Computer Vision Project Report

Kitchen Item Detection

Group Number : 13

Artificial Intelligence And Data Science

Jio Institute

Contents

01

Introduction

Kitchen Item Detection is a computer vision application that detects kitchen items in an image frame. It can be useful in scenarios such as tracking inventory in a restaurant and all types of kitchens. The technology involves using object detection and recognition techniques, with machine learning algorithms trained on images. The algorithms can accurately detect specific items, providing insights into usage patterns, predicting future demand, and optimizing inventory management. Kitchen Item Detection has the potential to revolutionize inventory management.

Data Overview

We have generated a dataset of 171 images, consisting of three distinct types of food ingredients: chickpea, rice, and Toor dal. The dataset contains 77 images of chickpea, 49 images of rice, and 45 images of Toor dal.

Data Conversion Methodology:

we converted Images to grayscale using the cv2.cvtColor() method. It then flattens the grayscale image into a one-dimensional array and appends the corresponding label (i.e., the name of the folder containing the image) to create a row of data. These rows of data are then added to a list called data. This process of converting images to numerical format is an essential preprocessing step in many computer vision applications, including image classification, object detection, and face recognition.

Modules/Libraries Used:

- ·CV2: A computer vision library used for image processing tasks such as reading, manipulating, and saving images.
- ·NumPy: A library for numerical computing in Python that is used to perform numerical operations on the images.
- OS: A module used for accessing files and directories in the file system.

Data Preprocessing

We performed a pre-processing step as a part of our computer vision application. Firstly, we converted the images to grayscale using the cv2.cvtColor() method. After that, we flattened the grayscale images into one-dimensional arrays and added the corresponding label, which was the name of the folder containing the image, to create a data row. These rows of data were then collected and added to a list called data. This pre-processing step was necessary to convert the images into numerical format, which is vital for many computer vision applications, such as image classification, object detection, and face recognition. By performing this step, we standardized the image format and ensured that our models could accurately interpret the data.

Data Preparation , Converting Color, Resizing, Flattening appending and Converting to Array

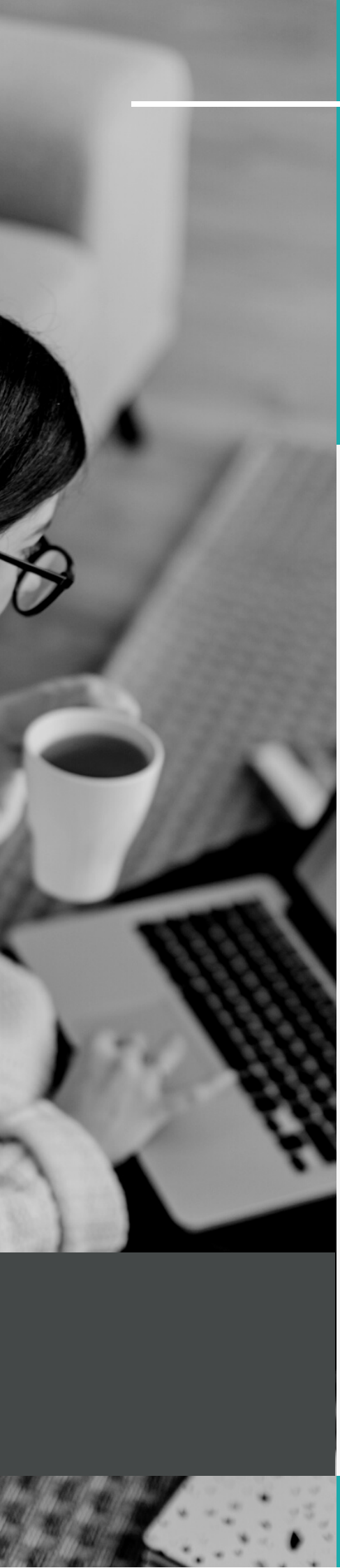
```
data = []
labels = []

for folder_name in os.listdir('C:/Users/SagarSRK/Desktop/DS/'):
    for image_name in os.listdir('C:/Users/SagarSRK/Desktop/DS/' + folder_name):
        image_path = 'C:/Users/SagarSRK/Desktop/DS/' + folder_name + '/' + image_name
        image = cv2.imread(image_path)
        image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
        image = cv2.resize(image, (128, 128))
        data.append(image.flatten())
        labels.append(folder_name)


data = np.array(data, dtype='float') / 255.0

print(data)
```

```
[?]
...
[[0.4627451  0.41176471 0.44705882 ... 0.14509804 0.14901961 0.15686275]
 [0.13333333 0.1372549  0.16470588 ... 0.36078431 0.35294118 0.36470588]
 [0.02352941 0.02352941 0.02352941 ... 0.07843137 0.07058824 0.08235294]
 ...
 [0.36862745 0.34901961 0.30588235 ... 0.08627451 0.08235294 0.08235294]
 [0.23921569 0.23529412 0.23137255 ... 0.12156863 0.12941176 0.12941176]
 [0.03529412 0.03137255 0.03137255 ... 0.03529412 0.03921569 0.04705882]]
```



ML Models

- **Logistic Regression**
- **KNN**
- **Naïve Bayes Classifier**
- **Ensemble Techniques : Random Forest**

- **Logistic Regression**

After performing the pre-processing steps on our image dataset, we applied the logistic regression algorithm for our analysis. The logistic regression model provided us with a good accuracy of around 40%, which was a promising result.

- **Random Forest**

However, upon further experimentation, we found that the Random Forest algorithm was better suited to our dataset and produced the best accuracy of around 55%. The Random Forest algorithm is a popular ensemble learning method that combines multiple decision trees to improve the model's performance. We found that this algorithm performed exceptionally well on our image dataset, and it outperformed the logistic regression model. we found that the Random Forest algorithm was the best fit for our analysis.

- We have also used KNN, Naïve Bayes Classifier Models.

Evaluation

Model selection and training with Multiple Models

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(max_iter=1000)
model.fit(trainX,trainY)
y_pred = model.predict(testX)
print(classification_report(testY, y_pred))
```

[127]

	precision	recall	f1-score	support
Chickpea	0.50	0.49	0.49	41
Rice	0.21	0.27	0.24	22
Toordal	0.47	0.35	0.40	23
accuracy			0.40	86
macro avg	0.39	0.37	0.38	86
weighted avg	0.42	0.40	0.40	86

Random forest

```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(trainX, trainY)
y_pred = model.predict(testX)
y_pred = model.predict(testX)
print(classification_report(testY, y_pred))
```

[129]

	precision	recall	f1-score	support
Chickpea	0.58	0.78	0.67	41
Rice	0.44	0.32	0.37	22
Toordal	0.53	0.35	0.42	23
accuracy			0.55	86
macro avg	0.52	0.48	0.49	86
weighted avg	0.53	0.55	0.52	86

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
model = GaussianNB()
model.fit(trainX, trainY)
y_pred = model.predict(testX)
y_pred = model.predict(testX)
print(classification_report(testY, y_pred))
```

[131]

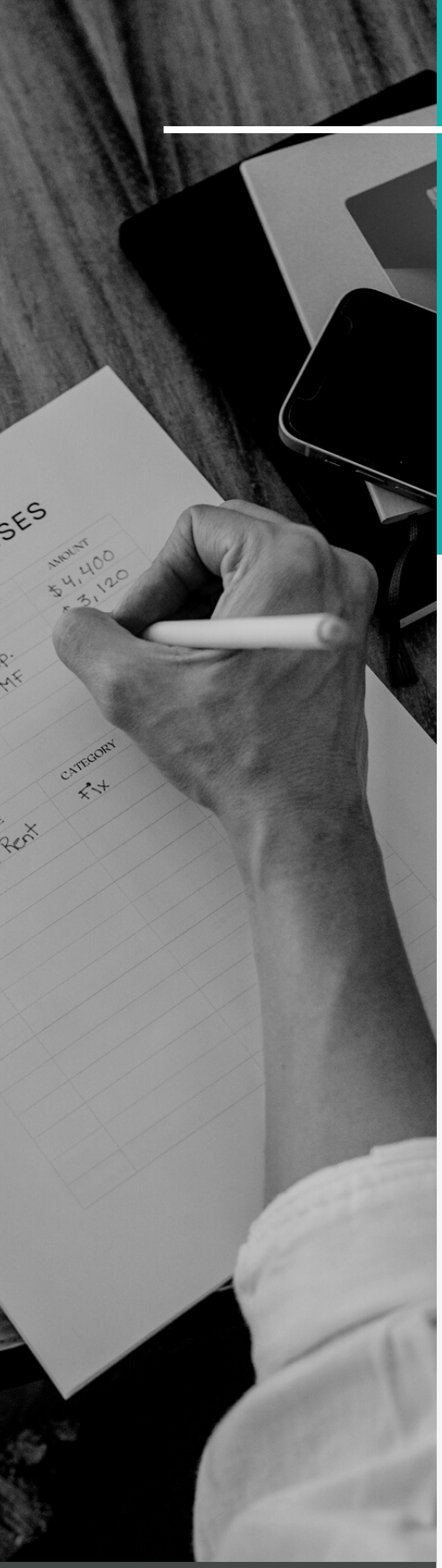
	precision	recall	f1-score	support
Chickpea	0.64	0.61	0.62	41
Rice	0.12	0.09	0.11	22
Toordal	0.45	0.61	0.52	23
accuracy			0.48	86
macro avg	0.41	0.44	0.42	86
weighted avg	0.46	0.48	0.46	86

KNN

```
from sklearn.neighbors import KNeighborsClassifier
model = KNeighborsClassifier(n_neighbors=3)
model.fit(trainX, trainY)
y_pred = model.predict(testX)
y_pred = model.predict(testX)
print(classification_report(testY, y_pred))
```

[133]

	precision	recall	f1-score	support
Chickpea	0.55	0.59	0.56	41
Rice	0.17	0.18	0.18	22
Toordal	0.53	0.43	0.48	23
accuracy			0.44	86
macro avg	0.42	0.40	0.41	86
weighted avg	0.45	0.44	0.44	86



Conclusion

Using deep learning techniques to improve the accuracy of the model. Deep learning techniques such as convolutional neural networks (CNNs) have been shown to be highly effective for image recognition tasks. By training CNN on the food ingredient dataset, we could potentially achieve even better accuracy in food ingredient detection.

References

- Medium.com
- Programming Computer Vision with Python
- The Hundred Page Machine Learning Book

Future Improvements

- ·One potential improvement for this project is the addition of quantity detection in addition to item detection. Currently, the project is focused on identifying the type of ingredient in the image, but it would also be useful to determine the quantity of each ingredient in the image. This could be done using object detection techniques to locate and count the individual pieces of food in the image.
- ·One more potential improvement for this project is to add more data to the dataset. Currently, the dataset is relatively small, and more images could be added to improve the accuracy of the machine learning model. This could be done by collecting additional images of the same food ingredients, or by adding images of other food ingredients to the dataset to make it more diverse.

Team Members

Six Sigma - 6Σ