Name - Chirag N

Class - 4MScDS-A

Reg No - 23122013

```
from __future__ import absolute_import, division, print_function

import tensorflow as tf
from tensorflow.keras import Model, layers
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image
import io
from google.colab import files  # For file upload in Google Colab

num_classes = 10
num_features = 784

learning_rate = 0.1
training_steps = 500
batch_size = 256
display_step = 100

n_hidden_1 = 128
n_hidden_2 = 256

from tensorflow.keras.datasets import mnist
(x_train, y_train), (x_test, y_test) = mnist.load_data()

x_train = np.array(x_train, np.float32).reshape([-1, num_features]) / 255.
x_test = np.array(x_test, np.float32).reshape([-1, num_features]) / 255.

def update_train_data(batch_size):
    train_data = tf.data.Dataset.from_tensor_slices((x_train, y_train))
    train_data = train_data.repeat().shuffle(5000).batch(batch_size).prefetch(1)
    return train_data

train_data = update_train_data(batch_size)

def create_model(activation=tf.nn.relu):
    class NeuralNet(Model):
        def __init__(self):
            super(NeuralNet, self).__init__()
            self.fc1 = layers.Dense(n_hidden_1, activation=activation)
            self.fc2 = layers.Dense(n_hidden_2, activation=activation)
            self.out = layers.Dense(num_classes)

        def call(self, x, is_training=False):
            x = self.fc1(x)
```

```
            x = self.fc2(x)
            x = self.out(x)
            return x

    return NeuralNet()

neural_net = create_model()

def cross_entropy_loss(x, y):
    y = tf.cast(y, tf.int64)
    loss = tf.nn.sparse_softmax_cross_entropy_with_logits(labels=y, logits=x)
    return tf.reduce_mean(loss)

def accuracy(y_pred, y_true):
    y_pred = tf.nn.softmax(y_pred)
    correct_prediction = tf.equal(tf.argmax(y_pred, 1), tf.cast(y_true, tf.int64)
    return tf.reduce_mean(tf.cast(correct_prediction, tf.float32), axis=-1)

optimizer = tf.optimizers.SGD(learning_rate)

def run_optimization(x, y):
    with tf.GradientTape() as g:
        pred = neural_net(x, is_training=True)
        loss = cross_entropy_loss(pred, y)
    trainable_variables = neural_net.trainable_variables
    gradients = g.gradient(loss, trainable_variables)
    optimizer.apply_gradients(zip(gradients, trainable_variables))

def train_model():
    for step, (batch_x, batch_y) in enumerate(train_data.take(training_steps), 1)
        run_optimization(batch_x, batch_y)
        if step % display_step == 0:
            pred = neural_net(batch_x, is_training=True)
            loss = cross_entropy_loss(pred, batch_y)
            acc = accuracy(pred, batch_y)
            print("step: %i, loss: %f, accuracy: %f" % (step, loss, acc))
    pred = neural_net(x_test, is_training=False)
    test_acc = accuracy(pred, y_test)
    print("Test Accuracy: %f" % test_acc)

def visualize_predictions():
    n_images = 7
    test_images = x_test[:n_images]
    predictions = neural_net(test_images, is_training=False)
    for i in range(n_images):
        plt.imshow(np.reshape(test_images[i], [28, 28]), cmap='gray')
        plt.title("Model prediction: %i" % np.argmax(predictions[i].numpy()))
        plt.show()

def upload_and_predict():
    uploaded = files.upload()
    for fn in uploaded.keys():
        img = Image.open(io.BytesIO(uploaded[fn])).convert('L')  # Convert to gra
        img = img.resize((28, 28))  # Resize to MNIST dimensions
        img_array = np.array(img) / 255.0  # Normalize
```

```
            img_array = img_array.reshape(1, 784)  # Reshape for model input

            prediction = neural_net(img_array, is_training=False)
            predicted_class = np.argmax(prediction[0].numpy())

            plt.imshow(img_array.reshape(28, 28), cmap='gray')
            plt.title("Model prediction: %i" % predicted_class)
            plt.show()
            print("The number predicted by the model is - %i" % predicted_class)

    while True:
        print("\nMenu:")
        print("1. Change learning rate")
        print("2. Change training steps")
        print("3. Change batch size")
        print("4. Change activation function")
        print("5. Start training")
        print("6. Visualize predictions")
        print("7. Upload and predict")
        print("8. Exit")

        choice = input("Enter your choice: ")

        if choice == '1':
            learning_rate = float(input("Enter new learning rate: "))
            optimizer.learning_rate = learning_rate
        elif choice == '2':
            training_steps = int(input("Enter new training steps: "))
        elif choice == '3':
            batch_size = int(input("Enter new batch size: "))
            train_data = update_train_data(batch_size)
        elif choice == '4':
            print("\nAvailable activation functions:")
            print("1. relu")
            print("2. sigmoid")
            print("3. tanh")
            print("4. linear")

            activation_choice = input("Enter your choice: ")
            if activation_choice == '1':
                activation = tf.nn.relu
            elif activation_choice == '2':
                activation = tf.nn.sigmoid
            elif activation_choice == '3':
                activation = tf.nn.tanh
            elif activation_choice == '4':
                activation = None
            else:
                print("Invalid choice. Using default activation (relu).")
                activation = tf.nn.relu

            neural_net = create_model(activation)
        elif choice == '5':
            train_model()
        elif choice == '6':
```

```
            visualize_predictions()
        elif choice == '7':
            upload_and_predict()
        elif choice == '8':
            break
        else:
            print("Invalid choice.")
```

```
...  4. Change activation function
    5. Start training
    6. Visualize predictions
    7. Upload and predict
    8. Exit
    Enter your choice: 3
    Enter new batch size: 250

    Menu:
    1. Change learning rate
    2. Change training steps
    3. Change batch size
    4. Change activation function
    5. Start training
    6. Visualize predictions
    7. Upload and predict
    8. Exit
    Enter your choice: 4

    Available activation functions:
    1. relu
```

```
2. Change training steps
3. Change batch size
4. Change activation function
5. Start training
6. Visualize predictions
7. Upload and predict
8. Exit
Enter your choice:
```