

```
In [3]: !unzip "/content/Copy of devnagari digit.zip"
```


[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

[illegible]

inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99799.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99800.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99802.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99803.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99804.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99805.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99806.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99808.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99809.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99811.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99812.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99813.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99814.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99815.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99816.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99817.png
inflating: DevanagariHandwrittenDigitDataset/Train/digit_9/99818.png

```
In [4]: import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.utils import to_categorical
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
from PIL import Image # Import Pillow

# Define dataset paths
train_dir = "/content/DevanagariHandwrittenDigitDataset/Train"
test_dir = "/content/DevanagariHandwrittenDigitDataset/Test"

# Define image size
img_height, img_width = 28, 28

# Function to load images and labels using PIL
def load_images_from_folder(folder):
    images = []
    labels = []
    class_names = sorted(os.listdir(folder)) # Sorted class names (digit_0,
    class_map = {name: i for i, name in enumerate(class_names)} # Map class

    for class_name in class_names:
        class_path = os.path.join(folder, class_name)
        label = class_map[class_name]

        # Loop through each image in the class folder
        for filename in os.listdir(class_path):
            img_path = os.path.join(class_path, filename)

            # Load image using PIL
            img = Image.open(img_path).convert("L") # Convert to grayscale
            img = img.resize((img_width, img_height)) # Resize to (28,28)
            img = np.array(img) / 255.0 # Normalize pixel values to [0,1]

            images.append(img)
            labels.append(label)
```

```

    return np.array(images), np.array(labels)

# Load training and testing datasets
x_train, y_train = load_images_from_folder(train_dir)
x_test, y_test = load_images_from_folder(test_dir)

# Reshape images for Keras input
x_train = x_train.reshape(-1, img_height, img_width, 1) # Shape (num_sample
x_test = x_test.reshape(-1, img_height, img_width, 1)

# One-hot encode labels
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

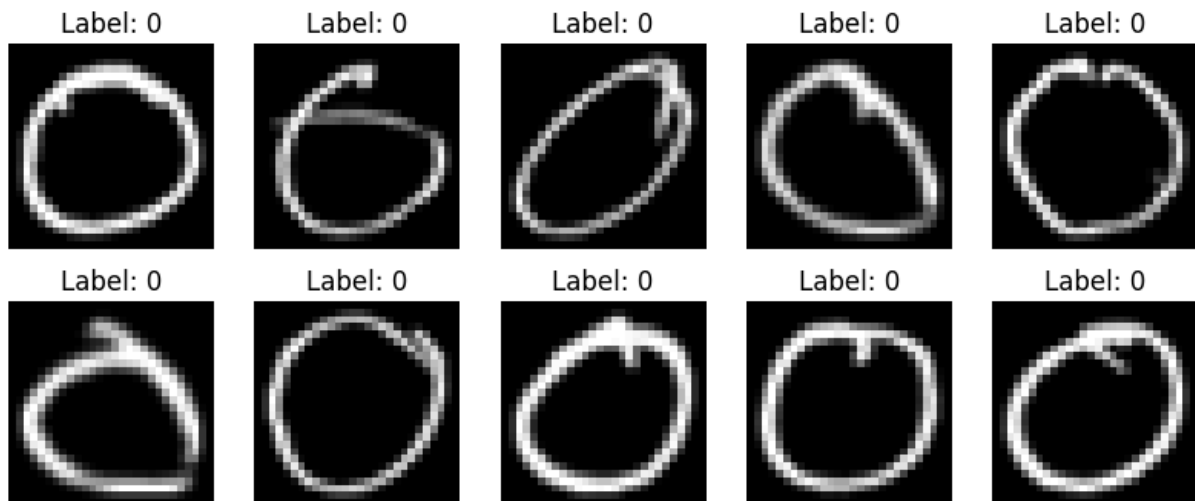
# Print dataset shape
print(f"Training set: {x_train.shape}, Labels: {y_train.shape}")
print(f"Testing set: {x_test.shape}, Labels: {y_test.shape}")

# Visualize some images
plt.figure(figsize=(10, 4))
for i in range(10):
    plt.subplot(2, 5, i + 1)
    plt.imshow(x_train[i].reshape(28, 28), cmap='gray')
    plt.title(f"Label: {np.argmax(y_train[i])}")
    plt.axis("off")
plt.show()

```

Training set: (17000, 28, 28, 1), Labels: (17000, 10)

Testing set: (3000, 28, 28, 1), Labels: (3000, 10)



```

In [ ]: x_train = x_train.reshape(-1, img_height, img_width, 1)
        x_test = x_test.reshape(-1, img_height, img_width, 1)

```

```

In [9]: # Define a simple Fully Connected Neural Network (FCN)
model = Sequential([
    Flatten(input_shape=(28, 28)), # Flatten input images (28x28 pixels)
    Dense(128, activation='relu'), # Fully connected layer with ReLU
    Dense(64, activation='relu'),
    Dense(10, activation='softmax') # Output layer with softmax for 10 clas
])

```

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

# Print the model summary
model.summary()
```

/usr/local/lib/python3.11/dist-packages/keras/src/layers/reshaping/flatten.py:37: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using an `Input(shape)` object as the first layer in the model instead.

```
super().__init__(**kwargs)
```

Model: "sequential"

| Layer (type) | Output Shape |
|-------------------|--------------|
| flatten (Flatten) | (None, 784) |
| dense (Dense) | (None, 128) |
| dense_1 (Dense) | (None, 64) |
| dense_2 (Dense) | (None, 10) |

Total params: 109,386 (427.29 KB)

Trainable params: 109,386 (427.29 KB)

Non-trainable params: 0 (0.00 B)

```
In [10]: # Load MNIST dataset
mnist = keras.datasets.mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()


# Normalize the data
x_train, x_test = x_train / 255.0, x_test / 255.0

# Train the model
model.fit(x_train, y_train, epochs=5, validation_data=(x_test, y_test))
```


Downloading data from <https://storage.googleapis.com/tensorflow/tf-keras-datasets/mnist.npz>

11490434/11490434  **0s** 0us/step


Epoch 1/5

1875/1875  **8s** 3ms/step - accuracy: 0.8731 - loss: 0.4337
- val_accuracy: 0.9555 - val_loss: 0.1417


Epoch 2/5

1875/1875  **5s** 2ms/step - accuracy: 0.9676 - loss: 0.1068
- val_accuracy: 0.9685 - val_loss: 0.0969


Epoch 3/5

1875/1875  **6s** 3ms/step - accuracy: 0.9779 - loss: 0.0720
- val_accuracy: 0.9746 - val_loss: 0.0821

Epoch 4/5

1875/1875  **5s** 2ms/step - accuracy: 0.9837 - loss: 0.0529
- val_accuracy: 0.9752 - val_loss: 0.0782

Epoch 5/5

1875/1875  **5s** 3ms/step - accuracy: 0.9885 - loss: 0.0368
- val_accuracy: 0.9745 - val_loss: 0.0840

Out[10]: <keras.src.callbacks.history.History at 0x7a1f4dcd7310>

```
In [11]: relu_layer = Dense(128, activation='relu')
```

```
In [12]: softmax_layer = Dense(10, activation='softmax')
```

```
In [15]: model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['a

binary_model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')
])

binary_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[
```

```
In [16]: model.compile(optimizer='sgd', loss='sparse_categorical_crossentropy', metri
```

```
In [17]: model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_
```

```
In [18]: model.compile(optimizer=Adam(learning_rate=0.001), loss='sparse_categorical_
```

```
In [19]: model = Sequential([
    Flatten(input_shape=(28, 28)),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metr
```

```
In [20]: datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
```

```

        height_shift_range=0.2,
        horizontal_flip=True,
        rescale=1./255
    )

augmented_data = datagen.flow(x_train.reshape(-1, 28, 28, 1), y_train, batch

```

```

In [21]: from tensorflow.keras.applications import VGG16

base_model = VGG16(weights='imagenet', include_top=False, input_shape=(224,

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

transfer_model = Sequential([
    base_model,
    Flatten(),
    Dense(256, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])

transfer_model.compile(optimizer='adam', loss='categorical_crossentropy', me

transfer_model.summary()

```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
 58889256/58889256 ————— 0s 0us/step
Model: "sequential_5"

| Layer (type) | Output Shape |
|---------------------------------------|-------------------|
| vgg16 (Functional) | (None, 7, 7, 512) |
| flatten_5 (Flatten) | (None, 25088) |
| dense_14 (Dense) | (None, 256) |
| dropout_1 (Dropout) | (None, 256) |
| dense_15 (Dense) | (None, 10) |

Total params: 21,140,042 (80.64 MB)
Trainable params: 6,425,354 (24.51 MB)
Non-trainable params: 14,714,688 (56.13 MB)