Universidad Autónoma de Aguascalientes

Centro de Ciencias Básicas

Departamento de Ciencias de la Computación

Optativa Profesionalizante II: Machine Learning y Deep Learning

10° "A"

Actividad 6: CNN

Docente: Dr. Francisco Javier Luna Rosas

Alumno: Joel Alejandro Espinoza Sánchez (211800)

Fecha de Entrega: Aguascalientes, Ags., 1 de mayo del 2023.

El alumno deberá elaborar un documento (`*.pdf`) y un archivo auto-reproducible (`*.html`) que analice, implemente y evalúe una red neuronal convolucional. El documento deberá contener:

- Portada
- Evidencias de la actividad
- Conclusiones
- Referencias (formato APA)

Se cargan las librerías

In [1]:
```
import os
import numpy as np
from PIL import Image
import tensorflow as tf
from tensorflow.keras import layers, models
```

Los números están separados en carpetas. Se cargan estos números

In [2]:
```
data_dir = 'numbers'
labels = os.listdir(data_dir)
```

Se cargan las imágenes

In [3]:
```
images = []
for label in labels:
    for filename in os.listdir(os.path.join(data_dir, label)):
        img = Image.open(os.path.join(data_dir, label, filename))
        img = img.resize((28, 28), resample=Image.BILINEAR)
        img = np.array(img)
        images.append((img, int(label)))
```

```
C:\Users\alexe\Anaconda3\envs\ici-thesis\lib\site-packages\ipykernel_launcher.py:5: D
eprecationWarning: BILINEAR is deprecated and will be removed in Pillow 10 (2023-07-0
1). Use Resampling.BILINEAR instead.
  """
```

Se mezclan y se separan las imágenes

In [4]:
```
np.random.shuffle(images)

split = int(0.8 * len(images))
train_data = images[:split]
test_data = images[split:]
```

Se crea el conjunto de entrenamiento y el de evaluación

In [5]:
```
x_train = np.array([train_data[i][0] for i in range(len(train_data))])
y_train = np.array([train_data[i][1] for i in range(len(train_data))])
x_test = np.array([test_data[i][0] for i in range(len(test_data))])
y_test = np.array([test_data[i][1] for i in range(len(test_data))])

x_train = x_train.reshape((-1, 28, 28, 3))
x_train = x_train.astype('float32') / 255.0
x_test = x_test.reshape((-1, 28, 28, 3))
x_test = x_test.astype('float32') / 255.0
y_train = y_train.reshape((-1, 1))
y_train = y_train.astype('float32') / 255.0
```

Se crea el modelo de la CNN

In [6]:
```
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(28, 28, 3)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

Se evalúa el modelo

In [7]:
```
model.fit(x_train, y_train, epochs=5, batch_size=32, validation_data=(x_test, y_test))
```

```
test_loss, test_acc = model.evaluate(x_test, y_test, verbose=2)
print('Accuracy:', test_acc)
```

```
Epoch 1/5
539/539 [==============================] - 7s 12ms/step - loss: 0.0075 - accuracy: 0.
1042 - val_loss: 36.4981 - val_accuracy: 0.1002
Epoch 2/5
539/539 [==============================] - 6s 10ms/step - loss: 0.0000e+00 - accurac
y: 0.1046 - val_loss: 36.4981 - val_accuracy: 0.1002
Epoch 3/5
539/539 [==============================] - 6s 11ms/step - loss: 0.0000e+00 - accurac
y: 0.1046 - val_loss: 36.4981 - val_accuracy: 0.1002
Epoch 4/5
539/539 [==============================] - 6s 12ms/step - loss: 0.0000e+00 - accurac
y: 0.1046 - val_loss: 36.4981 - val_accuracy: 0.1002
Epoch 5/5
539/539 [==============================] - 6s 11ms/step - loss: 0.0000e+00 - accurac
y: 0.1046 - val_loss: 36.4980 - val_accuracy: 0.1002
135/135 - 0s - loss: 36.4980 - accuracy: 0.1002 - 449ms/epoch - 3ms/step
Accuracy: 0.10020876675844193
```

Podemos observar que la precisión no es tan buena por lo que el modelo no es muy fiable para la clasificación de las imágenes seleccionadas.

Conclusiones

Es interesante e importante poder implementar una red neuronal convolucional pues, posteriormente no basta con sólo importar librerías que realicen el trabajo pesado, ya que, implementar manualmente estos algoritmos nos enseña a qué hay detrás del algoritmo, cómo funciona y poder comprender realmente qué está ocurriendo como la base de una red neuronal convolucional y la forma en la que ésta aprende. Es muy útil la implementación de estos algoritmos en estas tareas para las futuras tareas de la materia y aplicaciones de Machine Learning en la vida personal.

Referencias

- Anónimo (s.f.) “Red neuronal artificial”. Obtenido de Wikipedia: https://es.wikipedia.org/wiki/Red_neuronal_artificial.
- Data Scientest (2021) “Perceptrón. ¿Qué es y para qué sirve?”. Obtenido de Data Scientest: https://datascientest.com/es/perceptron-que-es-y-para-que-sirve.
- Luna, F. (2023) “El Modelo de McCulloch – Pitts”. Apuntes de ICI 10°.