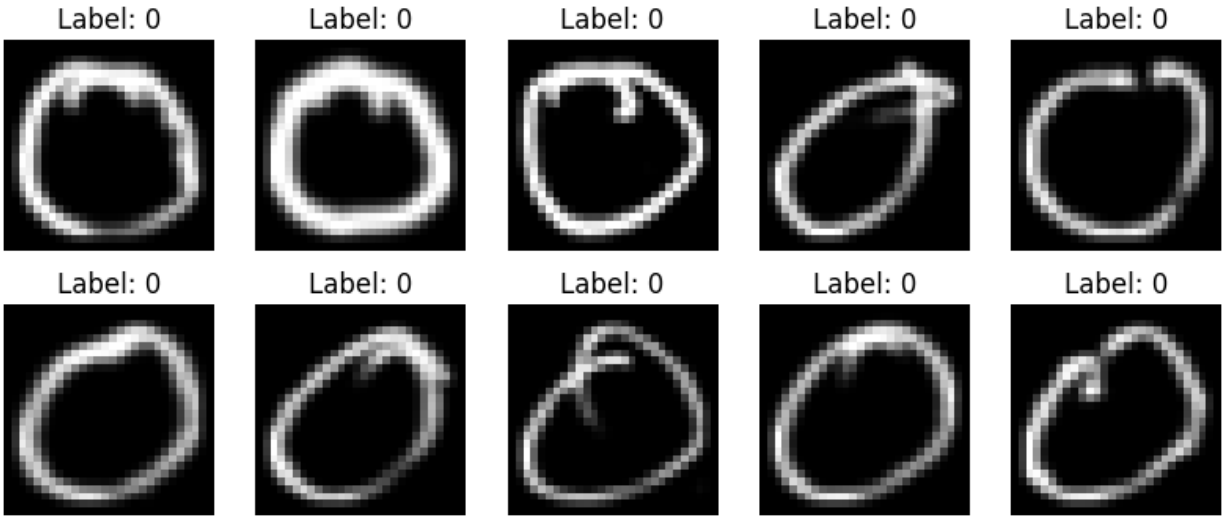
#Task 1: Data Preparation

```
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

# Training and testing directory
train_dir = "/media/samyog/My Folder/AI and
ML/AI-and-ML/Worksheet_04/DevanagariHandwrittenDigitDataset/Train"
test_dir = "/media/samyog/My Folder/AI and
ML/AI-and-ML/Worksheet_04/DevanagariHandwrittenDigitDataset/Test"

# Defining the 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([name for name in os.listdir(folder) if
os.path.isdir(os.path.join(folder, name))])
    print(f"Class names: {class_names}")
    class_map = { name: i for i, name in enumerate(class_names) }
    for class_name in class_names:
        class_path = os.path.join(folder, class_name)
        label = class_map[class_name]
        for filename in os.listdir(class_path):
            img_path = os.path.join(class_path, filename)
            try:
                img = Image.open(img_path).convert("L")
                img = img.resize((img_width, img_height))
                img = np.array(img) / 255.0
                if img.shape != (img_height, img_width):
                    print(f"Skipping image {img_path}: incorrect shape
{img.shape}")
                    continue
                images.append(img)
                labels.append(label)
            except Exception as e:
                print(f"Error loading image {img_path}: {e}")
                continue
    images = np.array(images, dtype=np.float32)
    labels = np.array(labels, dtype=np.int32)
    print(f"Loaded {len(images)} images with shape {images.shape},
labels shape {labels.shape}")
```

```
    return images, 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)
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()
```

```
Class names: ['digit_0', 'digit_1', 'digit_2', 'digit_3', 'digit_4',
'digit_5', 'digit_6', 'digit_7', 'digit_8', 'digit_9']
Loaded 17000 images with shape (17000, 28, 28), labels shape (17000,)
Class names: ['digit_0', 'digit_1', 'digit_2', 'digit_3', 'digit_4',
'digit_5', 'digit_6', 'digit_7', 'digit_8', 'digit_9']
Loaded 3000 images with shape (3000, 28, 28), labels shape (3000,)
Training set: (17000, 28, 28, 1), Labels: (17000, 10)
Testing set: (3000, 28, 28, 1), Labels: (3000, 10)
```



#Task 2: Build the FCN Model

```
import tensorflow as tf
from tensorflow import keras

num_classes = 10
input_shape = (28, 28, 1)
model = keras.Sequential([
    keras.layers.Input(shape=input_shape),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation="relu"),  # Changed to relu for
better performance
    keras.layers.Dense(128, activation="relu"),
    keras.layers.Dense(256, activation="relu"),
    keras.layers.Dense(num_classes, activation="softmax"),
])
```

```
E0000 00:00:1743052218.406952   88211 cuda_executor.cc:1228] INTERNAL:
CUDA Runtime error: Failed call to cudaGetRuntimeVersion: Error
loading CUDA libraries. GPU will not be used.: Error loading CUDA
libraries. GPU will not be used.
W0000 00:00:1743052218.410588   88211 gpu_device.cc:2341] Cannot
dlopen some GPU libraries. Please make sure the missing libraries
mentioned above are installed properly if you would like to use GPU.
Follow the guide at https://www.tensorflow.org/install/gpu for how to
download and setup the required libraries for your platform.
Skipping registering GPU devices...
```

```
model.summary()
```

```
Model: "sequential"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃
```

```
┃ Layer (type)                    ┃ Output Shape           ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━
━━━━┩
│ flatten (Flatten)               │ (None, 784)            │
0 │
├─────────────────────────────────┼────────────────────────┼─────────
────┤
│ dense (Dense)                   │ (None, 64)             │
50,240 │
├─────────────────────────────────┼────────────────────────┼─────────
────┤
│ dense_1 (Dense)                 │ (None, 128)            │
8,320 │
├─────────────────────────────────┼────────────────────────┼─────────
────┤
│ dense_2 (Dense)                 │ (None, 256)            │
33,024 │
├─────────────────────────────────┼────────────────────────┼─────────
────┤
│ dense_3 (Dense)                 │ (None, 10)             │
2,570 │
└─────────────────────────────────┴────────────────────────┴─────────
────┘

 Total params: 94,154 (367.79 KB)

 Trainable params: 94,154 (367.79 KB)

 Non-trainable params: 0 (0.00 B)
```

#Task 3: Compile the Model

###Compiling the Model

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

#Task 4: Train the Model

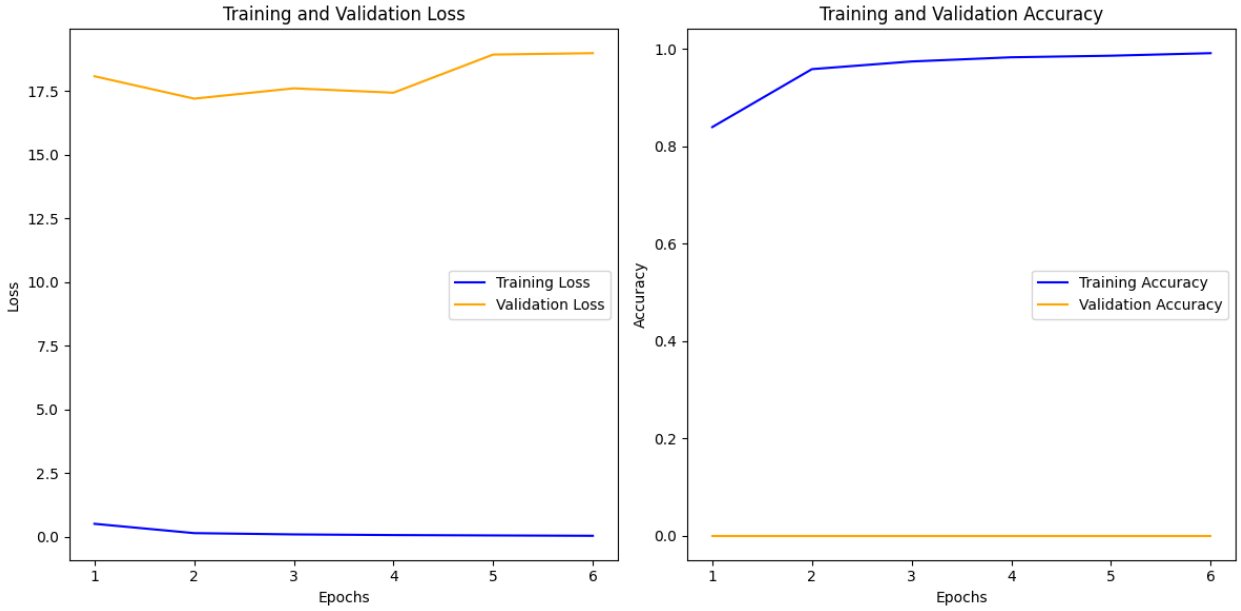
```
batch_size = 128
epochs = 20

callbacks = [

keras.callbacks.ModelCheckpoint(filepath="model_at_epoch_{epoch}.keras
"),
```

```
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=4),
]

history = model.fit(
    x_train,
    y_train,
    batch_size=batch_size,
    epochs=epochs,
    validation_split=0.2,
    callbacks=callbacks,
)
```

```
Epoch 1/20

2025-03-27 10:55:35.482294: W
external/local_xla/xla/tsl/framework/cpu_allocator_impl.cc:83]
Allocation of 42649600 exceeds 10% of free system memory.

107/107 ━━━━━━━━━━━━━━━━━━━━ 3s 9ms/step - accuracy: 0.6783 - loss:
0.9805 - val_accuracy: 0.0000e+00 - val_loss: 18.0704
Epoch 2/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 9ms/step - accuracy: 0.9574 - loss:
0.1436 - val_accuracy: 0.0000e+00 - val_loss: 17.1919
Epoch 3/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9699 - loss:
0.0959 - val_accuracy: 0.0000e+00 - val_loss: 17.5965
Epoch 4/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step - accuracy: 0.9842 - loss:
0.0566 - val_accuracy: 0.0000e+00 - val_loss: 17.4247
Epoch 5/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 14ms/step - accuracy: 0.9844 - loss:
0.0545 - val_accuracy: 0.0000e+00 - val_loss: 18.9224
Epoch 6/20
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 11ms/step - accuracy: 0.9905 - loss:
0.0318 - val_accuracy: 0.0000e+00 - val_loss: 18.9769
```

```
# Plot training and validation metrics
import matplotlib.pyplot as plt

train_loss = history.history['loss']
val_loss = history.history['val_loss']
train_acc = history.history.get('accuracy', [])
val_acc = history.history.get('val_accuracy', [])

plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(range(1, len(train_loss) + 1), train_loss, label="Training
Loss", color="blue")
plt.plot(range(1, len(val_loss) + 1), val_loss, label="Validation
Loss", color="orange")
```

```
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.title("Training and Validation Loss")
plt.legend()

plt.subplot(1, 2, 2)
if train_acc and val_acc:
    plt.plot(range(1, len(train_acc) + 1), train_acc, label="Training
Accuracy", color="blue")
    plt.plot(range(1, len(val_acc) + 1), val_acc, label="Validation
Accuracy", color="orange")
    plt.xlabel("Epochs")
    plt.ylabel("Accuracy")
    plt.title("Training and Validation Accuracy")
    plt.legend()

plt.tight_layout()
plt.show()
```



#Task 5: Evaluate the Model

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print(f"Test accuracy: {test_acc:.4f}")

94/94 - 1s - 7ms/step - accuracy: 0.7823 - loss: 3.8178
Test accuracy: 0.7823
```

#Task 6: Save and Load the Model

###1. Saving the Model:

```
model.save("mnist_fully_connected_model.keras")
```

###2. Loading the Model:

```
loaded_model =
tf.keras.models.load_model("mnist_fully_connected_model.keras")
```

#Task 7: Predictions

```
predictions = model.predict(x_test)
predicted_labels = np.argmax(predictions, axis=1)
print(f"Predicted label for first image: {predicted_labels[0]}")
print(f"True label for first image: {np.argmax(y_test[0])}")
```

```
94/94 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Predicted label for first image: 0
True label for first image: 0
```