



e-ISSN: 2319-8753 | p-ISSN: 2347-6710

IJRSET

International Journal of Innovative Research in
SCIENCE | ENGINEERING | TECHNOLOGY



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

Volume 12, Issue 13, April 2023

ISSN INTERNATIONAL
STANDARD
SERIAL
NUMBER
INDIA

9940 572 462

6381 907 438

ijrset@gmail.com

www.ijrset.com



AI Based Garbage Segregation Dustbin to Reduce Environmental Pollution

S.D.Vijayakumar¹, J.Pragadeeswaran², J.Priyadharshini³

Associate Professor Department of ECE, Builders Engineering College, Nathakadaiyur, India¹

U G Student Department of ECE, Builders Engineering College, Nathakadaiyur, India.^{2,3}

ABSTRACT: In India, We know that we have a serious garbage problem. The problem is how to integrate the technology with a system of household-level segregation so that waste does not end up in landfills, but is processed and reused. We know that there will be no value from waste, as energy or material, if it is not segregated. So we are planning to segregate the waste by using Machine Learning algorithms to analyze the waste in a picture and it will categorize the waste into Biodegradable waste, Non-biodegradable waste, Hazardous waste. Then by integrating the microcontroller with the stepper motor, which is placed under the dustbin, it rotates the dustbin in certain degrees and stores the waste in the dustbin based on their category and the servo motor is placed above the dustbin which is used to push the wastes into the dustbin. The benefits of waste segregation are lower waste costs, increased recycling rate, potential revenue streams and reduced landfill impact.

KEYWORDS: Garbage segregation, Artificial Intelligence, Waste management, Environment, pollution control, Arduino, Machine learning, Convolutional Neural Network, Dustbin, Servo motor.

I. INTRODUCTION

The collection, material handling, treatment, and disposal of garbage have all been closely interdependent under the heading of waste management. The monitoring and regulation of the waste management procedure is critical. We should always focus on ensuring that the garbage disposal and treatment techniques are just about as advantageous as conceivable considering the growing amount of waste generation. When residential garbage is taken into consideration, the method of disposal encompasses garbage collection by the government at the doorstep of the inhabitants. Even though the overwhelming majority of the time, improper waste segregation has a detrimental impact on the subsequent segregation stages. We want to identify the most effective way to segregate waste that is also efficient and affordable. The Solid Waste Management Rules 2016, the generators of waste are supposed to segregate the waste into three parts which are biodegradable, non-biodegradable and domestic hazardous wastes and hand them over to the local authorities on time to time basis. Untreated waste become a breeding point for many harmful microorganisms. The gases released at the site of untreated waste is harmful and causes diseases like nausea and many other airborne diseases. Untreated waste mixes with water and causes many waterborne diseases like cholera and diarrhea. We propose to solve the waste segregation problem using machine learning. This model consists of training a neural network based to classify the images into biodegradable and non-biodegradable wastes. Logistic regression and basic binary classification is applied.

II. METHODOLOGY

SOFTWARE LAYER

For image recognition, classification, and detection, deep learning neural networks are crucial. The neural network's several layers cooperate to produce the desired result. Each layer has the option of working on image sampling or shrinking the dimensions. A multilayer perceptron is utilized in convolution neural networks (CNNs), a form of neural network. Finding a generic 2D image and gathering data from it is beneficial. On the basis of the functionality of the system, the architecture of the project consists following layers. They are briefly described below. The network is made up of:

- Input layer

- Convolutional layer
- Sample layer
- Output layer

The sample layer and convolution layer can both have multiple layers to increase accuracy and structure. Instead of looking at the entire image, CNN looks at the small area that has useful information about the image. This area is frequently referred to as Region CNN or RCNN. Think of creating a tiny neural network with k outputs on a portion of the input image, then representing the results vertically. The image's dimensions can be lowered to a point where they are suitable for pooling with the use of a suitable kernel. In order to create an image with varying width, height, and depth, the network is moved throughout the entire image. In the latter phases, we apply an activation function (RELU or SoftMax function) to the convolved image to obtain the output Convolution is accomplished via matrix multiplication.

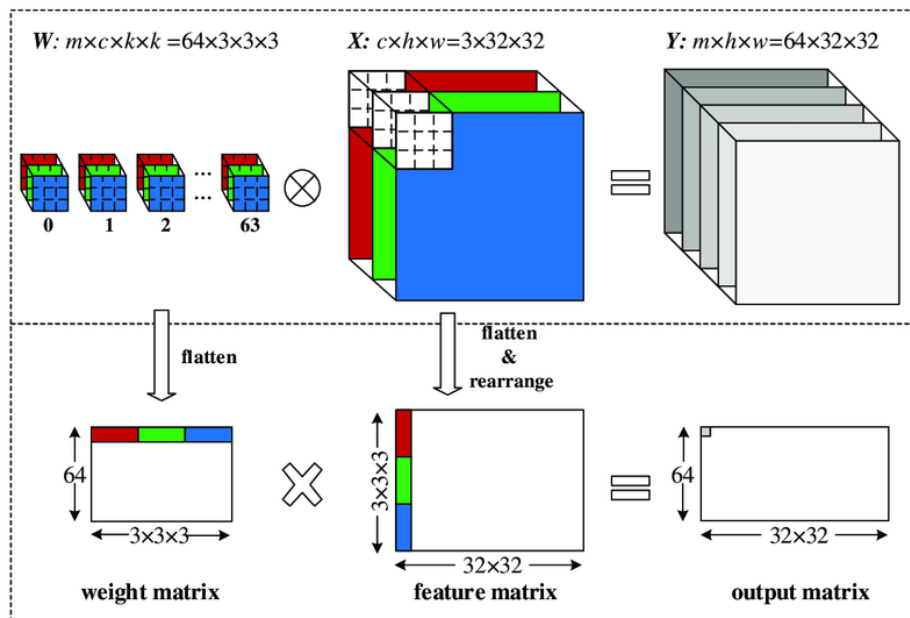


Fig.1 (CNN – Convolutional Neural Network)

III. IMPLEMENTATION & SPECIFICATION

The supervised learning strategy is used to resolve the segregation problem. First, photographs of cardboard, glass, paper, plastic, metal and other materials were gathered, including soda cans. These categories each contain about 500 pictures. Each image was resized to 64x64 and changed from color to grayscale. This is crucial for altering the neural network to reduce processing time and computational complexity. Grayscale conversion enables the RGB and black and white images support are removed from the network. This is done in light of the fact that trash color is not a crucial consideration. When the data is prepared, we produce output labels for every dataset class. To forecast the output category, this is done.



```
# Loading the model and start predicting classes based on image date captured by
import cv2
from tensorflow.keras.models import load_model
import numpy as np

# Load the model
model = load_model("../desktop/navi_model1.h5")

# Set the expected input size for the model
input_size = (224, 224)

# Start capturing images from the default camera
cap = cv2.VideoCapture(0)

while True:
    # Read a frame from the camera
    ret, frame = cap.read()

    if not ret:
        # Display the frame
        cv2.imshow("frame", frame)

    # Wait for the 'c' key to be pressed to capture the image
    if cv2.waitKey(1) & amp; KPF == ord('c'):
        # Resize the captured image to the expected input size for the model
        img = cv2.resize(frame, input_size)
        img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

Fig.2 (Capturing image through camera and predicting the class)

```
main_folder_path = "/content/drive/MyDrive/Garbage-rec1and4"

def get_num_files(directory):
    count = 0
    for filename in os.listdir(directory):
        if os.path.isfile(os.path.join(directory, filename)):
            count += 1
    return count

for class_name in os.listdir(main_folder_path):
    class_path = os.path.join(main_folder_path, class_name)
    num_files = get_num_files(class_path)
    print(class_name)
    print("Number of files:", num_files)
    img_path = os.path.join(class_path, os.listdir(class_path)[0])
    img = image.open(img_path)
    print("Shape of the image:", img.size)

# Non-degradable
Number of files: 3971
Shape of the image: (224, 224)
# Recyclable
Number of files: 3971
Shape of the image: (224, 224)
# Non-degradable
Number of files: 3971
Shape of the image: (224, 224)
```

Fig.3 (Classifying three classes and their respective image files)



```

from keras.callbacks import ModelCheckpoint, EarlyStopping, TensorBoard
history = model.fit(X_train, y_train, batch_size=32, epochs=10, validation_data=(X_val, y_val))

test_loss, test_acc = model.evaluate(X_test, y_test)

print('test accuracy:', test_acc)

Epoch 1/10 [-----] 53s 169ms/step - loss: 1.5166 - accuracy: 0.7954 - val_loss: 0.4909 - val_accuracy: 0.8625
Epoch 2/10 [-----] 39s 183ms/step - loss: 0.1260 - accuracy: 0.9394 - val_loss: 0.4475 - val_accuracy: 0.8757
Epoch 3/10 [-----] 39s 165ms/step - loss: 0.1001 - accuracy: 0.9663 - val_loss: 0.5117 - val_accuracy: 0.8741
Epoch 4/10 [-----] 48s 169ms/step - loss: 0.0568 - accuracy: 0.9827 - val_loss: 0.5109 - val_accuracy: 0.8999
Epoch 5/10 [-----] 48s 159ms/step - loss: 0.0073 - accuracy: 0.9948 - val_loss: 0.6246 - val_accuracy: 0.8799
Epoch 6/10 [-----] 41s 170ms/step - loss: 0.0060 - accuracy: 0.9955 - val_loss: 0.5866 - val_accuracy: 0.8875
Epoch 7/10 [-----] 41s 179ms/step - loss: 0.0023 - accuracy: 0.9991 - val_loss: 0.6240 - val_accuracy: 0.8951
Epoch 8/10 [-----] 41s 179ms/step - loss: 0.0029 - accuracy: 0.9921 - val_loss: 0.8086 - val_accuracy: 0.8994
Epoch 9/10 [-----] 38s 153ms/step - loss: 0.0013 - accuracy: 0.9998 - val_loss: 0.7681 - val_accuracy: 0.8936
Epoch 10/10 [-----] 37s 150ms/step - loss: 0.0075 - accuracy: 0.9970 - val_loss: 0.8090 - val_accuracy: 0.8946
75/75 [-----] - 12s 150ms/step - loss: 0.0370 - accuracy: 0.8850
test accuracy: 0.8850188851356506
    
```

Fig.4 (Model fitting)

```

# Importing the pretrained model VGG16 adding custom layers to model.

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten
from tensorflow.keras.applications.vgg16 import VGG16
from sklearn.model_selection import train_test_split
from keras.applications.vgg16 import VGG16

vgg_model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

for layer in vgg_model.layers:
    layer.trainable = False

model = Sequential()
model.add(vgg_model)
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(3, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

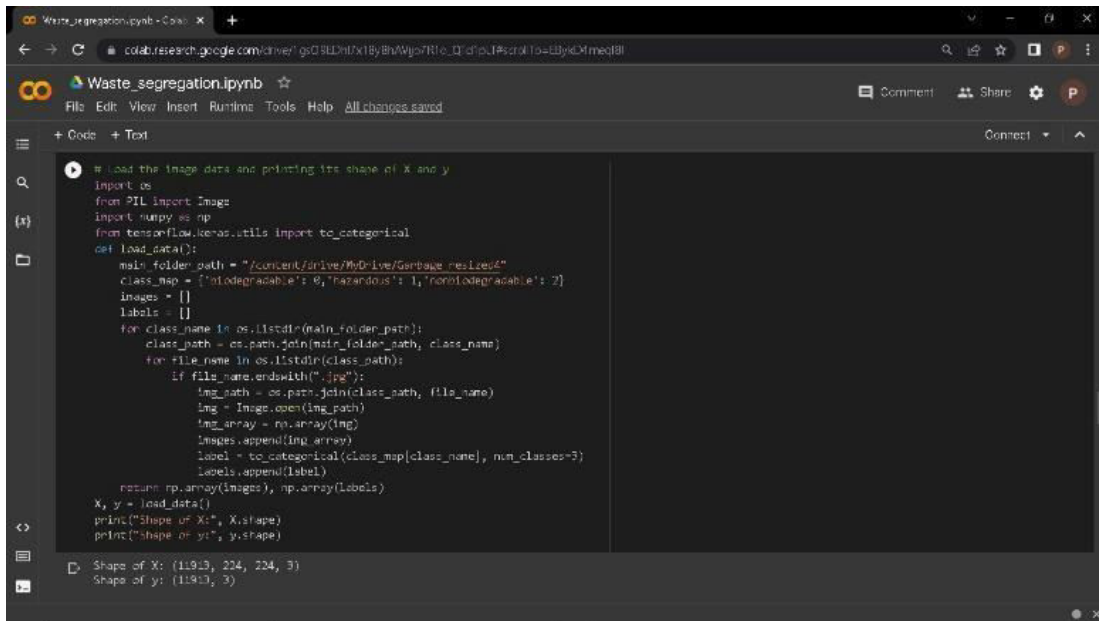
[ ] from sklearn.model_selection import train_test_split

# Split the data into training, validation, and test sets
X_train_val, X_test, y_train_val, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    
```

Fig.5 (Creating the Model)

TRAINING

To prepare the data for training, each category's photos are first turned into an array. Hence, a Numpy array will be used to hold cardboard images. Here, we use the Numpy library since it works well with matrices and arrays. All of the categories will be combined into a single array of arrays together with their respective labels, creating an array of arrays containing all photographs. The neural network is fed with this array of arrays, and training is completed.



```

# Load the image data and printing its shape of X and y
import os
from PIL import Image
import numpy as np
from tensorflow.keras.utils import to_categorical

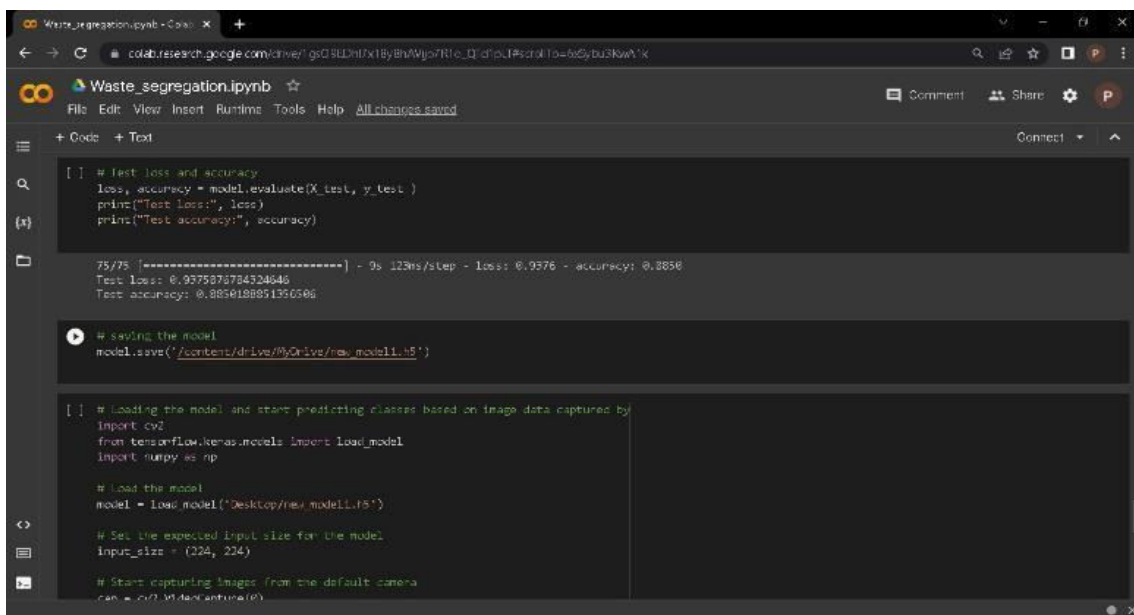
def load_data():
    main_folder_path = "/content/drive/MyDrive/Garbage resized"
    class_map = {'biodegradable': 0, 'hazardous': 1, 'nonbiodegradable': 2}
    images = []
    labels = []
    for class_name in os.listdir(main_folder_path):
        class_path = os.path.join(main_folder_path, class_name)
        for file_name in os.listdir(class_path):
            if file_name.endswith(".jpg"):
                img_path = os.path.join(class_path, file_name)
                img = Image.open(img_path)
                img_array = np.array(img)
                images.append(img_array)
                label = to_categorical(class_map[class_name], num_classes=3)
                labels.append(label)
    return np.array(images), np.array(labels)
X, y = load_data()
print("Shape of X:", X.shape)
print("Shape of y:", y.shape)
    
```

Shape of X: (119, 3)
Shape of y: (119, 3)

Fig.6 (Image data into array)

VALIDATION

The model's validation is crucial because it lets us know how well our model works with the test data set's random data. Our goal is to increase validation accuracy. Snippet 3 employs a sequential model with a single input convolution layer, 64x64 input pixels, and a 3x3 kernel size. Kernel size essentially specifies the size of the matrix that traverses the picture matrix, compares every pixel in the 3x3 pixel grid, and shrinks it to a single pixel. In order to pad each image with zeroes and prevent it from shrinking below 64x64 due to convolution, a max pooling layer is introduced.



```

[] # Test loss and accuracy
loss, accuracy = model.evaluate(X_test, y_test)
print("Test Loss:", loss)
print("Test accuracy:", accuracy)

75/75 [-----] - 9s 123ms/step - loss: 0.9976 - accuracy: 0.8856
Test loss: 0.9975878784324646
Test accuracy: 0.8856188891356566

[] # saving the model
model.save('/content/drive/MyDrive/new_model1.15')

[] # Loading the model and start predicting classes based on image data captured by
import cv2
from tensorflow.keras.models import load_model
import numpy as np

# Load the model
model = load_model('/Desktop/new_model1.15')

# Set the expected input size for the model
input_size = (224, 224)

# Start capturing images from the default camera
cap = cv2.VideoCapture(0)
    
```

Fig.7 (Test loss and accuracy)



In this case, three convolution layers are added, each with a convolution size of 64, 64, and 128. Moreover, a dropout layer with a 0.5 dropout is added. By excluding some neurons from the training process, this layer helps to increase computation even further. The model now includes a hidden layer with 128 neurons and a rectified linear activation function (relu). The output layer uses the softmax function and has 6 neurons because there are 6 classes. Multiclass categorization uses the Softmax function. more computation by sparingly using particular neurons during training. The model now includes a hidden layer with 128 neurons and a rectified linear activation function (relu). The output layer uses the softmax function and has 6 neurons because there are 6 classes. Multiclass categorization uses the Softmax function. We obtained an accuracy of approximately 75% on the test case and 65% on the validation case, for 25 epochs, using the provided tiny dataset and the fundamental convolution model. The accuracy would rise even further if run over a larger number of epochs. Using pre-established models like the VGG16 or the ImageNet is another option to improve accuracy.

Sample output

Be aware that the model gets it right eight out of the ten times, but gets it wrong in just two of the ten samples. The Solid Waste Management Rules of 2000 only required waste generators to periodically turn over waste to local authorities. Waste management is essential for maintaining public health, the environment, and human hygiene. The suggested system offers a quick and efficient way to separate waste using machine learning, eliminating all need for human involvement in the segregation stage. An efficiency of 80% has been achieved in the testing process.

IV.HARDWARE LAYER

The hardware layer is the most important part of the project. It has been placed in the dustbin itself .The hardware layer consists of the following components:

ARDUINO UNO

The microcontroller is the main part of the garbage segregation dustbin. Arduino UNO is an open-access microcontroller. The Arduino controller chip has 16 input and output pins for interfacing with embedded devices. It operates on a 5v power supply. It has 32KB of flash memory and 2KB of static RAM. The Arduino IDE allows you to configure the Arduino controller in embedded C.



Fig.8 (Arduino UNO)

SERVO MOTOR

There was two servo motors placed in dustbin as a hardware component which were interconnected with Arduino. Based on the command of Arduino the servo motor works. There was two types of servo motor used. They were positional control servo motor and continuous servo motor.

Continuous servo motor:

This motor have the capability of controlling the direction of motor spin. This motor rotates the separation of the three section which are Biodegradable, Non-biodegradable and Hazardous. Based on the type classified by predicted by the computer vision and image classification process. The motor rotates the separator to the opening where the waste was pushed into it. The motor rotates based on the command instructed by the Arduino under the classification prediction by the image processing method.

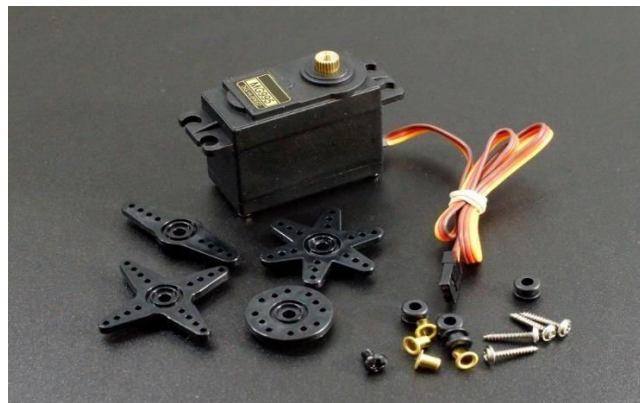


Fig.9 (Continuous servo motor)

Positional control servo motor:

The positional control servo motor can be moved up to 180° where 90° each direction. This also works similar to continuous servo motor based on the command but instead of the direction we can able to control the position of the motor. This works as the hand to push the garbage to the dustbin.



Fig.10 (Positional servo motor)

ULTRASONIC SENSOR

Ultrasonic sensors work by emitting sound waves at a frequency too high for humans to hear. They then wait for the sound to be reflected back, calculating distance based on the time required. This is similar to how radar measures the time it takes a radio wave to return after hitting an object. We use this to monitor the level of the dustbin whether it is full or not. If the dustbin is full the indication message will be send to the user. This sensor has been interconnected with Arduino and GSM module.

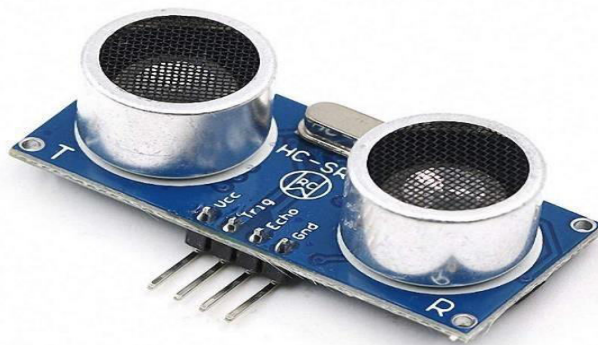


Fig.11 (Ultrasonic sensor)

GSM MODULE

GSM (Global System for Mobile communication) is a digital mobile network that is widely used by mobile phone users in various parts of the world. GSM uses a variation of time division multiple access (TDMA) and is the most widely used of the three digital wireless telephony technologies: TDMA, GSM and code-division multiple access (CDMA). GSM digitizes and compresses data, then sends it down a channel with two other streams of user data, each in its owntime slot. It operates at either the 900 megahertz (MHz) or 1,800 MHz frequency band. We use this to send the indication of the level of the dustbin. It sends the indication to the user when the dustbin is full.



Fig.12 (GSM module)



REFERENCES

- [1] B.Dev, "Automatic Waste Segregation using Image Processing and Machine Learning", May 2018.
- [2] Adhithya .P.M, S.V Kaushal, P. Mahalakshmi, "Survey on Identification and Classification of Waste for efficient disposal and recycling", International Journal of Engineering and Technology, 2018.
- [3] Y. LeCun, K. Kavukcuoglu, and C. Farabet. Convolutional networks and applications in vision. In Circuits and Systems (ISCAS), Proceedings of 2010 IEEE International Symposium on, pages 253–256. IEEE, 2010.
- [4] Content-based Image Retrieval using Deep Convolution Neural Network | D.D.Pukale S.G.Bhirud V.D. Katkar 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA) 2017.
- [5] An efficient license plate recognition system using convolution neural networks | Cheng-Hung Lin ; Yong-Sin Lin ; Wei-Chen Liu, 2018 IEEE International Conference on Applied System Invention (ICASI), 2018.
- [6] Image convolution optimization using sparse matrix vector multiplication technique | B Bipin Jyothisha J Nair, International Conference on Advances in Computing, Communications and Informatics (ICACCI) 2016.
- [7] Fast algorithm using summed area tables with unified layer performing convolution and average pooling | Akihiko Kasagi Tsuguchika Tabaru Hirotaka Tamura, IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP) 2017.
- [8] Deep Learning using Linear Support Vector Machines |, Yichuan Tang,
- [9] Department of Computer Science, University of Toronto. Toronto, Ontario, Canada.



INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

 9940 572 462  6381 907 438  ijirset@gmail.com



www.ijirset.com

Scan to save the contact details