PROYECTO FINAL: REDES NEURONALES CONVOLUCIONALES

24/05/2023

Nombre:

Ernesto Mendoza Martínez

Thakar Daniel Cepeda Ramos

Profesor:

López Arce Delgado, Jorge Ernesto

Materia:

SSP Algoritmia

Sección D11

Contenido

Introducción

Las redes neuronales son muy utilizadas hoy en día, puesto que son la base de las inteligencias artificiales que están en su apogeo en este momento.
Una red neuronal es un método de la inteligencia artificial que enseña a las computadoras a procesar datos como un cerebro humano.
Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente.
Las redes neuronales pueden ayudar a las computadoras a tomar decisiones inteligentes
Una de sus muchas utilidades, que es en la que nos enfocaremos mas adelante es la clasificación.

Objetivos

Objetivo General:

Comprender la utilización y las estructuras de las redes neuronales

Objetivo Particular

Elaborar una red neuronal para la clasificación de objetos o imágenes.

Clasificador de frutas y verduras

Una red neuronal es un método de la inteligencia artificial que enseña a las computadoras a procesar datos como un cerebro humano.
Crea un sistema adaptable que las computadoras utilizan para aprender de sus errores y mejorar continuamente.
Las redes neuronales pueden ayudar a las computadoras a tomar decisiones inteligentes
Una de sus muchas utilidades, que es en la que nos enfocaremos mas adelante es la clasificación.
Ahora, existe varias arquitecturas de redes neuronales, pero nos enfocaremos solamente en una, una red convolucional
Esta especialmente diseñada para procesar datos con estructura de cuadrícula, como imágenes y videos.
Las CNN son muy efectivas en tareas como clasificación de imágenes, detección de objetos, segmentación semántica, entre otras.
Su diseño se basa en el concepto de convolución, que es una operación matemática que permite extraer características locales de la imagen utilizando filtros o kernels.
A través de la convolución, las CNN pueden detectar patrones visuales simples en regiones locales y combinarlos en capas posteriores para detectar características más complejas.
Capa de entrada: Las imágenes se introducen en la red neuronal como tensores de píxeles, donde cada valor de píxel representa una característica.
Capas convolucionales: Estas capas aplican filtros a pequeñas regiones de la imagen de entrada para extraer características locales.
Capas de agrupación (pooling): Esto ayuda a disminuir la cantidad de parámetros y a obtener una representación más compacta de las características.
Capa de salida: La última capa de la CNN produce las salidas finales, que dependen de la tarea que se esté abordando.
Utilizamos lo que es la biblioteca tensorflow de donde exportaremos lo que es keras
Son utilizadas en el área de aprendizaje automático como lo puede ser entrenar redes neuronales.

`import os`
`import tensorflow as tf`
`from tensorflow import keras`
	`data_entrenamiento = './dataset/train'`
	`data_validacion = './dataset/validation'`
	Cargamos las carpetas con las imágenes.

Utilizamos las librerías de tensorflow y keras

```
# Parámetros
epocas = 40 #Número de iteraciones sobre el set de datos
altura, longitud = 100, 100 #Ajuste de tamaño de las imagenes
batch_size = 32  #numero de imagenes a procesar en cada paso
pasos = 50 #numero de veces que se procesa la informacion de cada epoca
pasos_validacion = 15 #al final de cada epoca tmabien se verificara el set de val
filtrosC1 = 32 #Dspues de cada convolucion tendra una profundidad de 32 pixeles
filtrosC2 = 64 #
tamano_filtro1 = (3, 3) #ira en un tamaño de 3*3
tamano_filtro2 = (2, 2) #
tamano_pool = (2, 2)
clases = 36 #numero de clases que tenemos en nuestro dataset
lr = 0.001 #Taza de aprendizaje de la red neuronal
```

Los parámetros que se pasan a la cnn

```
# Preprocesamiento de imágenes, escalado, generar imagenes inclinadas, zoom, voltea las imagenes
entrenamiento_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True
)

validacion_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255
)

# Carga de imágenes
imagen_entrenamiento = entrenamiento_datagen.flow_from_directory(
    data_entrenamiento,
    target_size=(altura, longitud),
    batch_size=batch_size,
    class_mode='categorical'
)
```

Preprocesamiento y carga de imágenes a la cnn con sus parámetros y filtros modificados

```
# Creación del modelo CNN
cnn = keras.Sequential()

cnn.add(keras.layers.Conv2D(filtrosC1, tamano_filtro1, padding='same', input_shape=(altura, longitud, 3), activation
cnn.add(keras.layers.MaxPooling2D(pool_size=tamano_pool))

cnn.add(keras.layers.Conv2D(filtrosC2, tamano_filtro2, padding='same', activation='relu'))
cnn.add(keras.layers.MaxPooling2D(pool_size=tamano_pool))

cnn.add(keras.layers.Flatten())
cnn.add(keras.layers.Dense(256, activation='relu'))
cnn.add(keras.layers.Dropout(0.5))
cnn.add(keras.layers.Dense(clases, activation='softmax'))
```

Creación de las redes neuronales

```
# Compilación del modelo
cnn.compile(loss='categorical_crossentropy', optimizer=keras.optimizers.Adam(lr=lr), metrics=['accuracy'])

# Entrenamiento del modelo
cnn.fit(imagen_entrenamiento, steps_per_epoch=pasos, epochs=epocas, validation_data=imagen_validacion, validation_steps=pa

# Guardar el modelo y los pesos
```

Compilacion y entrenamiento del modelo.

```
# Guardar el modelo y los pesos
dir = './modelo/'

if not os.path.exists(dir):
    os.mkdir(dir)

cnn.save('./modelo/modelo.h5')
cnn.save_weights('./modelo/pesos.h5')
```

Guardar un modelo preentrenado

```
import numpy as np
import tensorflow as tf
from tensorflow import keras
```

Realizamos un programa de predicción para probar el funcionamiento de la cnn utilizando la librería keras

```
longitud, altura = 100, 100
modelo = './modelo/modelo.h5'
pesos = './modelo/pesos.h5'
```

Abre el modelo preentrenado

```
# Función que recibirá la imagen y dirá lo que es
def predict(file):
    x = keras.preprocessing.image.load_img(file, target_size=(longitud, al
    x = keras.preprocessing.image.img_to_array(x)
    x = np.expand_dims(x, axis=0)
    arreglo = cnn.predict(x)
    resultado = arreglo[0]
    respuesta = np.argmax(resultado)
```

Realiza la lectura de la imagen para leer las características al imagen que le pasamos

```
# Mostrar respuesta
if respuesta == 0:
    print('0')
elif respuesta == 1:
    print('1')
elif respuesta == 2:
    print('2')
elif respuesta == 3:
    print('3')
elif respuesta == 4:
    print('4')
elif respuesta == 5:
```

Posibles resultados

```
# Ejemplo de uso
imagen = 'Image_3.jpg'
predict(imagen)
```

```
1/1 [==============================] - 3s 3s/step
0
```

Resultado de la imagen que pasamos (0=manzana)

Conclusión:

Las redes neuronales son una herramienta muy útil hoy en día, nuestra cnn es utilizada en este momento para clasificar frutas y verduras, puesto que la cantidad de neuronas o copas que implementamos no son las suficientes para lograr un modelo mejor y mas complejo. El afinamiento realmente fue nuestro mayor desafio puesto que cambiamos muchos parametros continuamente.