**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 Submitted to

UMANG GOSWAMI Prof. Avadesh Sharma

**TABLE OF CONTENTS**

[1. Abstract 1](#_Toc197284422)

[2. Technologies Used 1](#_Toc197284423)

[3. All Libraries used – requirements.txt 3](#_Toc197284424)

[4. System Requirements 4](#_Toc197284425)

[4.1. Hardware Requirements: 4](#_Toc197284426)

[4.2. Software Requirements: 5](#_Toc197284427)

[5. Dataset used for training 5](#_Toc197284428)

[6. Source Code 6](#_Toc197284429)

[6.1. main.py 6](#_Toc197284430)

[6.2. src/data\_loader.py 7](#_Toc197284431)

[6.3. src/ evaluate.py 8](#_Toc197284432)

[6.4. src/model.py 8](#_Toc197284433)

[6.5. src/predict.py 8](#_Toc197284434)

[6.6. src/train.py 9](#_Toc197284435)

[7. Output 10](#_Toc197284436)

[8. References 11](#_Toc197284437)

**Devanāgarī Numerals Recognition System**

# 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.

# 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

# 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**

# System Requirements

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

## 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.

## 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.

# 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*](https://doi.org/10.17632/pxrnvp4yy8.4) *Retrived from* [*https://data.mendeley.com/datasets/pxrnvp4yy8/4*](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*](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*](https://pdf.sciencedirectassets.com/311593/1-s2.0-S2352340921X00078/1-s2.0-S2352340921009987/main.pdf)

# Source Code

## 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()**

## 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**

## 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}')**

## 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**

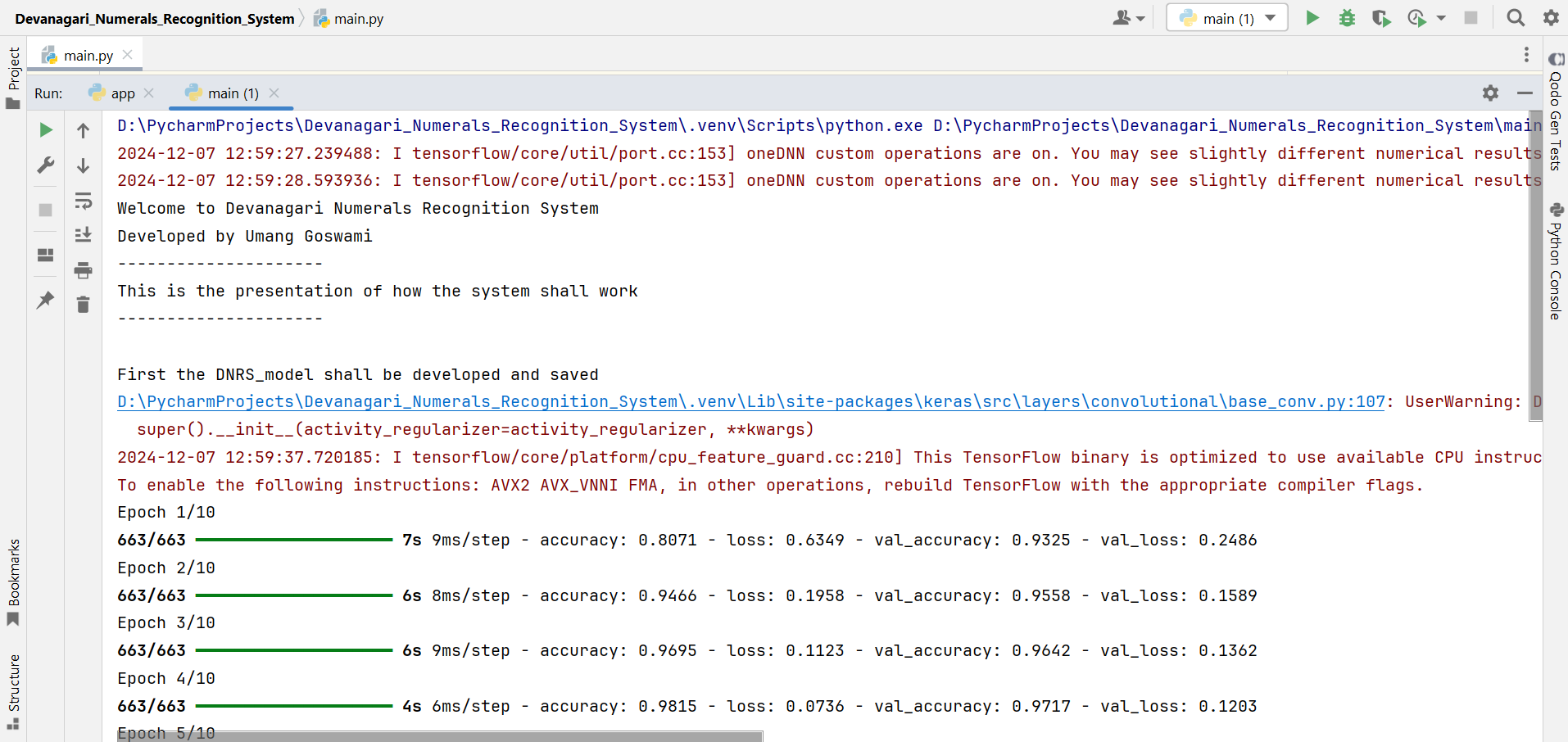
## src/predict.py

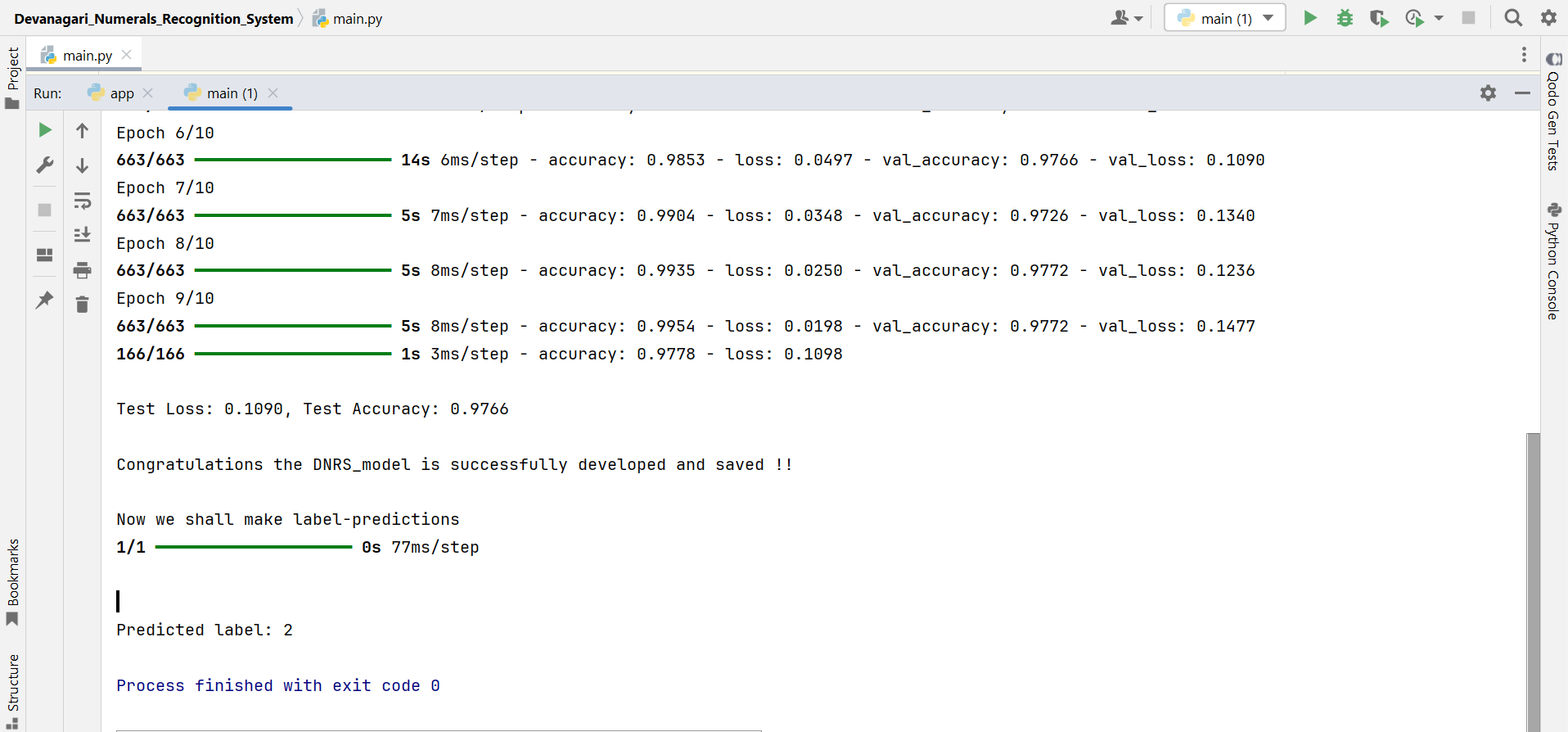
**import cv2  
import numpy as np  
  
  
def Load\_and\_Preprocess\_Image(image\_path):  
 *"""Load and preprocess the image for prediction."""* image = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)  
 image = cv2.resize(image, (28, 28))  
 image = image.astype('float32') / 255.0  
 image = image.reshape(-1, 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**

## 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')**

# Output





# 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](https://doi.org/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>