 10% improvement between the simplest model and the most advanced model used in the test. The CNN model after removing the maxpooling function and the density function improved the accuracy up to 87.94%; however, rom three times as many epochs, uptime has increased for enhanced CNNs. Now we illustrate the model accuracy and model loss of each of the tested CNN model shown in figure 8.

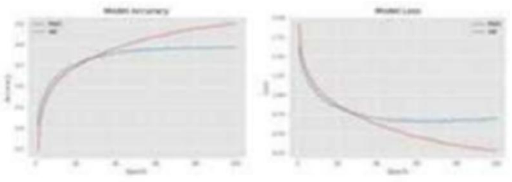


Figure 8: Simple CNN model accuracy and loss

From the first figure, it can be observed that the model stops improving its accuracy after about 60 epochs, while the loss stabilizes after about 40 epochs. The second graph shown in Figure 9 of the advanced model shows the same situation as the simple CNN model. Similarly, the model stops improving its accuracy after about 60 epochs, while the model loss starts to increase slowly after about 150 epochs, resulting in overfitting of the data.

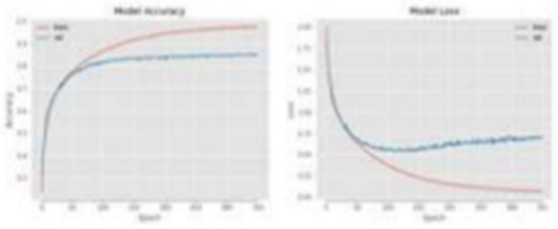


Figure 9: Improved CNN model accuracy and loss

V. Conclusion

The results obtained in this study are important because they suggest that the accuracy of the CNN model can be improved simply by using a programming language to execute and modify traditional structures. One of the interesting conclusions is the ratio of accuracy to delay. This is because the last model required the most computational power to achieve the highest accuracy, while the most traditional CNN structure achieved the highest delay-to accuracy ratio. Based on existing components, AI implementers need to determine if it is worth relying on a more robust model.

References

[1]. Raniah Zaheer , Humera Shaziya “A Study of the Optimization Algorithms in Deep Learning” in IEEE 2019.
[2]. K. Simonyan and A. Zisserman, “Very deep convolutional networks for Large-Scale image recognition,” Sep. 2014.
[3]. Ian Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, 2016.
[4]. https://www.ijcaonline.org/archives/volume176/nu mb er37/pandit-2020-ijca-920489.pdf
[5]. http://www.divaportal.org/smash/get/diva2:11111 44/F ULLTEXT02.pdf
[6]. https://scihub.hkvisa.net/10.1109/IAEAC47372.2019. 8997743
[7]. https://scihub.hkvisa.net/10.1109/AICCSA.2018.861 2 873
[8]. http://www.divaportal.org/smash/get/diva2:11111 44/F ULLTEXT02.pdf
[9]. https://www.ijert.org/rfid-based intelligent- busmanagement-and-monitoring system
[10]. https://scihub.hkvisa.net/10.1109/ICCASM.2010.5 62 0407
[11]. https://scihub.hkvisa.net/10.1109/IAEAC47372.2019 .8997743
[12]. https://scihub.hkvisa.net/10.1109/IAEAC47372.2019 .8997743
[13]. https://ieeexplore.ieee.org/document/8997743
[14]. http://www.divaportal.org/smash/get/diva2:1111144/FULLTEXT02.pdf