documentation

by Neelima Santoshi

Submission date: 06-Apr-2023 11:10AM (UTC+0530)
Submission ID: 2057318313
File name: Documentation__plagarism.pdf (1.35M)
Word count: 2929
Character count: 16364

ABSTRACT

Recycling is already a significant work for all countries. For developing country like India, the major problem is about the waste management. Among the work needed for recycling, garbage classification is the most fundamental step to enable cost-efficient recycling. This paper aims to address the importance of garbage classification as a crucial step towards efficient recycling. The study focuses on identifying and categorizing individual garbage objects in images using different approaches such as ResNet-50 neural network architecture. The dataset used in this study consists of 2467 images from six different classes of waste. The proposed system can effectively solve garbage classification problems in the targeted database, thanks to the utilization of deep learning techniques. The results of this study contribute to the development of an automated garbage classification system that can enhance recycling efficiency.

INTRODUCTION

• MOTIVATION:

Our motivation for research can help to prevent Environment impact, Resource conservation and promotes sustainability, conserve resources, protect public health, and ensure regularity compliance.

• SCOPE:

Our scope is to develop the model which helps us to detect classes of the image from the dataset.

• PROBLEM:

The Problem with previous one is that the performance is low when it comes to the larger datasets.

• SOLUTION:

The solution for this project is to build a model for the Garbage Classification using more images containing in a data set. This helps for the model to be efficient and the performance will be increased in real time.

• PROBLEM DEFINITION

In this project, we try to implement a Garbage Classification model that helps to identify or classify the images from the dataset. The necessary details regarding the dataset are: dataset provides the images which consists of 2467 images that classifies the cardboard, glass, metal, paper, plastic.

• LIMITATIONS

Classification can be done through the images.

LITERATURE SURVEY

INTRODUCTION:

Waste materials are sorted and categorised according to their kinds as part of waste management in order to minimise environmental contamination and increase recycling efforts. Convolutional neural networks (CNNs) have become a potent tool for image classification problems, particularly trash classification, with the introduction of deep learning.

The literature review has looked at different CNN architectures and approaches to increase the accuracy of garbage classification, according to a literature review on the subject.

"Garbage Classifying Application Using Deep Learning Techniques" by Arpit Patel ,2021.in these three different models have been tested for higher accuracy. For trash classification CNN and vgg16 having accuracy of 90.6% and 93.46%.

"A Study of Garbage Classification with Convolutional Neural Networks" by Shanshen Meng & Wei-Ta-Chu(2020) suggested a CNN and Convolution neural network architectures have been measured and compared on the based model and having a dataset of different categories with having an test accuracy of 95.35%.

For instance, Zhang et al. (2019) suggested a CNN-based model in their article titled "Garbage Classification Based on CNN and Transfer Learning" that makes use of transfer learning from previously trained models to increase classification accuracy. For a trash dataset of six categories, the model had an accuracy of 94.26%.

According to the literature review, CNN architecture models have produced best results for waste classification. The accuracy and effectiveness of trash classification models can be considerably impacted by the CNN architecture (Resnet, denset) and training methods chosen from that is resnet. There is a rising opportunity for the creation of more precise and effective waste classification models to aid waste management efforts, given the growing accessibility of garbage data and developing in deep learning.

Proposed System:

From the basis of the literature survey the model developed on the garbage classification using convolutional neural network architecture. The architecture used for training the data is resnet or resnet50.We developed a model base using residual network having an accuracy of 96.18% by using the functions pytorch, torchvision. The dataset is taken from Kaggle which is differentiated into six categories which consist of 2527 images in the given dataset.

Advantage of Proposed System:

Our model performance is faster comparative to the previous one.

PROBLEM IDENTIFICATION AND OBJECTIVES

PROBLEM IDENTIFICATION:

In this problem identification Model takes huge amount of time when it comes to large amount of data to be trained, by using the algorithms such as resnet or resnet50. we cannot be trained with huge number of images in a dataset quickly and it takes more time for training the model and also gives less performance for low number of epochs. Garbage classification on images involves identifying classes, whether they are cardboard, glass, metal, paper, and plastic.

OBJECTIVES:

The garbage classifier's goal is to use computer vision and deep learning techniques to precisely recognize and classify individual garbage objects into their corresponding recycling categories. The method intends to address the intricate issue of managing solid waste in big cities, where the rising volume of waste produced daily by people and businesses has made waste disposal extremely difficult. The automatic detection and classification of waste types by the garbage classifier can make recycling and waste management procedures more effective. Accurate and automatic classification of trash objects is possible with the aid of machine learning approaches like support vector machines (SVM) with HOG features, simple convolutional neural networks (CNN), and CNN with residual blocks.

SYSTEM METHODOLOGY

Introduction:

Design is crucial for knowing or understanding the project implementation and the way project operates, as well as for making it simple for others to understand. UML diagrams are being used to describe the working process here.

Purpose:

The feasibility, structure and affordability of the project with expectations of stakeholders are all ensured through effective design. It helps the project team to spot possible risks, reduce them, make the most of their resources. The purpose for the design documentation stores the record of the decisions, implementation details. The object-oriented design document's goal is to give an description of how the suggested system would be utilized to acquire the in required data to determine the best way to deploy our system.

Design Goals:

The goals of designing are to create high-quality models of systems and to demonstrate how systems appear after the coding is accomplished. It gives the programmer the ability to successfully complete his or her work as a developer and create good, error-free systems; these also enable the developer to create systems that are simple to maintain. This makes it simple for maintenance to modify the system once it has been placed into operations.

Data Flow Diagram:

Data flow Diagram (DFD) inside a system is graphically represented by a data flow diagram (DFD). It is a modelling approach that enables you to see how data is transferred across various system components, including inputs, processes, outcomes, and storage.

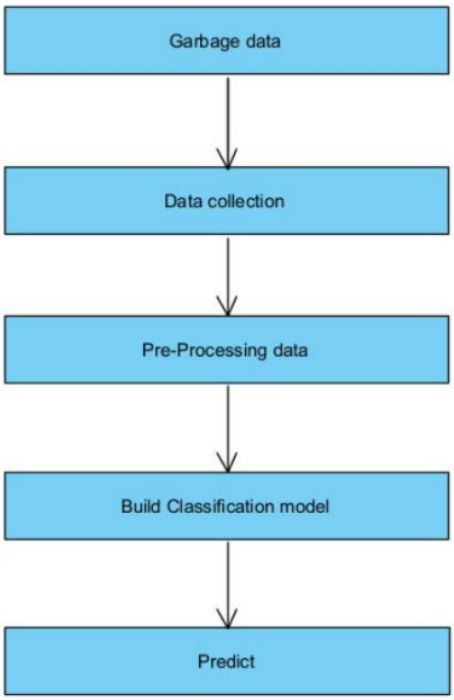
A DFD generally consists of the following four elements:

(i)sources and recipients of data: They are the outside parties that communicate with the system and either transfer or send data from it.
(ii)Processes: These are the actions that the system uses to alter data. Processes can accept inputs, apply logic or rules, and generate outputs.
(iii)Data flows are the routes that data travels on when it is processed by the system. The transfer of data between various processes or between procedures and external entities can be represented by data flows.
(iv)Data stores: These are the locations in the system where data is kept. Databases, files, and other permanent storage types can all be represented as data stores.

An existing system may be analyzed or a new one can be designed using a DFD. Identification of the data passing through the system and the connections between the various parts is helpful. You can find possible issues and areas for improvement by disassembling the system into smaller parts and studying how data transfers between them.

DFDs are frequently used in software engineering to design new systems or to describe the specifications of existing systems. They may be used to describe business processes and workflows in business analysis as well.

Multiple levels of information are included in DFDs, ranging from high-level diagrams that depict the overall data flow to more complex diagrams that reveal the inner workings of certain processes. It's critical to select the amount of detail that best suits your demands and confirms that the diagram truly depicts the system being represented.



UML Diagram:

A software system is represented using the graphical modelling language known as Unified Modeling Language (UML). The many components of a software system are modelled using UML diagrams, which also serve as a visual representation of the system's architecture, design, and implementation. Software engineers frequently use UML to better understand and explain the many components of a system.

The two types of UML diagrams are structural diagrams and behavioral diagrams.

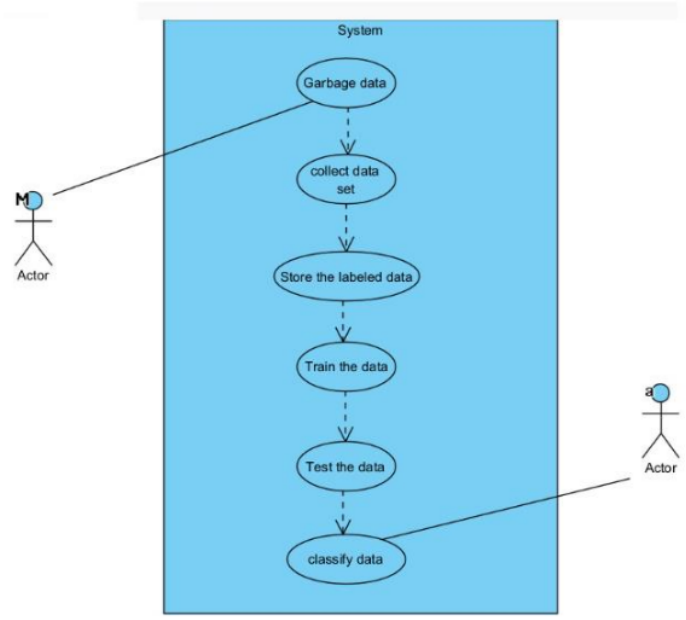
Structural Diagrams: These diagrams show the system's static structure and the connections between its parts. The various types of structural diagrams consist of:
Class Diagrams, Object, Component, Deployment and Package Diagram.

Behavioral Diagram: Diagrams showing the system's dynamic behavior and how it reacts to outside inputs are known as behavioral diagrams. The many forms of behavioral diagrams consist of: use case diagram, activity diagram, sequence diagram, state, and communication.

Project managers, clients, and developers may all benefit from understanding complex systems and processes through the use of UML diagrams. They assist in making sure that everyone understands the system being designed and may be used to spot possible problems or areas that could have improvement.

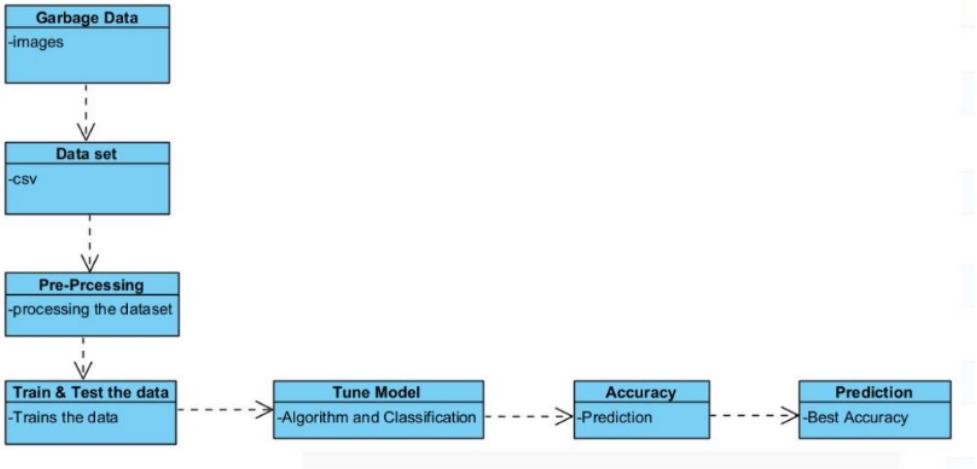
Use case:

use-cases help identify the primary requirements of the system. use-cases are used to ensure that the evolving design is always relevant to what the user required. Indeed, the use cases act as the one consistent thread throughout the whole of the development process. For example, at the beginning of the design phase, one of the two primary inputs to this phase is the use-case model. Then explicitly within the design model are use-case realizations that illustrate how each use-case is supported by the designated.



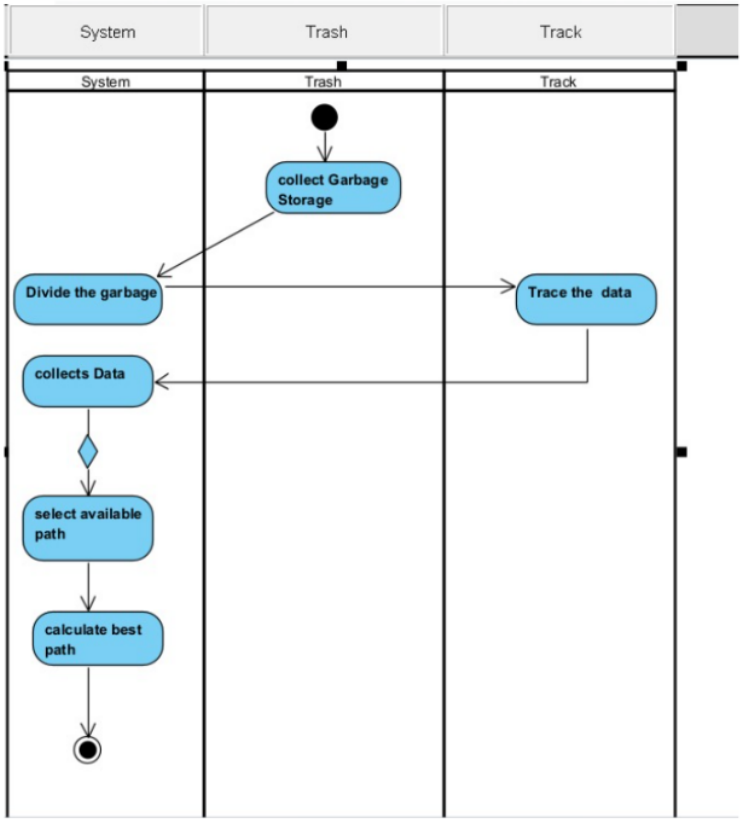
Class diagram:

Creating of the class diagram helps to understand the structure and their sequence to consider system behavior and data models to consider the implication on a database. It also shows their behavior of the one or more classes.



Activity Diagram:

The flow of activities or actions inside a system is modelled using activity diagrams, a form of UML diagram. They are frequently utilized to represent complicated systems, software operations, and business processes. Activity diagrams may be used at any stage of the software development lifecycle, from requirements gathering to design and testing, and are a very effective tool for comprehending a system's behavior.

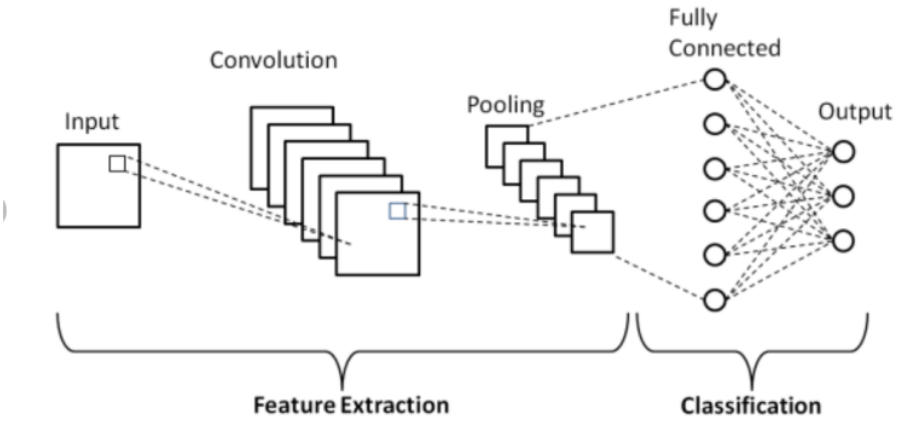

System
Trash
Track
System
Trash
Track
collect Garbage Storage
Divide the garbage
Trace the data
collects Data
select available path
calculate best path

OVERVIEW OF TECHNOLOGIES

CNN:

CNNs are a deep learning method commonly used in image categorization, object identification, and other object recognition applications. They are based on the convolution idea, which comprises extracting characteristics from an input picture using a collection of filters.

CNNs are composed of numerous layers, each of which serves a distinct purpose. The input layer receives visual data, and the convolutional layers extract picture properties like as edges, textures, and forms via a series of filters. One or more pooling layers then down sample and dimensionally lower the output of the convolutional layers. The output is then routed through one or more entirely linked layers.



CNNs can automatically learn and extract features from input photos, eliminating the requirement for feature creation. This makes them particularly useful for tasks such as object identification and recognition, when the qualities of interest are complicated and difficult to identify manually.

Another feature of CNNs is its capacity to handle pictures of varied sizes and resolutions through the use of techniques such as pooling and stride. As a result, they are extremely flexible to a wide range of applications.

CNNs have gained popularity in recent years because to their great accuracy and performance in a variety of computer vision applications. They've been employed in self-driving vehicles and face recognition, among other things.

RESIDUAL NETWORK:

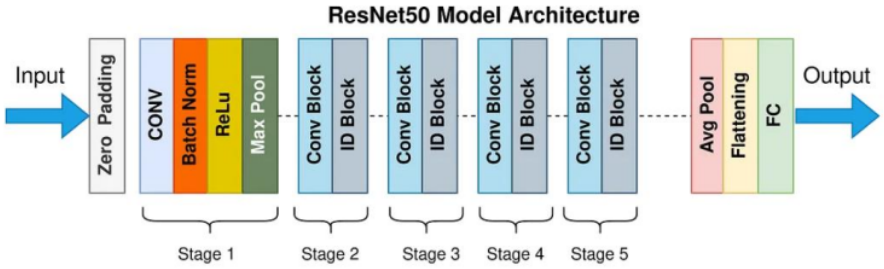
Resnet (short for Residual Network) is a group of neural network topologies developed in 2015 to solve the issue of disappearing gradients in very deep neural networks. The Resnet model is based on the concept of residual learning, which enhances creating shortcut connections that escape one or more network layers.

The ResNet50 model is a 50-layer variation of the Resnet model. The ResNet50 design is composed of many building elements known as residual blocks. Each residual block has two or three convolutional layers as well as a shortcut connection that skips one or more convolutional layers. The shortcut link enables the network to learn identity mappings and retrieve lost data, potentially improving accuracy.

The ResNet50 model is often trained on a large dataset of labelled pictures, such as the ImageNet dataset, using supervised learning. During training, the model adjusts the weights of the convolutional layers to learn to detect patterns in the input pictures. Training entails submitting batches of pictures to the network iteratively, determining the loss (measuring of the difference between anticipated and real labels), and changing the weights via backpropagation.

After training, the ResNet50 model may be used for picture classification, object recognition, and other computer vision applications. An input picture is transmitted through the layers of the network with the weights fixed throughout inference (when the model is used to generate predictions on fresh, unseen data). The network's last layer generates a probability distribution across all potential classes, and also the class with highest probability is chosen as the image's expected label.

Overall, the ResNet50 model is very effective tool for computer vision problems, and it has the capacity to generate deep representations via residual connections and had a significant effect on the deep learning field.



IMPLEMENTATION

CODING:

Below line of codes imports the necessary libraries, including OS, torch, and torchvision, for building and training deep learning models for image classification tasks.

The os library provides a way of using operating system dependent functionality like reading or writing to the file system.

torch library is the primary library for building and training deep learning models in Python.

torchvision library provides datasets, transforms, and models specific to computer vision tasks.

random_split is a function from the torch.utils.data module that is used to split a dataset into random non-overlapping subsets.

models is a sub-module of torchvision that contains pre-trained models for various computer vision task.

nn is a sub-module of torch that contains various neural network modules, such as layers, loss functions.

```
import os
import torch
import torchvision
from torch.utils.data import random_split
import torchvision.models as models
import torch.nn as nn
import torch.nn.functional as F
```

Transformations:

Here we are using **data_dir** variable to locate the data set file and then using **torchvision** modules to resize the images present in datasets

```
data_dir  = "C:/Users/Kailash/Downloads/archive/Garbageclassification/Garbageclassification"

classes = os.listdir(data_dir)
print(classes)
```

```
#Resizing image
from torchvision.datasets import ImageFolder
import torchvision.transforms as transforms

transformations = transforms.Compose([transforms.Resize((256, 256)), transforms.ToTensor()])

dataset = ImageFolder(data_dir, transform = transformations)
```

To see the image

```
import matplotlib.pyplot as plt
%matplotlib inline

def show_sample(img, label):
    print("Label:", dataset.classes[label], "(Class No: "+ str(label) + ")")
    plt.imshow(img.permute(1, 2, 0))
```

```
img, label = dataset[10]
show_sample(img, label)
```

```
Label: cardboard (Class No: 0)
```

Splitting Data:

```
#loading and splitting the data
random_seed = 42
torch.manual_seed(random_seed)
```

```
train_ds, val_ds, test_ds = random_split(dataset, [1593, 176, 758])
len(train_ds), len(val_ds), len(test_ds)
```

```
#Training and Validation part
train_dl = DataLoader(train_ds, batch_size, shuffle = True, num_workers = 4, pin_memory = True)
val_dl = DataLoader(val_ds, batch_size*2, num_workers = 4, pin_memory = True)
```

training and validation :

```
#Training and Validation part
train_dl = DataLoader(train_ds, batch_size, shuffle = True, num_workers = 4, pin_memory = True)
val_dl = DataLoader(val_ds, batch_size*2, num_workers = 4, pin_memory = True)
```

Used to visualize data :

```
from torchvision.utils import make_grid

def show_batch(dl):
    for images, labels in dl:
        fig, ax = plt.subplots(figsize=(12, 6))
        ax.set_xticks([])
        ax.set_yticks([])
        ax.imshow(make_grid(images, nrow = 12).permute(1, 2, 0))
        break
```

```
show_batch(train_dl)
```

Model Creation:

```
def accuracy(outputs, labels):
    _, preds = torch.max(outputs, dim=1)
    return torch.tensor(torch.sum(preds == labels).item() / len(preds))

class ImageClassificationBase(nn.Module):
    def training_step(self, batch):
        images, labels = batch
        out = self(images)
        loss = F.cross_entropy(out, labels)
        return loss

    def validation_step(self, batch):
        images, labels = batch
        out = self(images)
        loss = F.cross_entropy(out, labels)
        acc = accuracy(out, labels)
        return {'val_loss': loss.detach(), 'val_acc': acc}

    def validation_epoch_end(self, outputs):
        batch_losses = [x['val_loss'] for x in outputs]
        epoch_loss = torch.stack(batch_losses).mean()   # Combine losses
        batch_accs = [x['val_acc'] for x in outputs]
        epoch_acc = torch.stack(batch_accs).mean()      # Combine accuracies
        return {'val_loss': epoch_loss.item(), 'val_acc': epoch_acc.item()}

    def epoch_end(self, epoch, result):
        print("Epoch {}: train_loss: {:.4f}, val_loss: {:.4f}, val_acc: {:.4f}".format(
            epoch+1, result['train_loss'], result['val_loss'], result['val_acc']))
```

By using ResNet50:

```
class ResNet(ImageClassificationBase):
    def __init__(self):
        super().__init__()
        # Use a pretrained model
        self.network = models.resnet50(pretrained=True)
        # Replace last layer
        num_ftrs = self.network.fc.in_features
        self.network.fc = nn.Linear(num_ftrs, len(dataset.classes))

    def forward(self, xb):
        return torch.sigmoid(self.network(xb))

model = ResNet()
```

```
def get_default_device():
    """Pick GPU if available, else CPU"""
    if torch.cuda.is_available():
        return torch.device('cuda')
    else:
        return torch.device('cpu')

def to_device(data, device):

    if isinstance(data, (list,tuple)):
        return [to_device(x, device) for x in data]
    return data.to(device, non_blocking=True)

class DeviceDataLoader():

    def __init__(self, dl, device):
        self.dl = dl
        self.device = device

    def __iter__(self):

        for b in self.dl:
            yield to_device(b, self.device)

    def __len__(self):
        """Number of batches"""
        return len(self.dl)
```

```
device = get_default_device()
device
```

```
train_dl = DeviceDataLoader(train_dl, device)
val_dl = DeviceDataLoader(val_dl, device)
to_device(model, device)
```

```
class ResNet(ImageClassificationBase):
    def __init__(self):
        super().__init__()
        # Use a pretrained model
        self.network = models.resnet50(pretrained=True)
        # Replace last layer
        num_ftrs = self.network.fc.in_features
        self.network.fc = nn.Linear(num_ftrs, len(dataset.classes))

    def forward(self, xb):
        return torch.sigmoid(self.network(xb))

model = ResNet()
```

Training the Model:

```
@torch.no_grad()
def evaluate(model, val_loader):
    model.eval()
    outputs = [model.validation_step(batch) for batch in val_loader]
    return model.validation_epoch_end(outputs)

def fit(epochs, lr, model, train_loader, val_loader, opt_func=torch.optim.SGD):
    history = []
    optimizer = opt_func(model.parameters(), lr)
    for epoch in range(epochs):

        model.train()
        train_losses = []
        for batch in train_loader:
            loss = model.training_step(batch)
            train_losses.append(loss)
            loss.backward()
            optimizer.step()
            optimizer.zero_grad()

        result = evaluate(model, val_loader)
        result['train_loss'] = torch.stack(train_losses).mean().item()
        model.epoch_end(epoch, result)
        history.append(result)
    return history
```

```
model = to_device(ResNet(), device)
```

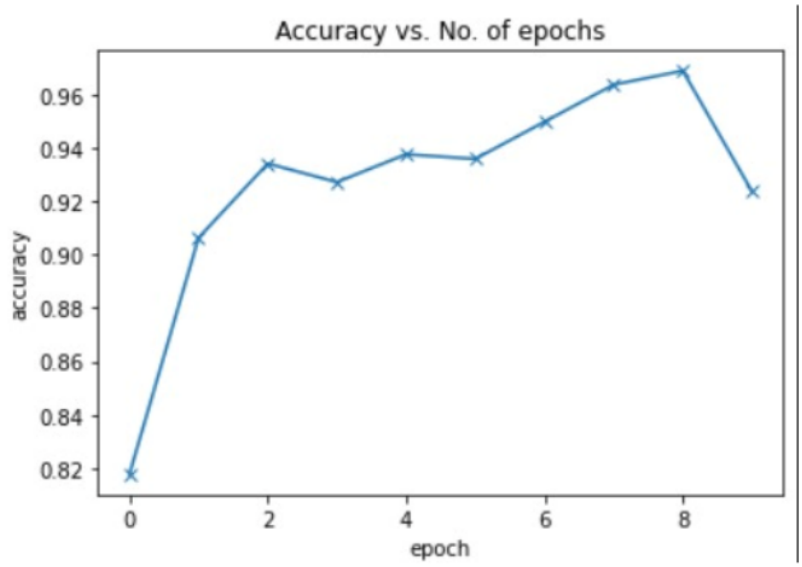
```
evaluate(model, val_dl)
```

Trains the model:

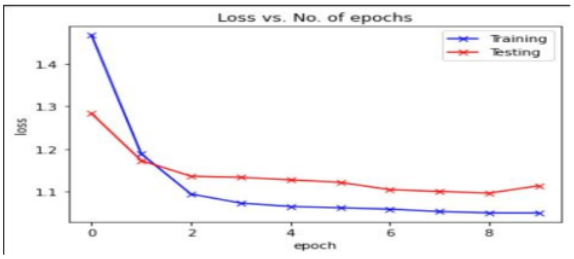
```
num_epochs = 10
opt_func = torch.optim.Adam
lr = 5.5e-5

history = fit(num_epochs, lr, model, train_dl, val_dl, opt_func)
```

```
Epoch 1: train_loss: 1.4668, val_loss: 1.2835, val_acc: 0.8177
Epoch 2: train_loss: 1.1888, val_loss: 1.1730, val_acc: 0.9062
Epoch 3: train_loss: 1.0946, val_loss: 1.1364, val_acc: 0.9340
Epoch 4: train_loss: 1.0733, val_loss: 1.1336, val_acc: 0.9271
Epoch 5: train_loss: 1.0653, val_loss: 1.1278, val_acc: 0.9375
Epoch 6: train_loss: 1.0623, val_loss: 1.1221, val_acc: 0.9358
Epoch 7: train_loss: 1.0591, val_loss: 1.1048, val_acc: 0.9497
Epoch 8: train_loss: 1.0535, val_loss: 1.1007, val_acc: 0.9635
Epoch 9: train_loss: 1.0505, val_loss: 1.0966, val_acc: 0.9688
Epoch 10: train_loss: 1.0501, val_loss: 1.1139, val_acc: 0.9236
```

```
def   (variable) accuracies: list
    accuracies = [x['val_acc'] for x in history]
    plt.plot(accuracies, '-x')
    plt.xlabel('epoch')
    plt.ylabel('accuracy')
    plt.title('Accuracy vs. No. of epochs');

plot_accuracies(history)
```




```
def plot_losses(history):
    train_losses = [x.get('train_loss') for x in history]
    val_losses = [x['val_loss'] for x in history]
    plt.plot(train_losses, '-bx')
    plt.plot(val_losses, '-rx')
    plt.xlabel('epoch')
    plt.ylabel('loss')
    plt.legend(['Training', 'Testing'])
    plt.title('Loss vs. No. of epochs');

plot_losses(history)
```

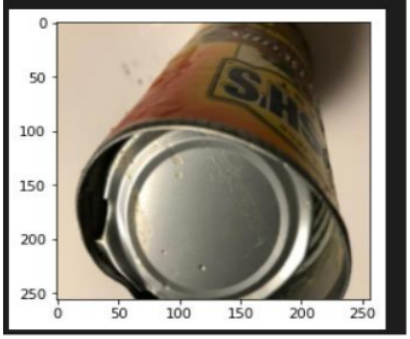


Classifying the Image:

```
def predict_image(img, model):

    xb = to_device(img.unsqueeze(0), device)

    yb = model(xb)

    prob, preds  = torch.max(yb, dim=1)

    return dataset.classes[preds[0].item()]
```

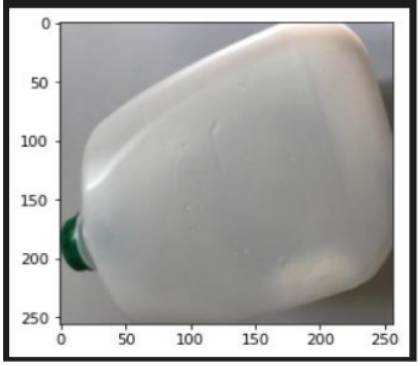
```
img, label = test_ds[17]
plt.imshow(img.permute(1, 2, 0))
print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))
```

```
Label: metal , Predicted: metal
```

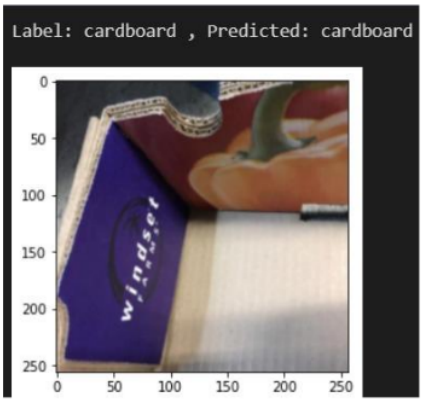



```
img, label = test_ds[45]
plt.imshow(img.permute(1, 2, 0))
print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))
```

```
Label: plastic , Predicted: plastic
```




```
img, label = test_ds[54]
plt.imshow(img.permute(1, 2, 0))
print('Label:', dataset.classes[label], ', Predicted:', predict_image(img, model))
```


Label: cardboard , Predicted: cardboard
0
50
100
150
200
250
0
50
100
150
200
250

RESULTS & DISCUSSIONS

Accuracy Measure	Value
Accuracy	96.88%

Table: Performance Measure

Model is based on the garbage classification using resnet having accuracy of 96.88%.

CONCLUSION

The Garbage classification could be a little tedious process using CNN, so we developed a Resnet based model for trash image classification and predicted classes for our dataset, so we utilized a pretrained ResNet50 model with a linear layer. We utilized data loaders to load and change the data for the model after dividing the dataset into training, validation, and testing sets. The Resnet class inherits from our Image Classification Base class, which contains the training and validation procedure. By finding accuracy and assess the model's performance during the training, we built helper functions and using data from the validation set, we trained the model and assessed it. For the validation set, the model processed with a good accuracy, and we may use it to determine classes from garbage images.

Future Scope:

Garbage classification using deep learning techniques has future scope of classifying trash images more accurately. This will help us to classify any image especially given in dataset.

REFERENCES

1. Arpit Patil, Anish Tatke, Nancy vachhani, Madura Patil and Priyanka Gulhane (Assistant Professor) developed a model on "Garbage Classifying Application Using Deep Learning Techniques" under Dept. Computer science and Engineering, MIT-WPU, Pune, India (2021).

2. Shanshan Meng, Wei-Ta chu developed a model on "A Study of Garbage Classification with Convolution Neural Networks" under Humbolt University of Berlin, Germany & National Cheng Kung University, Taiwan (2020).

3. K srivatsan, Surender Dhiman, Anuj Jain developed a model on "Waste Classification using Transfer learning and Convolutional Neural Networks" under Dept. Electronics and Communication Department, Lovely Professional University, India (2021).

4. Alex Krizhevsky, ilya Sutskever, Geoffrey E. Hinton developed a model on "ImageNet Classification with Deep Convolutional Neural Networks" under University of Toronto, USA (2012).

5. S. Kaza, L. Yao, P. Bhada-Tata, and F. Van Woerden," What a Waste 2.0: A Global Snapshot of Solid Waste Management to 2050". The World Bank, 2018.

6. H. Wang, "Garbage Recognition and Classification System Based on Convolutional Neural Network VGG16," 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), April 2020.

7. B. Gan and C. Zhang, "Research on the algorithm of urban waste classification and recycling based on deep learning technology," 2020 International Conference on Computer Vision, Image and Deep Learning, July 2020.

8. Shuang Wu, Zeyu Li, Xinxiang Chen, Peiwen Zhong, Xing Cai, done survey on "Garbage Classification Problem Based on Convolutional Neural Network" under Dept. of Mathematics and Civil Engineering, Zhuhai Campus, Beijing Institute of Technology, Zhuhai, China (2021).

9. Ishika Mittal, Anjali Tiwari, Bhoomika Rana, Pratibha Singh, developed a model on "Trash Classification: Classifying garbage using Deep Learning" under Krishna engineering college, India (2020).

documentation

ORIGINALITY REPORT

16%	12%	12%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Arpit Patil, Anish Tatke, Nancy Vachhani, Madhura Patil, Priyanka Gulhane. "Garbage Classifying Application Using Deep Learning Techniques", 2021 International Conference on Recent Trends on Electronics, Information, Communication & Technology (RTEICT), 2021 Publication	3%
2	www.docme.ru Internet Source	2%
3	John Hunt. "The Unified Process", Guide to C# and Object Orientation, 2002 Publication	1%
4	Submitted to University of Wales Institute, Cardiff Student Paper	1%
5	eprints.whiterose.ac.uk Internet Source	1%
6	mmcv.csie.ncku.edu.tw Internet Source	1%

7	Shuang Wu, Zeyu Li, Xinqiong Chen, Peiwen Zhong, Liangcai Mei, Xing Cai. "Research on the Garbage Classification Problem Based on Convolutional Neural Network", Journal of Physics: Conference Series, 2021 Publication	1 %
8	Submitted to King's Own Institute Student Paper	1 %
9	Submitted to Infile Student Paper	1 %
10	Submitted to West Herts College Student Paper	1 %
11	9dok.org Internet Source	1 %
12	Haruna Abdu, Mohd Halim Mohd Noor. "Domestic Trash Classification with Transfer Learning Using VGG16", 2022 IEEE 12th International Conference on Control System, Computing and Engineering (ICCSCE), 2022 Publication	1 %
13	Submitted to University of Sydney Student Paper	<1 %
14	paper.ijcsns.org Internet Source	<1 %
15	ieeexplore.ieee.org Internet Source	<1 %

16	web.archive.org Internet Source	<1 %
17	Chinmai Shetty, Pratheeksha Hegde N, Dhananjaya B, Deepa, Rashmi N, Sarojadevi H, Anusha Shenoy. "noTrash.AI – An Approach Towards the Hygienic City based on Deep Learning", 2022 6th International Conference on Intelligent Computing and Control Systems (ICICCS), 2022 Publication	<1 %
18	Submitted to Berlin School of Business and Innovation Student Paper	<1 %
19	exaly.com Internet Source	<1 %
20	www.ijert.org Internet Source	<1 %
21	www.researchgate.net Internet Source	<1 %

Exclude quotes	Off	Exclude matches	Off
Exclude bibliography	Off		