



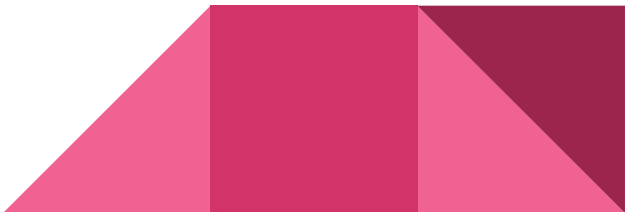
# **Classification of Handwritten Digits Using CNN Using MNIST Dataset**

# Team

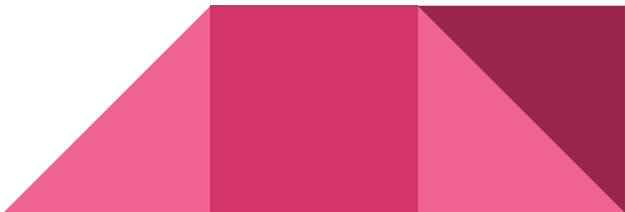
- **Haripreeth Dwarakanath Avarur (RA2011003010011)**
- **Kartik Paliwal (RA2011003010013)**
- **Aditi (RA2011003010004)**



# Societal Benefits

- **Improved automation:** Handwritten digit recognition can be used in various industries, such as banking, where checks and other handwritten documents need to be processed automatically. By using automated recognition systems based on CNN, we can improve the speed and accuracy of processing such documents.
  - **Medical Applications:** The use of handwritten digits recognition using CNN can have a significant impact on medical diagnosis, especially in the field of pathology, where doctors often examine images of cells and tissues to make diagnoses. CNN-based systems can help identify patterns and structures in these images, leading to more accurate and timely diagnosis.
- 

# Societal Benefits

- **Educational Tools:** Handwritten digit recognition can also be used as an educational tool to help children learn to write and recognize numbers. By providing instant feedback and visual examples, children can improve their handwriting skills and learn more effectively.
  - **Accessibility:** Digit recognition can be used as a tool to aid people with disabilities, such as those who have difficulty writing or reading. By using digit recognition systems, people can communicate and access information more easily.
  - **Crime Detection:** Digit recognition can also be used as a tool in forensic science to identify handwriting in documents and notes that may be related to criminal activities.
- 

# Problem Statement

- The Problem Statement is to develop a machine learning model using Convolutional Neural Networks (CNN) to accurately classify handwritten digits from the MNIST dataset. The goal is to create a system that can automatically recognize and classify digits from 0 to 9 with high accuracy. This model can be used in a variety of applications, including automating handwritten digit recognition tasks in industries such as banking, improving medical diagnosis, and aiding in educational tools. The project aims to build a reliable and accurate digit recognition system that can have a positive impact on society by making tasks more efficient, accessible, and effective.




# Objective


The objective of this project is to develop a machine learning model using Convolutional Neural Networks (CNN) to accurately classify handwritten digits from the MNIST dataset. The goal is to create a system that can automatically recognize and classify digits from 0 to 9 with high accuracy.

## Technical Depth:

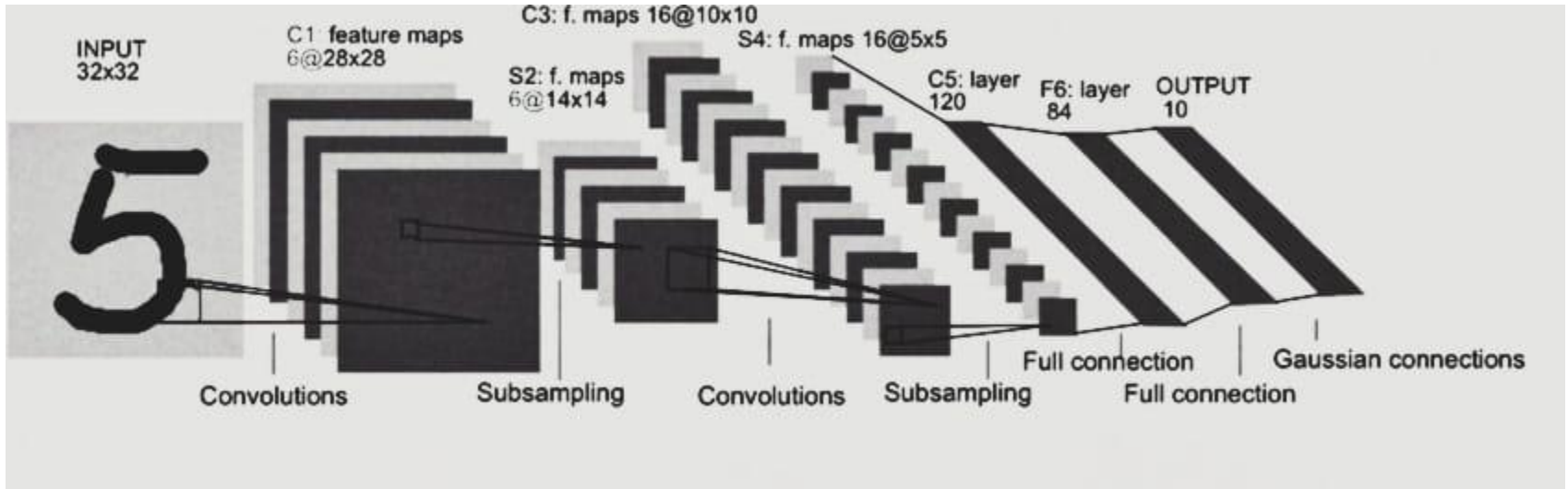
To achieve the objective, the project requires a good understanding of various machine learning concepts and techniques, including:

- **Convolutional Neural Networks (CNN)**
  - **Preprocessing Techniques**
  - **Training and Optimization**
  - **Evaluation and Fine-tuning**
- 

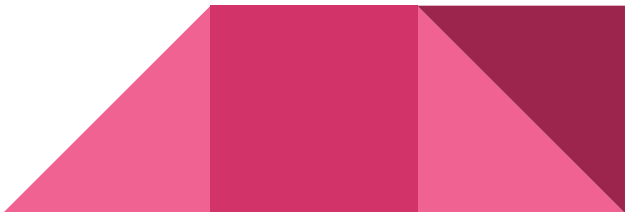
# Technical Depth

- **Convolutional Neural Networks (CNN):** CNN is a type of deep neural network that is widely used for image recognition tasks. The project will require a good understanding of CNNs, including the various layers involved and their functions.
  - **Preprocessing Techniques:** The MNIST dataset contains a large number of handwritten digit images. Before training a machine learning model, the images need to be preprocessed to remove noise and enhance features. The project will require a good understanding of various preprocessing techniques, such as normalization, filtering, and augmentation.
  - **Training and Optimization:** The machine learning model needs to be trained on the preprocessed dataset. The project will require a good understanding of various optimization techniques, such as backpropagation and stochastic gradient descent, to train the model effectively and efficiently.
  - **Evaluation and Fine-tuning:** After training the model, it needs to be evaluated using various metrics to assess its accuracy and performance. The project will require a good understanding of various evaluation techniques, such as confusion matrix and accuracy, to evaluate the model's performance. Fine-tuning techniques, such as hyperparameter tuning and regularization, can also be used to improve the model's performance.
- 

# Diagram





- **Input Data:** This component represents the MNIST handwritten digits dataset, which contains 60,000 training images and 10,000 testing images of handwritten digits from 0 to 9.
  - **Preprocessing Layers:** This component represents the preprocessing techniques used to clean and enhance the input data, including normalization, filtering, and augmentation.
  - **Convolutional Layers:** This component represents the core of the CNN model, where features are extracted from the preprocessed input data through multiple convolutional layers.
  - **Pooling Layers:** This component performs downsampling to reduce the dimensions of the feature maps and make the model computationally efficient.
  - **Fully Connected Layers:** This component represents the final layers of the CNN model, where the extracted features are classified into different digit classes.
  - **Output:** This component represents the predicted digit class for a
- 

# Half of the Code

```
import tensorflow as tf
from tensorflow import keras

# Load the MNIST dataset
mnist = keras.datasets.mnist
(train_images, train_labels), (test_images, test_labels) = mnist.load_data()

# Preprocess the data
train_images = train_images.reshape((60000, 28, 28, 1))
train_images = train_images / 255.0
test_images = test_images.reshape((10000, 28, 28, 1))
test_images = test_images / 255.0

# Define the model architecture
model = keras.Sequential([
    keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(28,28,1)),
    keras.layers.MaxPooling2D((2,2)),
    keras.layers.Conv2D(64, (3,3), activation='relu'),
    keras.layers.MaxPooling2D((2,2)),
    keras.layers.Conv2D(64, (3,3), activation='relu'),
    keras.layers.Flatten(),
    keras.layers.Dense(64, activation='relu'),
    keras.layers.Dense(10)
])

# Compile the model
model.compile(optimizer='adam',
              loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])

# Train the model
model.fit(train_images, train_labels, epochs=5, validation_data=(test_images, test_labels))
```

# Output

```
Epoch 1/5
1875/1875 [=====] - 38s 20ms/step - loss: 0.1431 - accuracy: 0.9569 -
val_loss: 0.0446 - val_accuracy: 0.9851
Epoch 2/5
1875/1875 [=====] - 37s 20ms/step - loss: 0.0475 - accuracy: 0.9855 -
val_loss: 0.0367 - val_accuracy: 0.9876
Epoch 3/5
1875/1875 [=====] - 36s 19ms/step - loss: 0.0347 - accuracy: 0.9890 -
val_loss: 0.0289 - val_accuracy: 0.9903
Epoch 4/5
1875/1875 [=====] - 36s 19ms/step - loss: 0.0271 - accuracy: 0.9916 -
val_loss: 0.0336 - val_accuracy: 0.9894
Epoch 5/5
1875/1875 [=====] - 36s 19ms/step - loss: 0.0215 - accuracy: 0.9928 -
val_loss: 0.0272 - val_accuracy: 0.9916
```