In [1]:
```
import tensorflow as tf
from tensorflow.keras.datasets import mnist
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
```

In [2]:
```
# Data Preprocessing
# Load MNIST dataset
(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/mni
st.npz
11490434/11490434 [==============================] - 105s 9us/step
```

In [3]:
```
# Reshape images to the required dimensions for the network
x_train = x_train.reshape(-1, 28, 28, 1)
x_test = x_test.reshape(-1, 28, 28, 1)
```

In [4]:
```
# Normalize images
x_train = x_train / 255.0
x_test = x_test / 255.0

# Convert labels to one-hot encoding format

y_train = tf.keras.utils.to_categorical(y_train, num_classes=10)
y_test = tf.keras.utils.to_categorical(y_test, num_classes=10)
```

In [5]:
```
# Network Architecture

# Create a Sequential object for defining the network architecture
model = Sequential()
# Add a Convolutional layer to the network with 32 filters of size 3x3 and ReLU activa
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 1)))
# Add a MaxPooling layer with pool size 2x2
model.add(MaxPooling2D((2, 2)))
# Flatten the data to a 1D vector
model.add(Flatten())
# Add a Dense layer with 10 neurons and softmax activation function for multi-class cl

model.add(Dense(10, activation='softmax'))
```

In [6]:
```
# Training

# Define the loss function and optimizer
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
# Train the network on the training data for 5 epochs with a batch size of 128 and 10%
model.fit(x_train, y_train, batch_size=128, epochs=5, validation_split=0.1)
```

```
Epoch 1/5
422/422 [==============================] - 6s 13ms/step - loss: 0.3839 - accuracy: 0.
8970 - val_loss: 0.1532 - val_accuracy: 0.9612
Epoch 2/5
422/422 [==============================] - 6s 13ms/step - loss: 0.1494 - accuracy: 0.
9578 - val_loss: 0.0958 - val_accuracy: 0.9740
Epoch 3/5
422/422 [==============================] - 6s 14ms/step - loss: 0.0985 - accuracy: 0.
9733 - val_loss: 0.0728 - val_accuracy: 0.9818
Epoch 4/5
422/422 [==============================] - 6s 14ms/step - loss: 0.0780 - accuracy: 0.
9783 - val_loss: 0.0654 - val_accuracy: 0.9827
Epoch 5/5
422/422 [==============================] - 6s 14ms/step - loss: 0.0664 - accuracy: 0.
9810 - val_loss: 0.0624 - val_accuracy: 0.9838
```

Out[6]: <keras.callbacks.History at 0x1f4644958e0>

In [7]:
```
# Evaluation

# Evaluate the network on the test data
loss, accuracy = model.evaluate(x_test, y_test)
# Print the test loss and accuracy
print('Test Loss:', loss)
print('Test Accuracy:', accuracy)
```

```
313/313 [==============================] - 7s 24ms/step - loss: 0.0662 - accuracy: 0.
9789
Test Loss: 0.06621025502681732
Test Accuracy: 0.9789000153541565
```

برای آموزش شبکه، در اولین دوره دقت آموزش 89.70% و دقت اعتبارسنجی 96.12% بوده است. سپس با ادامه آموزش، دقت آموزش و اعتبارسنجی افزایش یافته و در دوره‌های بعدی به دقت بالاتری رسیده است. در آخرین دوره آموزش، دقت آموزش 98.10% و دقت اعتبارسنجی 98.38% بوده است . برای ارزیابی شبکه روی داده‌های تست، دقت 97.89% و خطا (loss) 0.0662 گزارش شده است. با توجه به نتایج، شبکه CNN به خوبی عمل می‌کند و دقت قابل قبولی در تشخیص ارقام دست‌نویس را دارد.

نتایج شبکه به طور کلی بسیار خوب بوده است. دقت بالای 97٪ در مجموعه تست به معنای این است که شبکه با موفقیت قادر به تشخیص اعداد دست‌نویس در دیتاست MNIST است. همچنین، خطا (loss) کمی نیز گزارش شده است، که نشان می‌دهد شبکه در فرایند یادگیری به خوبی پیشرفت کرده است. با این حال، همیشه می‌توان بهبودی در عملکرد شبکه ایجاد کرد. برخی از راهکارهای ممکن برای بهبود عملکرد شبکه عبارتند از: افزایش عمق شبکه: با افزودن لایه‌های پیچشی و لایه‌های Fully Connected به شبکه، می‌توانید پیچیدگی مدل را افزایش داده و عملکرد آن را بهبود بخشید. تغییر هایپرپارامترها: هایپرپارامترهای مانند نرخ یادگیری، اندازه دسته (batch size) و تعداد دوره‌های آموزش (epochs) می‌توانند تأثیر زیادی بر عملکرد شبکه داشته باشند. با آزمایش و تنظیم این پارامترها می‌توانید به نتایج بهتری برسید. استفاده از تکنیک‌های منظم‌سازی (regularization): تکنیک‌هایی مانند Dropout و L2 regularization می‌توانند برای کاهش بیش‌برازش (overfitting) مدل و افزایش قدرت تعمیم‌پذیری آن مفید باشند. آزمون شبکه با دیتاهای جدید: اگر امکان دارد، می‌توانید شبکه خود را با دیتاهای جدید یا تعداد بیشتری از داده‌های تست ارزیابی کنید تا مطمئن شوید که عملکرد آن بهبود یافته است. پیاده‌سازی این تغییرات و آزمایش‌های مختلف می‌تواند کمک کند تا بهبودی در عملکرد شبکه دست‌یافت.