1. Implementation of CNN model on MNIST dataset

Source code:

import tensorflow as tf

from tensorflow.keras import layers, models

from tensorflow.keras.datasets import mnist

# Step 2: Load the MNIST dataset

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

# Step 3: Preprocess the data

train\_images = train\_images.reshape((60000, 28, 28, 1))

train\_images = train\_images.astype('float32') / 255

test\_images = test\_images.reshape((10000, 28, 28, 1))

test\_images = test\_images.astype('float32') / 255

# Step 4: Define the CNN model

model = models.Sequential([

layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.MaxPooling2D((2, 2)),

layers.Conv2D(64, (3, 3), activation='relu'),

layers.Flatten(),

layers.Dense(64, activation='relu'),

layers.Dense(10, activation='softmax')

])

# Step 5: Compile the model

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Step 6: Train the model

model.fit(train\_images, train\_labels, epochs=5, batch\_size=64)

# Step 7: Evaluate the model

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('Test accuracy:', test\_acc)

lenet

# prompt: now perform lenet

# Step 1: Define the LeNet model

model = models.Sequential([

layers.Conv2D(6, (5, 5), activation='sigmoid', input\_shape=(28, 28, 1)),

layers.AvgPool2D(),

layers.Conv2D(16, (5, 5), activation='sigmoid'),

layers.AvgPool2D(),

layers.Flatten(),

layers.Dense(120, activation='sigmoid'),

layers.Dense(84, activation='sigmoid'),

layers.Dense(10, activation='softmax')

])

# Step 2: Compile the model

model.compile(optimizer='adam',

loss='sparse\_categorical\_crossentropy',

metrics=['accuracy'])

# Step 3: Train the model

model.fit(train\_images, train\_labels, epochs=5, batch\_size=64)

# Step 4: Evaluate the model

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('Test accuracy:', test\_acc)

vgg16

# prompt: vgg16 for mnist dataset

from tensorflow.keras.layers import Dense, Flatten, Conv2D, MaxPooling2D

from tensorflow.keras.models import Sequential

# Define the VGG16 model

model = Sequential()

model.add(Conv2D(64, (3, 3), activation='relu', padding='same', input\_shape=(28, 28, 1)))

model.add(Conv2D(64, (3, 3), activation='relu', padding='same'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(128, (3, 3), activation='relu', padding='same'))

model.add(MaxPooling2D((2, 2)))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))

model.add(Conv2D(256, (3, 3), activation='relu', padding='same'))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(1024, activation='relu'))

model.add(Dense(1024, activation='relu'))

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

# Compile the model

model.compile(loss='sparse\_categorical\_crossentropy', optimizer='adam', metrics=['accuracy'])

# Train the model

model.fit(train\_images, train\_labels, epochs=5, batch\_size=64)

# Evaluate the model

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('Test accuracy:', test\_acc)