SHRI VAISHNAV VIDYAPEETH VISHWAVIDYALAYA

SHRI VAISHNAV INSTITUTE OF INFORMATION TECHNOLOGY

DEPARTMENT OF COMPUTER SCIECNCE AND ENGINEERING

The Project Report for the project entitled

"Devanāgarī Numerals Recognition System"

submitted for the course

Programming with Python [BTIT507N]

of Bachelor of Technology in

CSE (AI specialization with IBM) degree program

Year 3 Semester 5 Section U

Enrollment Number ~ 22100BTCSAII11062

Submitted by

UMANG GOSWAMI

Submitted to

Prof. Avadesh Sharma

TABLE OF CONTENTS

Devanāgarī Numerals Recognition System

1. Abstract

This project, **"Devanāgarī Numerals Recognition System",** developed by Umang Goswami, aims to create a robust and reliable system for the recognition of Devanāgarī numerals. The is to develop a highly accurate and efficient Devanāgarī numeral recognition system that can be seamlessly integrated into a wide range of applications, ultimately enhancing the processing and management of Devanāgarī-based data.

The system leverages state-of-the-art machine learning algorithms to accurately identify hand-written Devanāgarī numerals, enabling highly efficient and precise data processing in various applications that necessitate Devanāgarī numeral recognition. These applications span diverse domains, including document processing, financial data analysis, and information extraction from Devanāgarī-based documents.

By harnessing advanced computational techniques, this innovative system promises to significantly enhance the accuracy and productivity of tasks involving Devanāgarī numeral recognition, contributing to improved data management and analysis across various industries and research fields.

2. Technologies Used

The Devanāgarī Numerals Recognition System utilizes a combination of advanced machine learning techniques and image processing algorithms to achieve high-accuracy recognition of Devanāgarī numerals. Specifically, the system employs Convolutional Neural Networks as the primary deep learning architecture for image classification.

The neural networks are trained on a large and diverse dataset of Devanāgarī numeral images, comprising a total of 26,500 samples. This extensive training enables the system to learn the distinctive features and patterns associated with each numeral, allowing for highly accurate recognition of hand-written Devanāgarī numerals. The system's performance has been extensively evaluated, demonstrating its robustness and reliability in a variety of real-world applications.

In brief various technologies used for the development of the aforementioned project are as follows:

- **IDE:** Pycahrm 2024.1 (Professional Edition)
- **Programming Language:** Python v. 3.12
- **Various Libraries used:** scikit-learn, OpenCV, TensorFlow, NumPy, SciPy, Keras, pillow, etc. For the complete list of libraries use see requirements.txt at next page.

Specifications of the computer systems used for development of the project are:

- **Processor:** 12th Gen Intel(R) Core(TM) i7-12650H @ 2.30 GHz
- **Installed RAM:** 16.0 GB (15.7 GB usable)
- **Internal Storage:** 1 GB SSD
- **System type:** 64-bit operating system, x64-based processor
- **OS:** Windows 11 Home Single Language, version 23H2

3. All Libraries used – requirements.txt

```
absl-py==2.1.0
astunparse==1.6.3
blinker==1.9.0
certifi==2024.8.30
charset-normalizer==3.4.0
click==8.1.7
colorama==0.4.6
Flask==3.1.0
Flask-Cors==5.0.0
flatbuffers==24.3.25
gast==0.6.0
google-pasta==0.2.0
grpcio==1.68.1
h5py==3.12.1
idna==3.10
itsdangerous==2.2.0
Jinja2==3.1.4
joblib==1.4.2
keras==3.7.0
libclang==18.1.1
Markdown==3.7
markdown-it-py==3.0.0
MarkupSafe==3.0.2
mdurl==0.1.2
ml-dtypes==0.4.1
namex==0.0.8
numpy==2.0.2
opencv-python==4.10.0.84
opt_einsum==3.4.0
optree==0.13.1
packaging==24.2
protobuf==5.29.1
Pygments==2.18.0
requests==2.32.3
rich==13.9.4
scikit-learn==1.5.2
scipy==1.14.1
setuptools==75.6.0
six==1.17.0
```

```
tensorboard==2.18.0
tensorboard-data-server==0.7.2
tensorflow==2.18.0
tensorflow_intel==2.18.0
termcolor==2.5.0
threadpoolctl==3.5.0
typing_extensions==4.12.2
urllib3==2.2.3
Werkzeug==3.1.3
wheel==0.45.1
wrapt==1.1
```

4. System Requirements

The system requirements for the successful installation and operation of the Devanāgarī Numerals Recognition System are as follows:

4.1. Hardware Requirements:

- Processor: The system requires at least an Intel Core i5 processor or an equivalent, with a clock speed of 2.0 GHz or higher. This level of processing power is necessary to handle the computationally intensive tasks involved in the recognition and classification of the Devanāgarī numerals.
- RAM: A minimum of 8 GB of RAM is recommended for the smooth operation of the system. This amount of memory ensures that the system can efficiently load and process the large datasets required for training and running the machine learning models.
- Storage: The system requires 25 GB of available disk space to accommodate the installation files, the training dataset, and any additional resources needed for the recognition process.
- Display: A display with a resolution of 1080p or higher is recommended for optimal visualization and interaction with the system's interface.

4.2. Software Requirements:

- Operating System: The Devanāgarī Numerals Recognition System is designed to operate on at least Windows 10 or Windows 11 operating systems, or their equivalent. These modern operating systems provide the necessary software environment and libraries to support the system's functionality.
- Python: The system requires Python version 3.12 or higher to be installed, as the codebase and supporting libraries are developed using this programming language. This version of Python ensures compatibility with the latest advancements in machine learning and image processing libraries.

5. Dataset used for training

The Devanāgarī Numerals Recognition System was developed using a comprehensive dataset of 26,500 Devanāgarī numeral images. I had modified/restructured the dataset, its file structures as to better suit my project needs and feasibility. The dataset was obtained from the following source:

Handwritten Devanāgarī characters - vowels and numerals (38,750 isolated images + 9,200 isolated images). (2023). [Dataset]. In Mendeley Data (Version 4). doi:10.17632/pxrnvp4yy8.4 Retrived from https://data.mendeley.com/datasets/pxrnvp4yy8/4

whose description may be found in the research paper –

Prashanth, D. S., Mehta, R. V. K., & Challa, N. P. (2021). A multi-purpose dataset of Devanāgarī script comprising of isolated numerals and vowels. Data in Brief, 40, 107723. https://doi.org/10.1016/j.dib.2021.107723 Retrived from https://pdf.sciencedirectassets.com/311593/1-s2.0-S2352340921X00078/1-s2.0-S2352340921009987/main.pdf

6. Source Code

6.1. main.py

```
import os
from src.data_loader import Load_Data, Preprocess_Data
from src.model import Create_Model
from src.train import Train_Model
from src.evaluate import Evaluate_Model
from src.predict import Predict_Character
from sklearn.model_selection import train_test_split
from tkinter import *
from tkinter.filedialog import askopenfilename


root = Tk()
data_dir = os.path.abspath(r".\dataset")  # Path to the
dataset folder


def prepare_DNRS_model():
    # Load and preprocess the data
    X, y = Load_Data(data_dir)
    X, y = Preprocess_Data(X, y)
    # Split the dataset into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

    # Create the model
    model = Create_Model(y.shape[1])
    # Train the model
    Train_Model(model, X_train, y_train, X_test, y_test)
    # Evaluate the model
    Evaluate_Model(model, X_test, y_test)

    return model


def make_label_predictions(model):
    Tk.lift(root)
    img_path = str(askopenfilename(defaultextension=".img",
initialdir=".",
          title="Select an image to make label-predictions"))

    return Predict_Character(model=model, image_path=img_path)


def main():
    print("Welcome to Devanagari Numerals Recognition System")
    print("Developed by Umang Goswami")
    print("-" * 21)
    print("This is the presentation of how the system shall
work")
    print("-" * 21)
    print("\nFirst the DNRS_model shall be developed and
saved")
    DNRS_model = prepare_DNRS_model()
    print("\nCongratulations the DNRS_model is successfully
developed and saved !!")
    print("\nNow we shall make label-predictions")
    predicted_label = make_label_predictions(DNRS_model)
    print(f'\nPredicted label: {predicted_label}')


if __name__ == '__main__':
    main()
```

6.2. src/data_loader.py

```
import os
import cv2
import numpy as np


def Load_Data(data_dir):
    """Load images and labels from the dataset directory."""
    images = []
    labels = []

    # Iterate through each subfolder in the dataset directory
    for label in os.listdir(data_dir):
       label_dir = os.path.join(data_dir, label)
       if os.path.isdir(label_dir):
          for image_file in os.listdir(label_dir):
             image_path = os.path.join(label_dir, image_file)
             image = cv2.imread(image_path,
cv2.IMREAD_GRAYSCALE)

             image = cv2.resize(image, (28, 28))  # Resize to
28x28
             if image is not None:
                images.append(image)
                labels.append(int(label))  # Convert label to
integer
    return np.array(images), np.array(labels)


def Preprocess_Data(X, y):
    """Preprocess the images and labels."""
    # Resize images to 28x28 and normalize
    X_resized = np.array([cv2.resize(img, (28, 28)) for img in
x])
    X_normalized = X_resized.astype('float32') / 255.0
    X_normalized = X_normalized.reshape(-1, 28, 28, 1)  # 
Reshape for CNN

    # One-hot encode the labels
    from tensorflow.keras.utils import to_categorical
    y_encoded = to_categorical(y)

    return X_normalized, y_encoded
```

6.3. src/ evaluate.py

```
from data_loader import Load_Data, Preprocess_Data
import os


data_dir = os.path.abspath(r".\dataset")  # Update this path
X, y = Load_Data(data_dir)
X, y = Preprocess_Data(X, y)


def Evaluate_Model(model, X_test, y_test):
    """Evaluate the model on the test data."""
    loss, accuracy = model.evaluate(X_test, y_test)
    print(f'\nTest Loss: {loss:.4f}, Test Accuracy: 
{accuracy:.4f}')
```

6.4. src/model.py

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, 
Flatten, Dense


def Create_Model(num_classes):
    model = Sequential()
    model.add(Conv2D(32, (3, 3), activation='relu', 
input_shape=(28, 28, 1)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    model.add(Dense(num_classes, activation='softmax'))

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

    return model
```

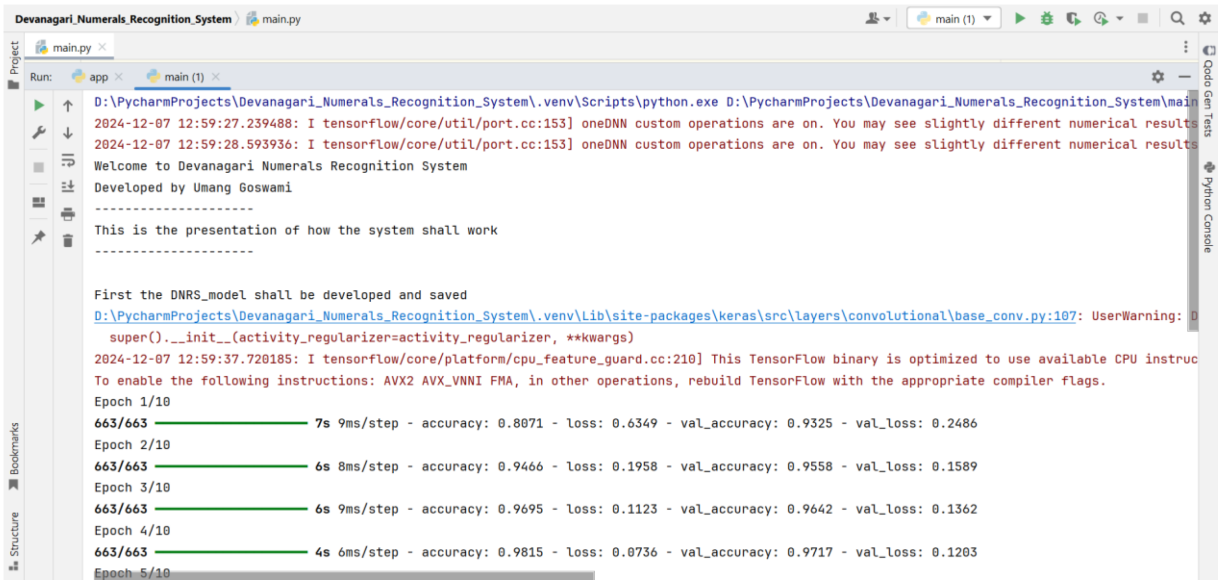
6.5. src/predict.py

```
import cv2
import numpy as np


def Load_and_Preprocess_Image(image_path):
    """Load and preprocess the image for prediction."""
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    image = cv2.resize(image, (28, 28))
    image = image.astype('float32') / 255.0
    image = image.reshape(-1, 28, 28, 1)  # Reshape for the
model

    return image


def Predict_Character(model, image_path):
    """Predict the character in the given image."""
    image = Load_and_Preprocess_Image(image_path)

    prediction = model.predict(image)
    predicted_label = np.argmax(prediction)

    return predicted_label
```

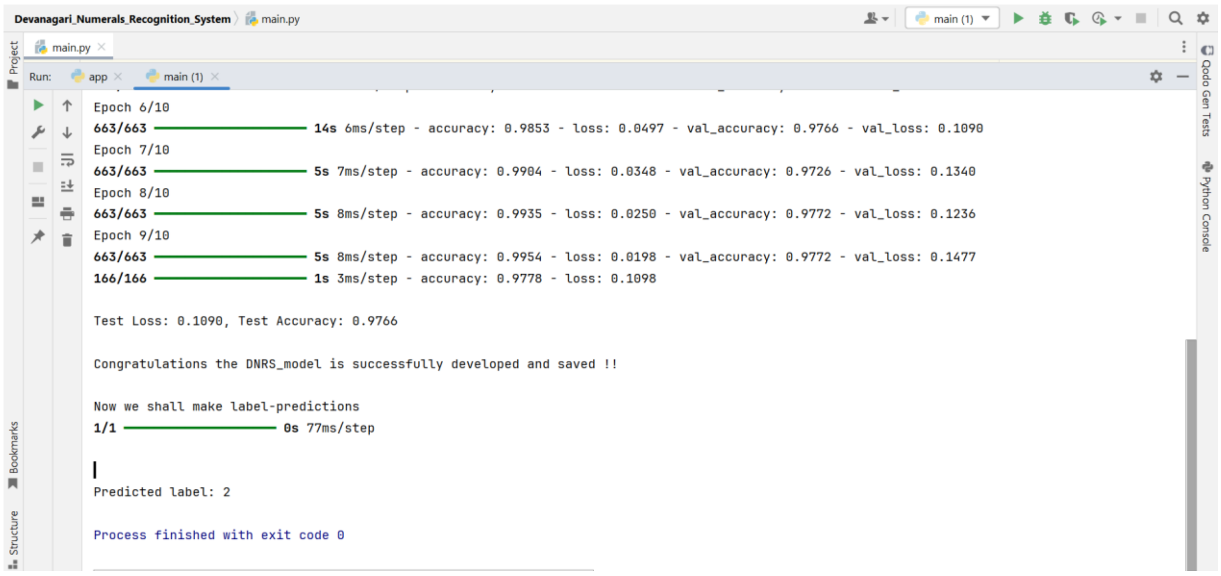
6.6. src/train.py

```
from tensorflow.keras.callbacks import EarlyStopping


def Train_Model(model, X_train, y_train, X_test, y_test,
epochs=10, batch_size=32):
    """Train the model on the training data."""

    # Early stopping to prevent overfitting
    early_stopping = EarlyStopping(monitor='val_loss',
patience=3, restore_best_weights=True)

    # Train the model
    model.fit(X_train, y_train, validation_data=(X_test,
y_test),
          epochs=epochs, batch_size=batch_size,
callbacks=[early_stopping])

    # Save the model after training
    model.save('Devanagari_Numerals_Recognition_Model.keras')
```

7. Output



```
D:\PycharmProjects\Devanagari_Numerals_Recognition_System\.venv\Scripts\python.exe D:\PycharmProjects\Devanagari_Numerals_Recognition_System\main
2024-12-07 12:59:27.239488: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results
2024-12-07 12:59:28.593936: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on. You may see slightly different numerical results
Welcome to Devanagari Numerals Recognition System
Developed by Umang Goswami
---------------------
This is the presentation of how the system shall work
---------------------

First the DNRS_model shall be developed and saved
D:\PycharmProjects\Devanagari_Numerals_Recognition_System\.venv\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: D
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
2024-12-07 12:59:37.720185: I tensorflow/core/platform/cpu_feature_guard.cc:210] This TensorFlow binary is optimized to use available CPU instruc
To enable the following instructions: AVX2 AVX_VNNI FMA, in other operations, rebuild TensorFlow with the appropriate compiler flags.
Epoch 1/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 7s 9ms/step - accuracy: 0.8071 - loss: 0.6349 - val_accuracy: 0.9325 - val_loss: 0.2486
Epoch 2/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 6s 8ms/step - accuracy: 0.9466 - loss: 0.1958 - val_accuracy: 0.9558 - val_loss: 0.1589
Epoch 3/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 6s 9ms/step - accuracy: 0.9695 - loss: 0.1123 - val_accuracy: 0.9642 - val_loss: 0.1362
Epoch 4/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 4s 6ms/step - accuracy: 0.9815 - loss: 0.0736 - val_accuracy: 0.9717 - val_loss: 0.1203
Epoch 5/10
```



```
Epoch 6/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 14s 6ms/step - accuracy: 0.9853 - loss: 0.0497 - val_accuracy: 0.9766 - val_loss: 0.1090
Epoch 7/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 5s 7ms/step - accuracy: 0.9904 - loss: 0.0348 - val_accuracy: 0.9726 - val_loss: 0.1340
Epoch 8/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - accuracy: 0.9935 - loss: 0.0250 - val_accuracy: 0.9772 - val_loss: 0.1236
Epoch 9/10
663/663 ━━━━━━━━━━━━━━━━━━━━ 5s 8ms/step - accuracy: 0.9954 - loss: 0.0198 - val_accuracy: 0.9772 - val_loss: 0.1477
166/166 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9778 - loss: 0.1098

Test Loss: 0.1090, Test Accuracy: 0.9766

Congratulations the DNRS_model is successfully developed and saved !!

Now we shall make label-predictions
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 77ms/step

|
Predicted label: 2

Process finished with exit code 0
```

8. References

- Handwritten Devanāgarī characters - vowels and numerals (38,750 isolated images + 9,200 isolated images). (2023). [Dataset]. In *Mendeley Data* (Version 4). doi:10.17632/pxrnvp4yy8.4 Retrived from https://data.mendeley.com/datasets/pxrnvp4yy8/4

- Prashanth, D. S., Mehta, R. V. K., & Challa, N. P. (2021). A multi-purpose dataset of Devanāgarī script comprising of isolated numerals and vowels. *Data in Brief*, *40*, 107723. https://doi.org/10.1016/j.dib.2021.107723 Retrived from https://pdf.sciencedirectassets.com/311593/1-s2.0-S2352340921X00078/1-s2.0-S2352340921009987/main.pdf