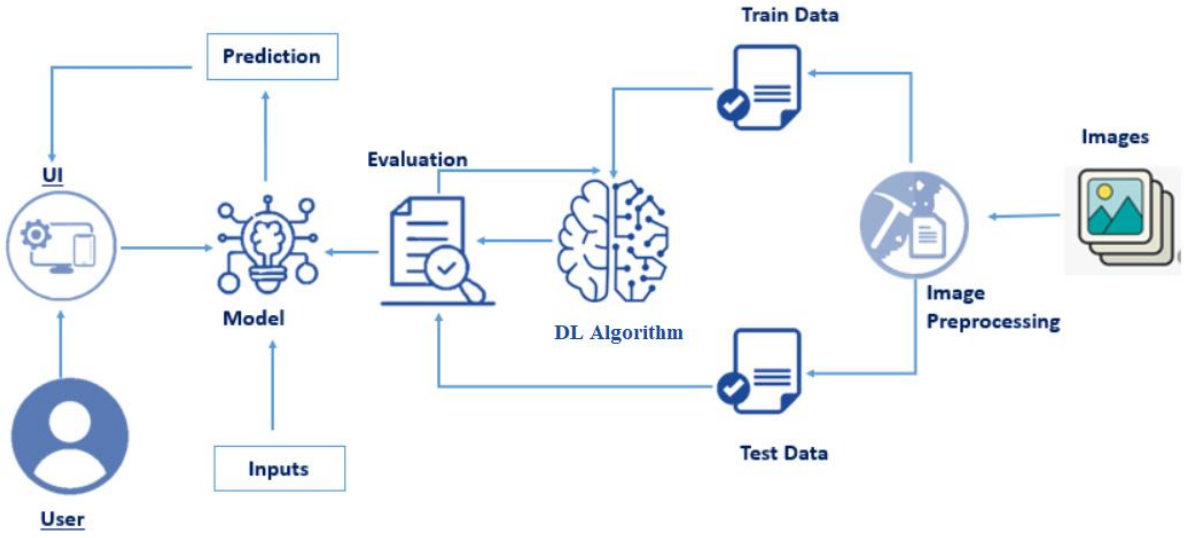
Intelligent Garbage Classifcation Using Deep Learning

According to the next 25 years, the less developed countries' waste accumulation will increase drastically. With the increase in the number of industries in the urban area,the disposal of the solid waste is really becoming a big problem, and the solid waste includes paper, wood, plastic, metal, glass etc. The common way of managing waste is burning waste and this method can cause air pollution and some hazardous materials from the waste spread into the air which can cause cancer. Hence it is necessary to recycle the waste to protect the environment and human beings' health, and we need to separate the waste into different components which can be recycled using different ways.

The present way of separating waste/garbage is the hand-picking method, whereby someone is employed to separate out the different objects/materials. The person who separates waste, is prone to diseases due to the harmful substances in the garbage. With this in mind, it motivated us to develop an automated system which is able to sort the waste. and this system can take a short time to sort the waste, and it will be more accurate in sorting than the manual way. With the system in place, the beneficial separated waste can still be recycled and converted to energy and fuel for the growth of the economy. The system that is developed for the separation of the accumulated waste is based on the combination of Convolutional Neural Network

Technical Architecture:



Python packages:

NumPy: NumPy is a Python package that stands for 'Numerical Python. It is the core library for scientific computing, which contains a powerful n-dimensional array of objects.

Pandas: pandas is a fast, powerful, flexible, and easy-to-use open-source data analysis and manipulation tool, built on top of the Python programming language.

Matplotlib: It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits

Keras: Keras is an open-source library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3, Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, R, Theano, and PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

TensorFlow: TensorFlow is just one part of a much bigger, and growing ecosystem of libraries and extensions that help you accomplish your machine learning goals. It is a free and open-source software library for data flow and differentiable programming across a range of tasks. It is a symbolic math library and is also used for machine learning applications such as neural networks.

Flask: Web framework used for building Web applications

Project Flow

- The user interacts with the UI (User Interface) to choose the image.
- The chosen image analyzed by the model which is integrated with flask application.
- CNN Models analyze the image, then prediction is showcased on the Flask UI.

To accomplish this, we have to complete all the activities and tasks listed below

- Data Collection.
    - Create Train and Test Folders.
- Data Preprocessing.
    - Import the ImageDataGenerator library
    - Configure ImageDataGenerator class
    - ApplyImageDataGenerator functionality to Trainset and Testset
- Model Building
    - Import the model building Libraries
    - Initializing the model
    - Adding Input Layer
    - Adding Hidden Layer
    - Adding Output Layer
    - Configure the Learning Process
    - Training and testing the model
    - Save the Model
- Application Building
    - Create an HTML file
    - Build Python Code

Data Collection

Collect images of Garbage then organized into subdirectories based on their respective names as shown in the project structure.Create folders of types of Garbage that need to be recognized.
In this project, we have collected images of 6 types of Garbage like CardBoard ,Paper,Plastic, Glass, Trash & Metal and they are saved in the respective sub directories with their respective names.

Import The ImageDataGenerator Library

Image data augmentation is a technique that can be used to artificially expand the size of a training dataset by creating modified versions of images in the dataset.

The Keras deep learning neural network library provides the capability to fit models using image data augmentation via the ImageDataGenerator class.
Let us import the ImageDataGenerator class from tensorflow Keras

Configure ImageDataGenerator Class

ImageDataGenerator class is instantiated and the configuration for the types of data augmentation

There are five main types of data augmentation techniques for image data; specifically:

- Image shifts via the width_shift_range and height_shift_range arguments.
- The image flips via the horizontal_flip and vertical_flip arguments.
- Image rotations via the rotation_range argument
- Image brightness via the brightness_range argument.
- Image zoom via the zoom_range argument.

Apply ImageDataGenerator Functionality To Trainset And Testset

Let us apply ImageDataGenerator functionality to Trainset and Testset by using the following code.For Training set using flow_from_directory function.

This function will return batches of images from the subdirectories Cardboard,Glass, Metal, Paper, Plastic & Trash 'together with labels 0 to 5{Cardboard: 0, Glass: 1, Metal: 2, Paper: 3,Plastic:4,Trash:5 }

Importing The Model Building Libraries

- ImageDataGenerator: The ImageDataGenerator is a class in the tensorflow.keras.preprocessing.image module that generates batches of augmented image data in real-time during model training.
    - Keras: Keras is a high-level neural network API written in Python that allows for fast experimentation and prototyping of deep learning models.
    - image: from tensorflow.keras.preprocessing import image imports the image module from Keras' tensorflow.keras.preprocessing package. This module provides a number of image preprocessing utilities, such as loading images, converting images to arrays, and applying various image transformations.

Initializing The Model

Keras has 2 ways to define a neural network:

- Sequential
- Function API

Adding CNN Layers

- As the input image contains three channels, we are specifying the input shape as (128,128,3).

- We are adding a convolution layer with activation function as “relu” and
  with a small filter size (3,3) and the number of filters (32) followed by a max-pooling layer.
- Max pool layer is used to downsample the input.(Max pooling is a pooling operation that selects the maximum element from the region of the feature map covered by the filter)
- Flatten layer flattens the input. Does not affect the batch size.

Adding Dense Layers

A dense layer is a deeply connected neural network layer. It is the most common and frequently used layer.

Adding Fully Connected Layers

The number of neurons in the Dense layer is the same as the number of classes in the training set. The neurons in the last Dense layer, use softmax activation to convert their outputs into respective probabilities.

Understanding the model is a very important phase to properly use it for training and prediction purposes. Keras provides a simple method, summary to get the full information about the model and its layers.

Configure The Learning Process

- The compilation is the final step in creating a model. Once the compilation is done, we can move on to the training phase. The loss function is used to find errors or deviations in the learning process. Keras requires a loss function during the model compilation process.
- Optimization is an important process that optimizes the input weights by comparing the prediction and the loss function. Here we are using adam optimizer
- Metrics are used to evaluate the performance of your model. It is similar to the loss function, but not used in the training process

Train The Model

Now, let us train our model with our image dataset. The model is trained for 30 epochs and after every epoch, the current model state is saved if the model has the least loss encountered till that time. We can see that the training loss decreases in almost every epoch till 30 epochs and probably there is further scope to improve the model.

fit_generator functions used to train a deep learning neural network
Arguments:

- steps_per_epoch: it specifies the total number of steps taken from the generator as soon as one epoch is finished and the next epoch has started. We can calculate the value of steps_per_epoch as the total number of samples in your dataset divided by the batch size.

- Epochs: an integer and number of epochs we want to train our model for.

- validation_data can be either:
  - an inputs and targets list
  - a generator
  - an inputs, targets, and sample_weights list which can be used to evaluate the loss and metrics for any model after any epoch has ended.
- validation_steps: only if the validation_data is a generator then only this argument can be used. It specifies the total number of steps taken from the generator before it is stopped at every epoch and its value is calculated as the total number of validation data points in your dataset divided by the validation batch size.

Save The Model

The model is saved with .h5 extension as follows
An H5 file is a data file saved in the Hierarchical Data Format (HDF). It contains multidimensional arrays of scientific data

Test The Model

Evaluation is a process during the development of the model to check whether the model is the best fit for the given problem and corresponding data.
Load the saved model using load_model

Application Building

Now that we have trained our model, let us build our flask application which will be running in our local browser with a user interface.

In the flask application, the input parameters are taken from the HTML page These factors are then given to the model to know to predict the type of Garbage and showcased on the HTML page to notify the user. Whenever the user interacts with the UI and selects the “Image” button, the next page is opened where the user chooses the image and predicts the output.

Create HTML Pages

- We use HTML to create the front end part of the web page.
- Here, we have created 3 HTML pages- index.html, prediction.html, and about.html,team.html
- index.html displays the home page.
- about.html displays an introduction about the project
- prediction.html gives the emergency alert

- We also use JavaScript-main.js and CSS-main.css to enhance our functionality and view of HTML pages.

Build Python Code

Task 1: Importing Libraries

The first step is usually importing the libraries that will be needed in the program.

Importing the flask module in the project is mandatory. An object of the Flask class is our WSGI application. Flask constructor takes the name of the current module (__name__) as argument Pickle library to load the model file.

Task 2: Creating our flask application and loading our model by using load_model method
Task 3: Routing to the html Page

Here, the declared constructor is used to route to the HTML page created earlier.

In the above example, '/' URL is bound with index.html function. Hence, when the home page of a web server is opened in the browser, the html page will be rendered. Whenever you browse an image from the html page this photo can be accessed through POST or GET Method.

Here we are defining a function which requests the browsed file from the html page using the post method. The requested picture file is then saved to the uploads folder in this same directory using OS library. Using the load image class from Keras library we are retrieving the saved picture from the path declared. We are applying some image processing techniques and then sending that preprocessed image to the model for predicting the class. This returns the numerical value of a class (like 0,1 ,2 etc.) which lies in the 0th index of the variable preds. This numerical value is passed to the index variable declared. This returns the name of the class. This name is rendered to the predict variable used in the html page.

Predicting the results
We then proceed to detect all type of Garbage in the input image using model.predict function and the result is stored in the result variable.

Run The Application

- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type "python app.py" command.
- It will show the local host where your app is running on http://127.0.0.1.5000/
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page.