International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

Int. J. Sci. Res. Arch.
IJSRA
International Journal of Science and Research Archive
Research Journal Archive, INDIA

(Research Article)

Check for updates

Image recognition in self-driving cars using CNN

Shreya Muppidi [*]

Department of Mechanical Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Hyderabad, India.

International Journal of Science and Research Archive, 2023, 09(02), 342–348

Publication history: Received on 11 June 2023; revised on 24 July 2023; accepted on 27 July 2023

Article DOI: https://doi.org/10.30574/ijsra.2023.9.2.0574

Abstract

The concept of neural networks has existed for over decades but was never considerably acknowledged as much as of today. The main reason happens to be "data." To analyze a problem statement using neural networks, large data is required in its various forms and therefore it has not been instigated back in the day. But now, with today's vast technology, neural networks have begun to take over some of the numerous machine learning applications with the help of huge datasets. In this research paper, a certain deep learning approach namely convolutional neural network (CNN) has been discussed which plays a major role in classifying and recognizing objects i.e., obstacles on the road. Earlier, computer-based algorithms have been followed for image processing in vehicles which seemed to be applicable to a certain extent. So much so, now with deep learning approaches, simpler yet faster networks can be implemented for a safe drive. Automatic vehicles such as Tesla which is examined to be "fully self-driving" nevertheless needs a driver to watch over the road at some particular point. This proves that there is not yet a fully controlled self-driving car created which can drive itself without a spectator. This appeal can be solved by means of image detection mechanisms using neural networks along with a programming language to deploy machine learning models at ease. The main objective is to develop a simple and accurate algorithm to make image recognition more precise for a better self-driving car.

Keywords: Image recognition; Self-driving cars; Deep learning; Neural networks; CNN

1. Introduction

Arthur Lee Samuel defines machine learning as the field of study that gives computers the ability to learn and perform tasks without being explicitly programmed. Of course, a software can be programmed to achieve a function but a new approach called machine learning followed by deep learning is radically changing how we create software to solve these problems. There are many different approaches to machine learning but all these different types of learning use statistical algorithms and data. Deep learning, on the other hand, is just a type of machine learning, inspired by the structure of a human brain. Deep learning algorithms attempt to draw similar conclusions as humans would, by simultaneously analysing data with a given radical structure. To achieve the following, deep learning uses a multi-layered structure of algorithms called neural networks.

For a system to inherit computer vision, it is required to program and train a computer by breaking the task down into smaller and easier details. The task is as terrifying as it sounds nevertheless, simpler when broken down into a hierarchical. This is where machine learning and artificial intelligence comes to use. Generally, image recognition is the ability of a system to identify people, places, objects, and actions in images. It makes the use of technologies involving machine vision and algorithms with artificial intelligence to recognize and distinguish images through a camera system.

As it is known that a computer sees an image or video in the form of 0s and 1s. It stores the image in the form of combination of pixels which is the smallest unit in an image. Each pixel contains a various number of channels. A

[*] Corresponding author: Shreya Muppidi

grayscale image has only one pixel, whereas a coloured image contains three channels namely red, green, and blue. In a digital-coloured image, each channel of each pixel has a value between 0 and 255. Each of these values when represented in binary can be understood by the computer. But the problem is just being able to read the image is of no use if it cannot understand what it means. This is where machine learning comes into the picture.

1.1. Use case of image recognition in today's technology

Several different use cases of image recognition are in production and are already deployed on a large scale in various sectors for example self-driving cars. The gaming field has started using image recognition technology combined with augmented reality to their advantage, as it helps in providing gamers with a realistic experience. Some of the popular games using this feature are Pokémon Go, The Walking Dead, Harry Potter: Wizards Unite and many more.

Social networks also use image data which can be analyzed and visualized to comprehend customer preferences, further this data can be used for customized marketing. A powerful commercial use can be seen in the field of stock photography and video as well where stock websites provide platforms so photographers and video makers can sell their content.

Google photos which consist of a huge visual dataset uses image recognition along with its subfield deep learning, to sort millions of images on the internet in order to classify them more accurately. Pinterest uses algorithms to identify the patterns in a picture that have been pinned so that similar images are displayed when you search for them which works as an image recommender Similarly, there are many other websites and companies that use this technology to develop and improvise their services and marketing.

2. Material and methods

Image recognition is the creation of a neural network that processes all the pixels that build to be a complete image. All image recognition models start with encoders which are made of blocks of layers that learn specific patterns of the pixels of images that correspond to their labels. These encoders then give outputs of confidence scores for every input label provided which varies depending on the type of class of the image recognition system. Every machine learning model requires one common element in order to process the whole operation which is data in the form of datasets. Popular image recognition benchmark datasets include CIFAR, ImageNet, COCO, Open Images, and many other datasets. With the help of these datasets the model can be trained and tested to its best accuracy.

2.1. Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are one of the most popular deep learning neural network methods. As deep learning deals with large amounts of data when compared to any other approach it makes CNN even more applicable in many real-life applications. Any deep learning model requires a large amount of data to train which also requires a lot of computing resources. This was a major drawback for CNNs during that time and hence they were only limited to the postal sectors and it failed to enter the world of machine learning. But now with the growing technology and increasing problem statements, more data is collected and stored into datasets which will thereby be used for making machine learning models.

These neural networks are composed of multiple layers of artificial neurons. The first layer usually extracts basic features such as horizontal or diagonal edges. The corresponding output is passed on to the next layer which detects more complex features such as corners or combinational edges. As we move deeper into the neural network, it can identify even more complex features such as objects, faces, etc. Alongside the input and output, the CNN consists of hidden layers which typically consist of a series of convolutional layers. ReLU is the typical activation function which is thereby followed by additional operations such as pooling layers, fully connected layers, and normalization layers. Backpropagation is a method used for error distribution and weight adjustment. The efficient use of convolutional neural networks depends on more layers and larger networks i.e., larger the dataset, higher the efficiency of CNNs.

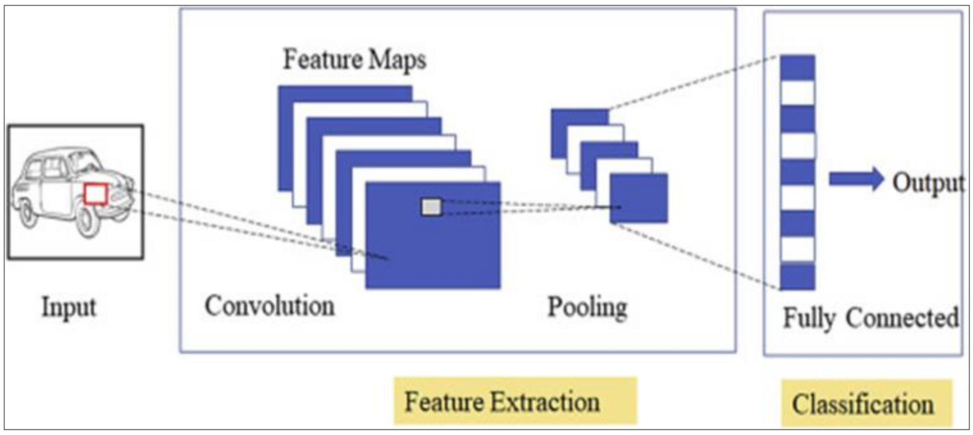


Figure 1 Structure of a convolutional neural network

2.2. Tools used in making an image recognition model:

Other than programming languages like Python, R and Java and basic Python libraries like NumPy, Matplotlib, Pandas, SciPy, etc. there are certainly other libraries and software which help in building an image recognition model in a much more accurate and faster way. Some of them are listed below:

- OpenCV is a large open-source, cross platform library for computer vision, machine learning, and image processing. Currently now it plays a major role in real-time operation which is very important in today's technology. One can process images and videos to identify objects, faces, or even handwriting of a human by using this library. The identification of image patterns and its other features is achieved by using vector space and performing mathematical operations. Using OpenCV we can also analyse videos and estimate the motion in it, subtract the background, and finally track objects in the video.
- PyTorch is an open-source framework that eases one's way to develop machine learning models and deploy them to production. PyTorch provides dynamic computation graphs and libraries for distributed training, which are tuned for high performance on AWS. Thus, it is one of the most popular deep learning libraries competing with Keras and TensorFlow. Using PyTorch, one can process images and videos to develop a highly accurate and precise computer vision model.
- YOLO is an acronym for "you only look once". It basically is a real time object detection system. Instead of using classifiers to perform detection of objects, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly. The biggest advantage of YOLO is its incredible speed. It is very fast and can process 45 frames per second. It has been used in various applications to detect traffic signals, people, parking meters, animals, and other objects.

2.3. Procedural Method

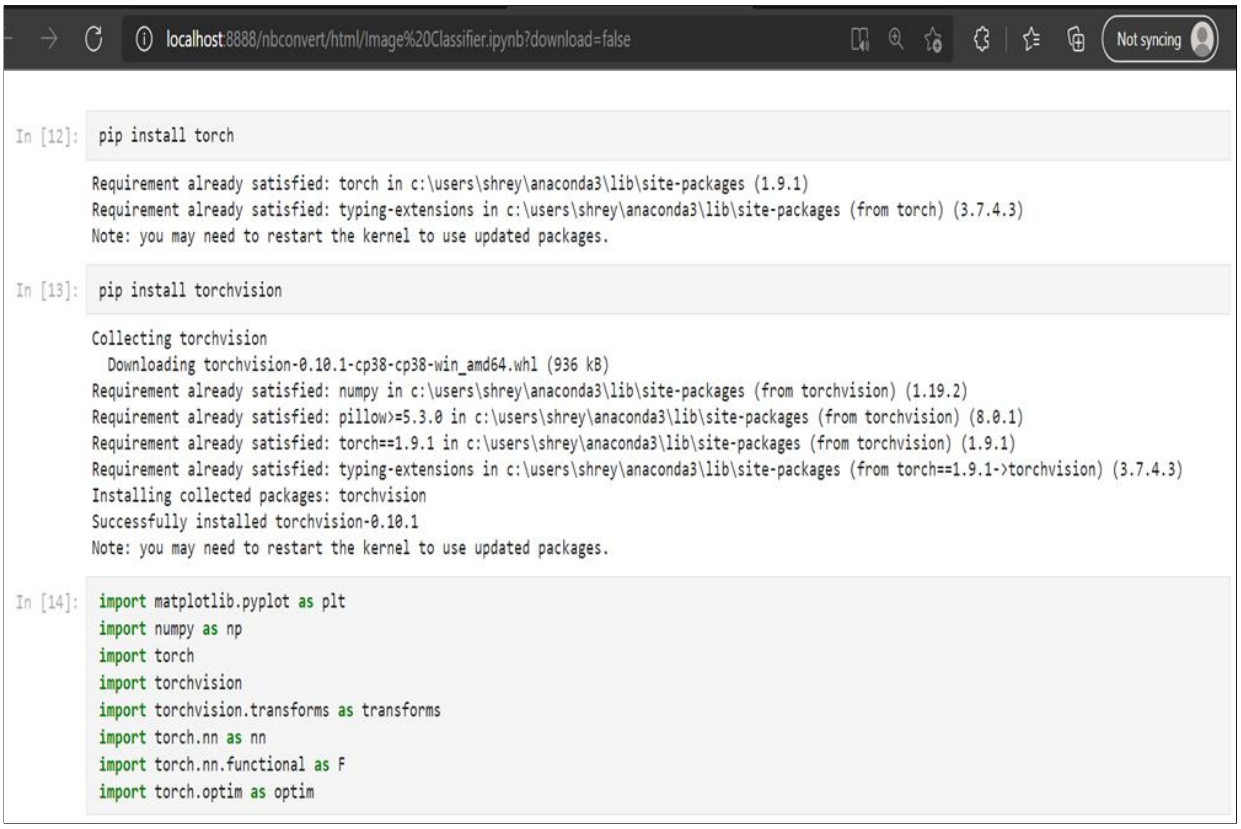
In order to build an image recognition model, we can use conventional statistical approaches such as decision trees or support vector machines but the disadvantage is that it is very time consuming and inaccurate. In today's time, the state-of-the-art method used is convolutional neural networks (CNNs). The larger the dataset, the higher is the performance of the model especially for deep neural networks. Based on the recent challenges, a CNN model made predictions on millions of pictures with 1000 classes which is close to the performance of an actual human being.

To build an image recognition model using CNN, we follow the listed steps:

- Import the required libraries
- Load and normalise the train and test datasets
- Define the convolutional neural network
- Define the loss function and optimizer
- Train the model on the train data
- Test the model on the test data

The dataset used in building this image recognition model is the CIFAR 10 dataset which consists of 10 classes as it says in the name.

2.4. Image Classifier model using CNN

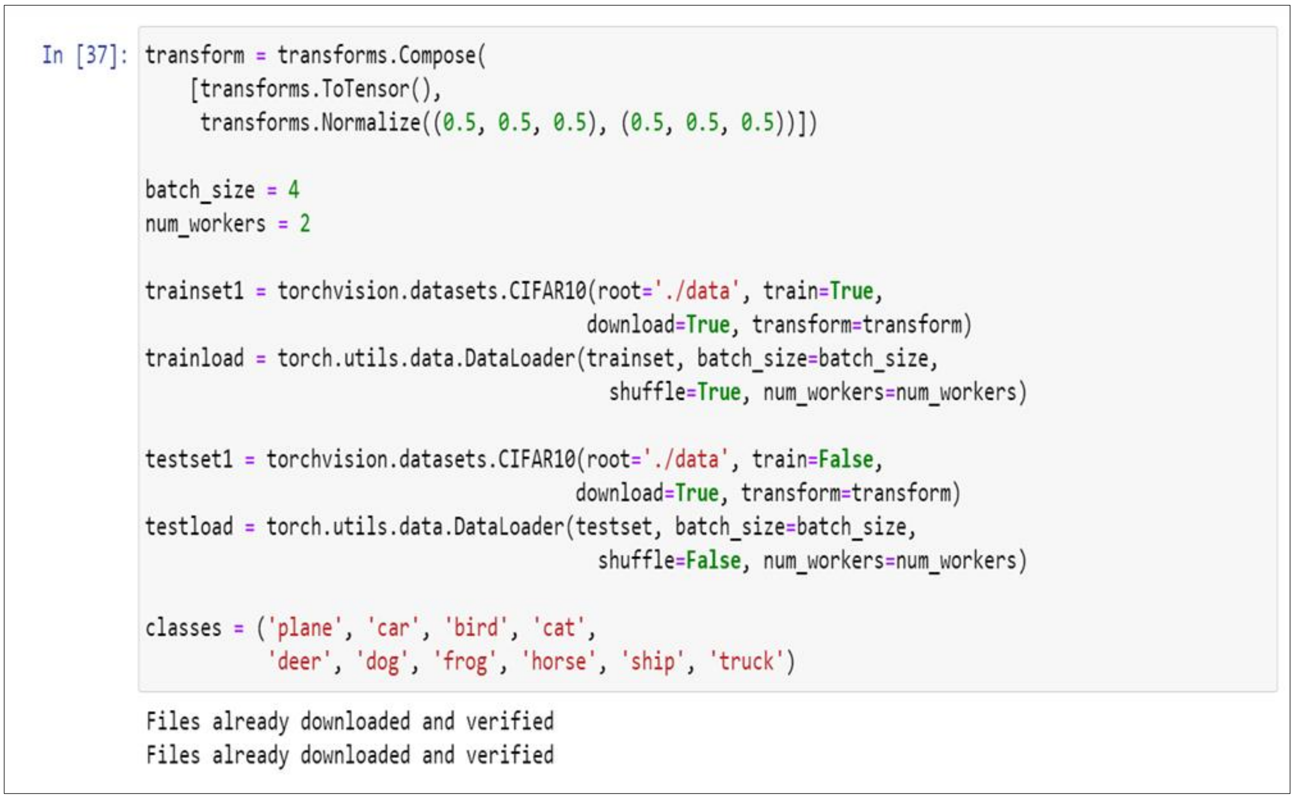


localhost:8888/nbconvert/html/Image%20Classifier.ipynb?download=false

Not syncing

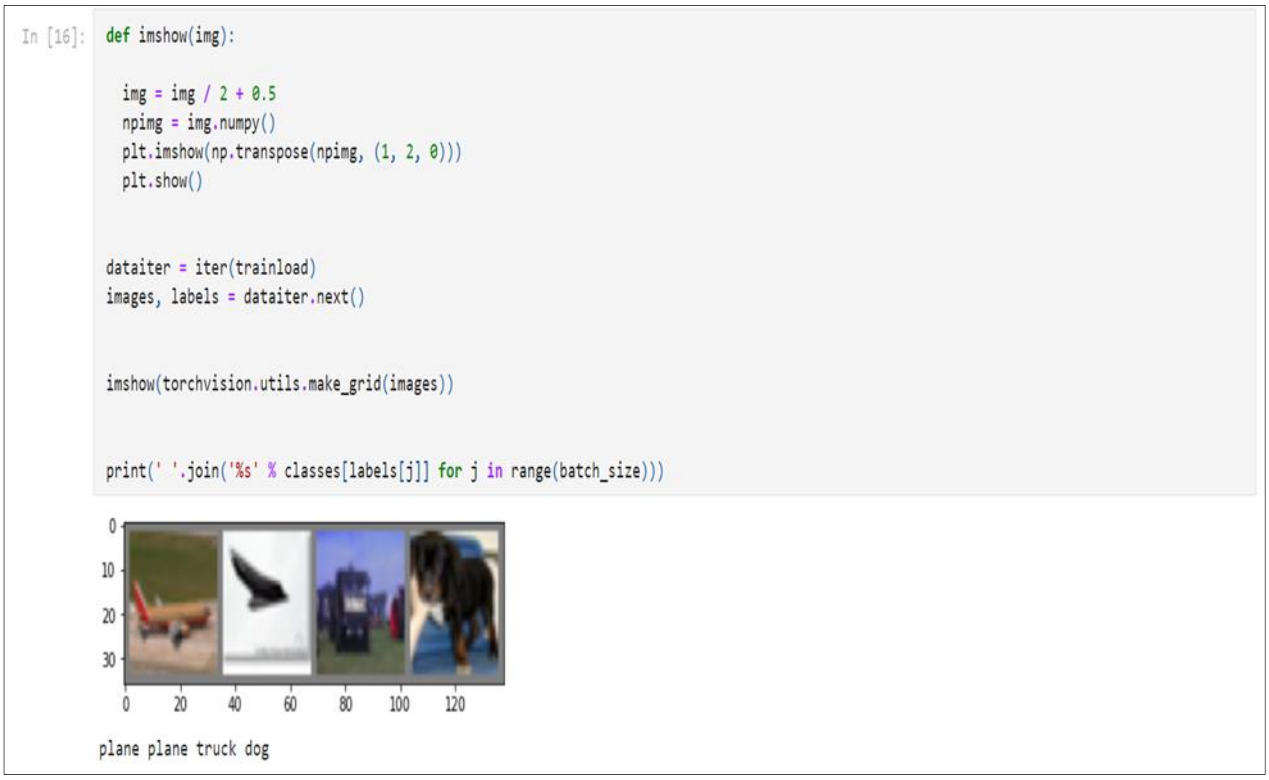
```
In [12]: pip install torch

Requirement already satisfied: torch in c:\users\shrey\anaconda3\lib\site-packages (1.9.1)
Requirement already satisfied: typing-extensions in c:\users\shrey\anaconda3\lib\site-packages (from torch) (3.7.4.3)
Note: you may need to restart the kernel to use updated packages.

In [13]: pip install torchvision

Collecting torchvision
  Downloading torchvision-0.10.1-cp38-cp38-win_amd64.whl (936 kB)
Requirement already satisfied: numpy in c:\users\shrey\anaconda3\lib\site-packages (from torchvision) (1.19.2)
Requirement already satisfied: pillow>=5.3.0 in c:\users\shrey\anaconda3\lib\site-packages (from torchvision) (8.0.1)
Requirement already satisfied: torch==1.9.1 in c:\users\shrey\anaconda3\lib\site-packages (from torchvision) (1.9.1)
Requirement already satisfied: typing-extensions in c:\users\shrey\anaconda3\lib\site-packages (from torch==1.9.1->torchvision) (3.7.4.3)
Installing collected packages: torchvision
Successfully installed torchvision-0.10.1
Note: you may need to restart the kernel to use updated packages.

In [14]: import matplotlib.pyplot as plt
         import numpy as np
         import torch
         import torchvision
         import torchvision.transforms as transforms
         import torch.nn as nn
         import torch.nn.functional as F
         import torch.optim as optim
```

Figure 2 Importing and installing libraries



```
In [37]: transform = transforms.Compose(
             [transforms.ToTensor(),
              transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])

         batch_size = 4
         num_workers = 2

         trainset1 = torchvision.datasets.CIFAR10(root='./data', train=True,
                                                  download=True, transform=transform)
         trainload = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                                   shuffle=True, num_workers=num_workers)

         testset1 = torchvision.datasets.CIFAR10(root='./data', train=False,
                                                 download=True, transform=transform)
         testload = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                                  shuffle=False, num_workers=num_workers)

         classes = ('plane', 'car', 'bird', 'cat',
                    'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

         Files already downloaded and verified
         Files already downloaded and verified
```

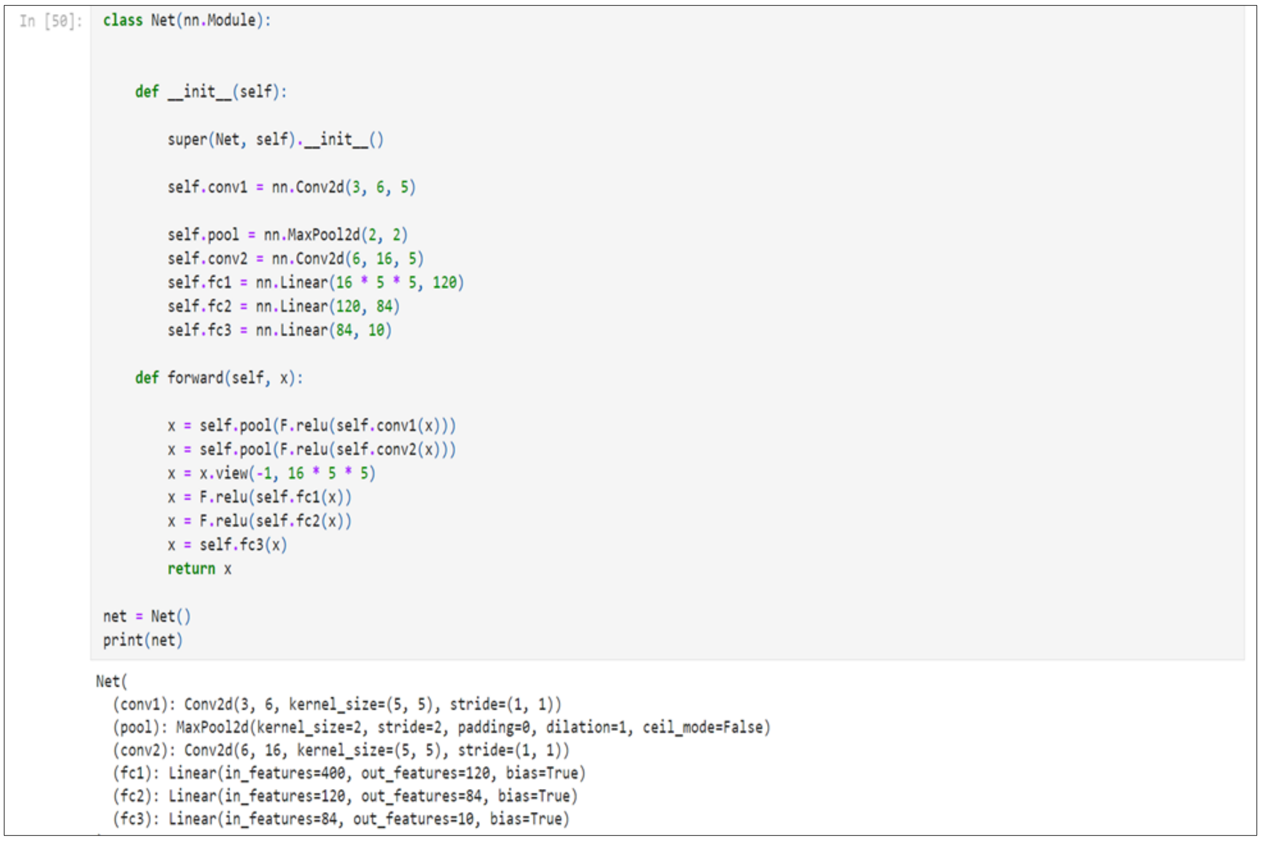
Figure 3 Loading data



```
def imshow(img):

  img = img / 2 + 0.5
  npimg = img.numpy()
  plt.imshow(np.transpose(npimg, (1, 2, 0)))
  plt.show()


dataiter = iter(trainload)
images, labels = dataiter.next()


imshow(torchvision.utils.make_grid(images))


print(' '.join('%s' % classes[labels[j]] for j in range(batch_size)))
```

```
plane plane truck dog
```

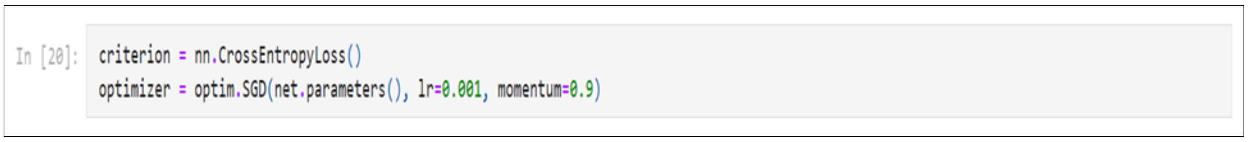
Figure 4 Normalizing the test and training datasets



```
class Net(nn.Module):


    def __init__(self):

        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(3, 6, 5)

        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):

        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = x.view(-1, 16 * 5 * 5)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

net = Net()
print(net)
```

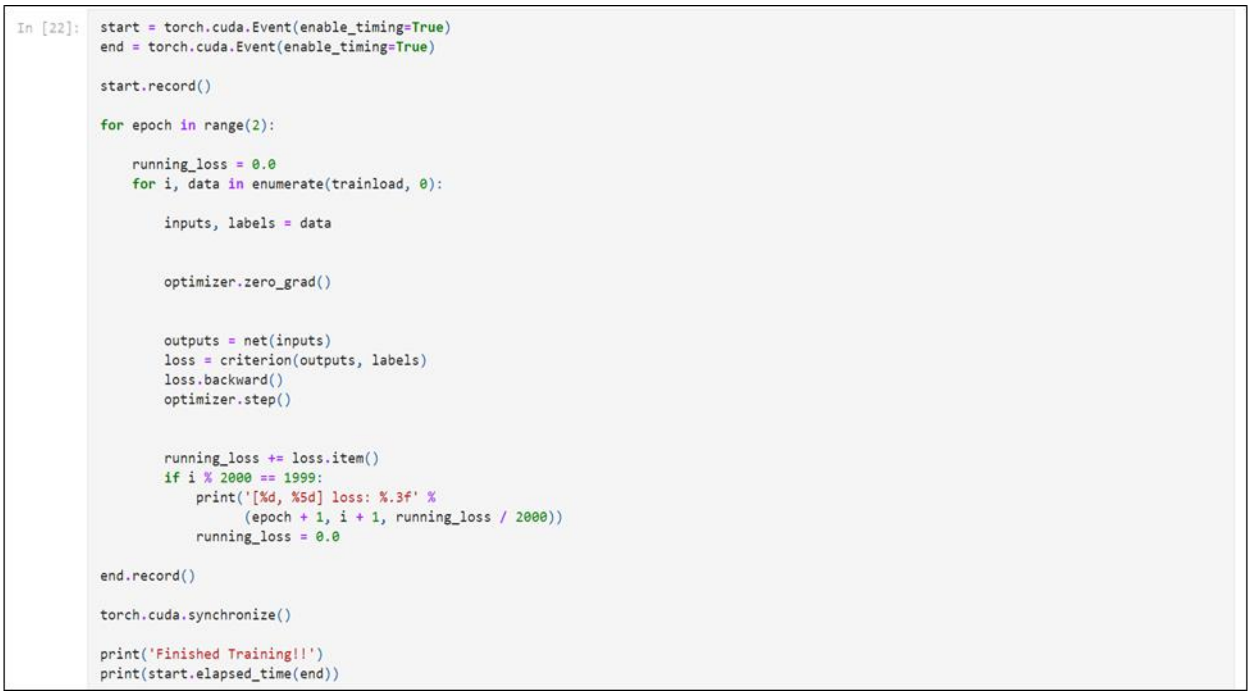
```
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
```

Figure 5 Defining the CNN class



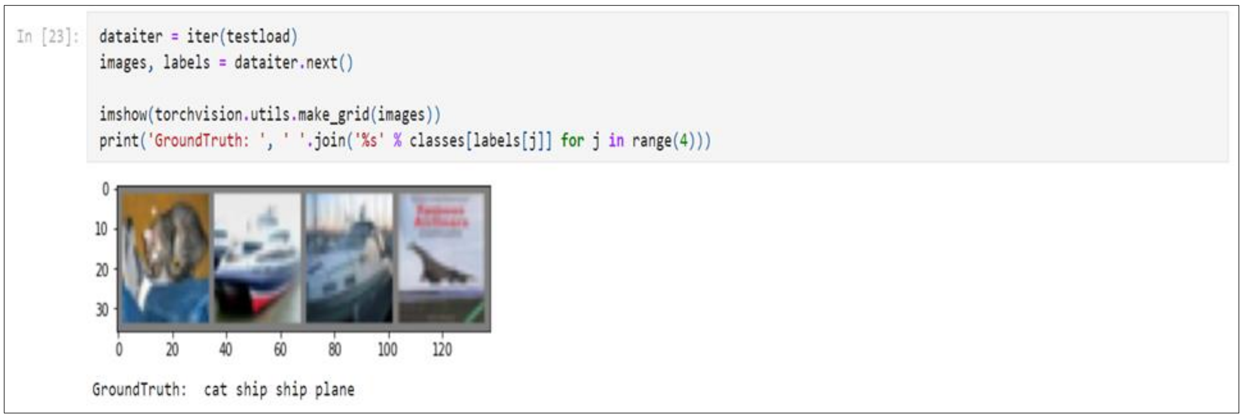
```
In [20]: criterion = nn.CrossEntropyLoss()
         optimizer = optim.SGD(net.parameters(), lr=0.001, momentum=0.9)
```

Figure 6 Defining the optimizer



```
In [22]: start = torch.cuda.Event(enable_timing=True)
         end = torch.cuda.Event(enable_timing=True)

         start.record()

         for epoch in range(2):

             running_loss = 0.0
             for i, data in enumerate(trainload, 0):

                 inputs, labels = data


                 optimizer.zero_grad()


                 outputs = net(inputs)
                 loss = criterion(outputs, labels)
                 loss.backward()
                 optimizer.step()


                 running_loss += loss.item()
                 if i % 2000 == 1999:
                     print('[%d, %5d] loss: %.3f' %
                           (epoch + 1, i + 1, running_loss / 2000))
                     running_loss = 0.0

         end.record()

         torch.cuda.synchronize()

         print('Finished Training!!')
         print(start.elapsed_time(end))
```

Figure 7 Running the loss factor and training the model



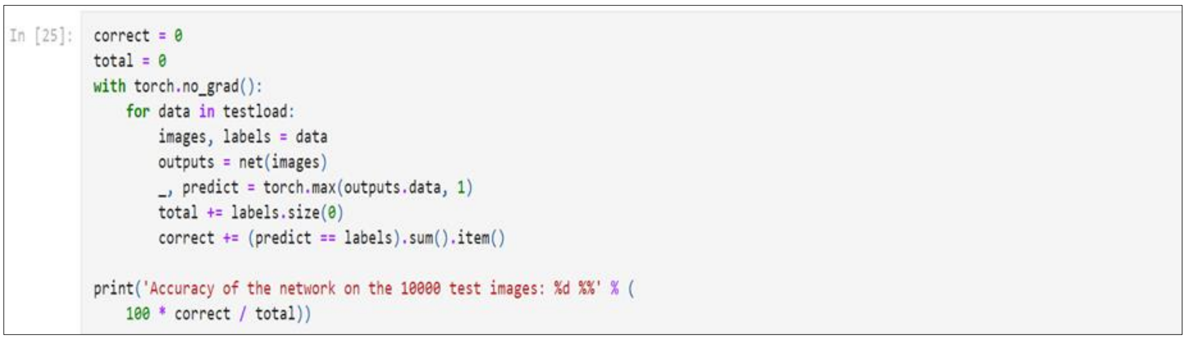
```
In [23]: dataiter = iter(testload)
         images, labels = dataiter.next()

         imshow(torchvision.utils.make_grid(images))
         print('GroundTruth: ', ' '.join('%s' % classes[labels[j]] for j in range(4)))
```

```
GroundTruth:  cat ship ship plane
```

Figure 8 Testing the model

3. Results and Discussion

By the means of PyCharm, an image classifier has been built by using one of the deep learning approaches i.e., convolutional neural networks. In the above model, a dataset called CIFAR 10 has been imported and split into training and test data which thereby was trained using the CNN algorithm to correctly detect and recognize the corresponding object viewed. The accuracy of the network on the 10000 images can be approximated using the below code which was about 75%, relevant enough for a first trial image classifier.



```
In [25]: correct = 0
         total = 0
         with torch.no_grad():
             for data in testload:
                 images, labels = data
                 outputs = net(images)
                 _, predict = torch.max(outputs.data, 1)
                 total += labels.size(0)
                 correct += (predict == labels).sum().item()

         print('Accuracy of the network on the 10000 test images: %d %%' % (
             100 * correct / total))
```

Figure 9 Determining the accuracy of the model

4. Conclusion

This method of detecting images through computer vision in self-driving cars has proven to be much more accurate than regular machine learning outlooks. The deep learning algorithm would perform a classification of the images by the extraction of the features. Whereas manually, a programmer must explicitly meddle in the action for the model to come to a conclusive statement. Although convolutional neural network models seem to showcase great performance, they all have their own consequences to address. For example, during computer vision, when an object viewed from a certain angle slightly different from the way the model has been trained, it may cause false predictions. This may lead the vehicle to end up in taking unnecessary actions during the drive. Alongside the position of the image of the object, ideal lighting as well as colour contrast can also affect the model to flaw. However, this can be solved by adding different variations to the image during the training process which is otherwise known as data augmentation. Nonetheless, this all brings us back to collecting and analysing proper data before feeding it to the training set.

Although the self-driving cars created in today's day to day life have plenty of features, they still have not been developed to its highest extreme. Soon driverless cars will be developed where people can pretty much take a small nap while taking a ride to their destination without worrying about the road.

References

[1] Hornigold, Thomas Building a Moral Machine: Who Decides the Ethics of Self Driving Cars? . Singularity Hub, (31 October 2018).

[2] Dr. Sebastian Raschka Chapter 1: Introduction to Machine Learning and Deep Learning .5 August 2020. Retrieved 28 October 2020.

[3] Phantom Auto will tour city . The Milwaukee Sentinel. 8 December 1926. Retrieved 23 July 2013.

[4] S. Mittal and S. Vaishay A Survey of Techniques for Optimizing Deep Learning on GPUs Archived 2021-05-09 at the Wayback Machine , Journal of Systems Architecture, 2019.

[5] Mesnil, Gregoire; Deng, Li; Gao, Jianfeng; He, Xiaodong; Shen, Yelong Learning Semantic Representations Using Convolutional Neural Networks for Web Search – Microsoft Research , Microsoft Research (April 2014).

[6] Takeo Kanade Three-Dimensional Machine Vision. Springer Science & Business Media. ISBN 978-1-4613-1981-8 (6 December 2012).

[7] Margaret Ann Boden Mind as Machine: A History of Cognitive Science. Clarendon Press. p. 781. ISBN 978-0-19-954316-8 (2006).

[8] Berton cello, Ten ways autonomous driving could redefine the automotive world . McKinsey & Company.