CNN

Rodrigo Cazarin 653060 1208

Explicacion del codigo

```
[ ]  import tensorflow as tf
     from tensorflow import keras
```

En este bloque de código se está importando la librería tensor flow la cual permite crear modelos y entrenarlos, la primera línea lo está importando como tf para facilitar su uso y en la segunda línea esta segun yo importando un módulo específico de la librería el cual es keras el cual es un API si no me equivoco.

```
[ ]  # Load the MNIST dataset
     (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_data()
```

En este bloque de código estamos poniendo como variable 2 argumentos, un argumento que entrena los modelos x & y , y el otro que testea los modelos x & y la variable está descargando del data con el módulo de keras que habíamos importado.

```
     # Print the shapes of the datasets
     print("x_train shape:", x_train.shape)  # (60000, 28, 28) - 60,000 images, 28x28 pixels
     print("y_train shape:", y_train.shape)  # (60000,) - 60,000 labels (digits 0-9)
     print("x_test shape:", x_test.shape)   # (10000, 28, 28) - 10,000 test images
     print("y_test shape:", y_test.shape)   # (10000,) - 10,000 test labels
```

En este bloque de código está haciendo las respectivas pruebas y testeos que previamente habíamos puesto como argumentos en una variable, en el x train shape se está entrenando con 60 mil imágenes de 20x20 pixeles y en él y trian shape solo se están entrenando los 60 mil labels. Ya cuando están testeando las shapes lo están haciendo con menor cantidad de imágenes y labels.

```
[ ]  # Normalize pixel values to be between 0 and 1
     x_train = x_train.astype("float32") / 255.0
     x_test = x_test.astype("float32") / 255.0
```

En este bloque de código solo se están modificando los valores de los pixeles para que solo estén entre 0 y 1, y los valores modificados están en una variable pues para poder usarlos más cómodamente.

```
[ ]  # Reshape the images to include a channel dimension (required for CNNs)
     img_width, img_height = 28, 28
     x_train = x_train.reshape(x_train.shape[0], img_width, img_height, 1)
     x_test = x_test.reshape(x_test.shape[0], img_width, img_height, 1)

     print("x_train shape after reshape:", x_train.shape) # (60000, 28, 28, 1)
     print("x_test shape after reshape:", x_test.shape)   # (10000, 28, 28, 1)
```

En este bloque en la primera línea está poniendo una variable con los argumentos que esta definiendo el ancho y altura de la imagen que están a 28x28 pixeles. En las otras dos variables se está haciendo un reshape con el primer dato de la variable train/test shape, el ancho y alto de la imagen con la variable que tenia los dos argumentos y un uno. Como

output esta poniendo las shapes ya con la forma cambiada.

```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Define the CNN model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_width, img_height, 1)),  # 32 filters, 3x3 kernel
    MaxPooling2D((2, 2)),  # Reduces spatial dimensions
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),  # Flatten the output for the dense layers
    Dense(128, activation='relu'),  # Fully connected layer with 128 units
    Dense(10, activation='softmax')  # Output layer with 10 units (one for each digit)
])

# Compile the model
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',  # Use sparse_categorical_crossentropy since labels are integers
              metrics=['accuracy'])

# Print the model summary
model.summary() # Useful for understanding the model's architecture

# Train the model
epochs = 5  # Adjust the number of epochs based on your available time/resources
batch_size = 64 # Adjust batch size as needed

history = model.fit(x_train, y_train,
                    epochs=epochs,
                    batch_size=batch_size,
                    validation_data=(x_test, y_test))
```

En este bloque de código se están importando dos módulos del módulo de keras el cual es de tensorflow, los módulos importados son Sequential, Comb2D,Maxpooling 2d , flatten y dense los cuales si no me equivoco sirven para definir los modelos. Luego se define el modelo CNN los cuales para hacer el modelo usamos todos los modulos que habiamos importado anteriormente por ejemplo con conv2d estamos poniendo filtros y haciendo un kernel con como se lo especifiquemos por ejemplo 32 filtros, 3x3 kernel, tambien con maxpoolingo estamos reduciendo los espacios dimensionales para que no tengamos problemas a la hora de hacer el modelo.

Después de crear el modelo este se compila y al compilar se pone como una variable algo para que no confunda los labels con otra cosa que no se integre. Al final solo está llamando un resumen del modelo e insertando unos datos para entrenarlo.

```python
# Evaluate the model
loss, accuracy = model.evaluate(x_test, y_test, verbose=0)
print(f"Test loss: {loss:.4f}")
print(f"Test accuracy: {accuracy:.4f}")

# Make predictions
import numpy as np
predictions = model.predict(x_test)  # Returns probabilities for each class
predicted_labels = np.argmax(predictions, axis=1)  # Get the class with the highest probability

# Print some predictions
print("Sample predictions:")
for i in range(10):
    print(f"Image {i}: Predicted label = {predicted_labels[i]}, Actual label = {y_test[i]}")
```

En este bloque de código lo primero que se está haciendo es queso está evaluando el modelo recién hecho, después con la librería de numpy se está haciendo probabilidades con los resultados de la evaluación del modelo, al final solo se están planteando las predicciones.

```python
from PIL import Image, ImageDraw

# Create a blank image (you'd replace this with code to capture student drawings)
image_size = 28
image = Image.new("L", (image_size, image_size), color=0)  # Black background
draw = ImageDraw.Draw(image)

# Simulate drawing a digit (e.g., a '3') - you'd get these coordinates from student input
# This is just example drawing!  You'd need a drawing application to get real coordinates.
draw.line([(5, 5), (23, 5)], fill=255, width=3)  # White lines to draw
draw.line([(23, 5), (23, 14)], fill=255, width=3)
draw.line([(5, 14), (23, 14)], fill=255, width=3)
draw.line([(5, 14), (5, 23)], fill=255, width=3)
draw.line([(5, 23), (23, 23)], fill=255, width=3)
image.save("drawn_digit.png")

# Load and preprocess the drawn image
drawn_image = Image.open("drawn_digit.png").convert("L") # Convert to grayscale.
drawn_image = drawn_image.resize((img_width, img_height))  # Resize to 28x28
drawn_image_array = np.array(drawn_image)  # Convert to NumPy array
drawn_image_array = drawn_image_array.astype("float32") / 255.0  # Normalize
drawn_image_array = drawn_image_array.reshape(1, img_width, img_height, 1)  # Reshape for the model

# Predict the digit
prediction = model.predict(drawn_image_array)
predicted_digit = np.argmax(prediction)

print(f"The model predicts the drawn digit is: {predicted_digit}")
```

En este bloque de código solo se van a hacer predicciones con una imagen dibujada insertada con las variables que le estamos dando al código integradas con los módulos importados de la librería PIL.

CNN en mi dia a dia

En mi dia a dia yo veo CNN por ejemplo en instagram o snapchat que tienen sus respectivos filtros los cuales ya habían sido previamente entrenados por sus diseñadores o creadores para que se adapten a la cara del sujeto, por ejemplo me imagino que los entrenamiento lo hicieron con personas random y los testeos lo hicieron tal vez instagram para que checara si los filtros no tenían fallos y pudieran ser subidos a su aplicación.

Regresión Lineal

La regresión lineal es una técnica estadística utilizada para modelar la relación entre una variable dependiente (también llamada variable objetivo o respuesta) y una o más variables independientes (también llamadas predictores o características). Su objetivo es encontrar una relación lineal que permita predecir el valor de la variable dependiente a partir de los valores de las variables independientes.

Referencias APA

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning: With applications in R. Springer.