# System Architecture & Design of Handwritten Digit Recognition ML model

**Framework and Libraries**: We uses TensorFlow/Keras, which means our model relies on a Convolutional Neural Network (CNN), to handle image classification tasks.

## Model Architecture Definition:

1. We used neural network using Keras, specifying layers such as Conv2D, MaxPooling2D, Flatten, Dense, etc.
2. This part constitutes the core algorithm design, which handles feature extraction (via convolutional layers) and classification (via dense layers).

```
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')  # 10 classes
])
```

**Training Process:** Training Algorithm: The training process uses TensorFlow's backpropagation algorithm to optimize the weights of the network, typically with an optimizer like Adam, SGD, or RMSProp.

This section specifies the:

- Loss function: Guides how the model learns (e.g., sparse_categorical_crossentropy for classification tasks).
- Metrics: Tracks model performance during training (accuracy).

```
from tensorflow.keras import layers, models
model = models.Sequential([
    layers.Conv2D(6, (5, 5), activation='relu', input_shape=(128, 128, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(16, (5, 5), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(120, activation='relu'),
    layers.Dense(84, activation='relu'),
    layers.Dense(10, activation='softmax')  # Use '10' if you have 10 classes (0-9 digits)
])

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

## Preprocessing Pipeline:

The preprocessing pipeline is part of the notebook and ensures images are properly fed into the model:

1. Resizing and Normalization.
2. Splitting into train/test sets.

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
train_dir = 'dataset/train'
test_dir = 'dataset/test'

# ImageDataGenerator for data augmentation
```

```
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

# Load images from directories
train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size=(128, 128),
    color_mode='grayscale',
    batch_size=32,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    test_dir,
    target_size=(128, 128),
    color_mode='grayscale',
    batch_size=32,
    class_mode='categorical'
)
print("Class indices:", train_generator.class_indices)
print("Classes:", train_generator.classes)
```

### Normalization:

Pixel values (0–255) are scaled to the range [0, 1].

```
train_data = train_data.map(lambda x, y: (x / 255.0, y))
test_data = test_data.map(lambda x, y: (x / 255.0, y))
```

### Shuffling and Batching:

Ensures the training data is randomized for better generalization.

```
train_data = train_data.shuffle(buffer_size=1000).batch(32)
test_data = test_data.batch(32)
```

Integration of Preprocessing With Model Architecture

- Input Layer: Expects preprocessed images of size (128x128x3).

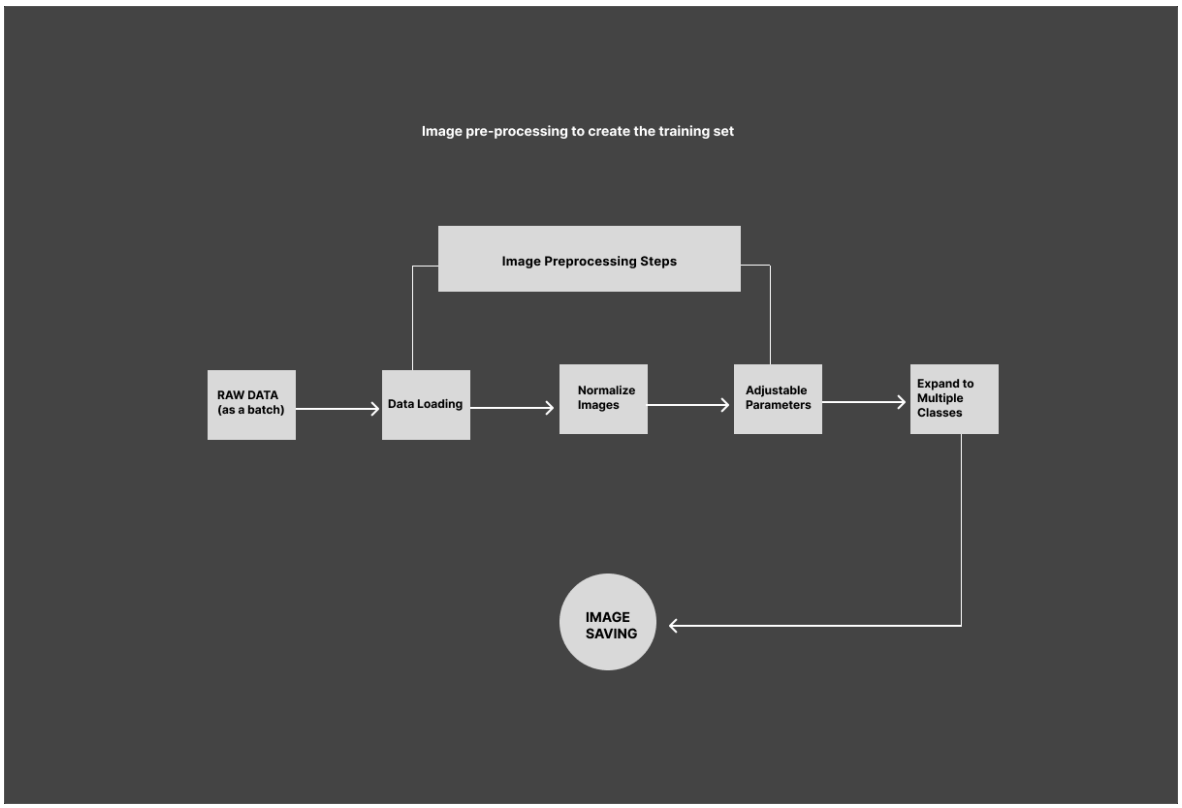


**Figure 1.0: Image pre-processing to create the training image set**

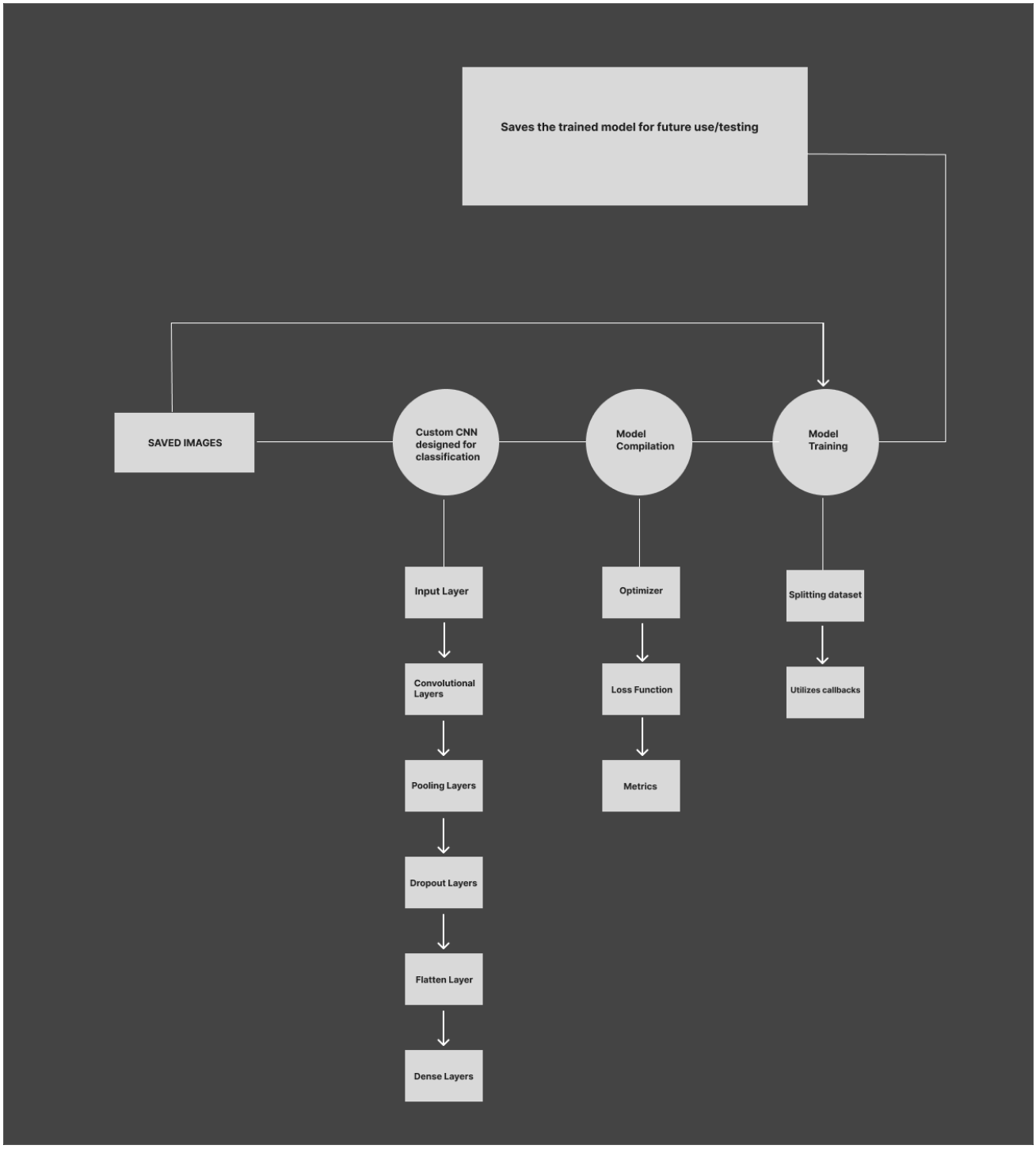


**Figure 2.0: Simplified Architecture of a Convolution Neural Network for Handwritten Digit Recognition.**