Práctica 5

Redes Neuronales Convolucionales

Objetivos

El objetivo de esta práctica es comprender el funcionamiento de las redes neuronales convolucionales

Temas

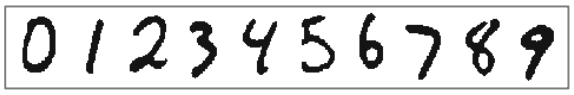
- Redes Neuronales Convolucionales. Capas Conv2D, MaxPooling, Flatten, BatchNormalization, Dense. Parada Temprana.
- Data Augmentation.

Lectura

Material de Lectura: Capítulo 6 del libro Neural Networks and Deep Learning.

Ejercicio 1

La base de datos MNIST contiene imágenes de 28×28, en escala de grises, de números escritos a mano. Está conformada por 60.000 ejemplos de entrenamiento y 10.000 ejemplos de prueba.



Para cargar las imágenes utilice:

```
from tensorflow.keras.datasets import mnist
(X_train, Y_train), (X_test, Y_test) = mnist.load_data()
```

Puede visualizar una imagen utilizando:

```
nImg = 0    # nro. de imagen a visualizar
plt.imshow(X_train[0, :,:], cmap='gray')
```

a) Con el conjunto de 60000 imágenes entrene una red neuronal convolucional para predecir el dígito presente en la imagen. Recuerde normalizar los valores de cada imagen. Salve el modelo para recuperarlo después.
b) Levante el modelo guardado en el punto a) y utilice la clase ***DrawPanel*** del módulo ***utils.images*** de la carpeta fuentes para generar un dibujo escrito a mano de un dígito y predecir la clase a la que pertenece.

Ejercicio 2

Se buscará resolver la clasificación de los dígitos de MNIST usando la siguiente configuración:

```
model = Sequential()
model.add(Input(shape=(28, 28, 1)))
model.add(Conv2D(F, kernel_size=K, strides=(S,S), activation=FUN))
model.add(MaxPooling2D(pool_size=(2,2)))    # -- opcional --
model.add(Flatten())
model.add(Dense(10,activation='softmax'))

model.summary()
```

donde F es la cantidad de filtros o de mapas de características, K es el tamaño del kernel o máscara, S es el valor del stride y FUN es la función de activación de la capa de convolución.

La tabla que aparece a continuación sugiere los valores a utilizar. Se recomienda emplear Parada Temprana para reducir el tiempo de entrenamiento. Para ello utilice

```
from tensorflow.keras.callbacks import EarlyStopping
es = EarlyStopping(monitor='val_accuracy', patience=5, min_delta=0.001)
```

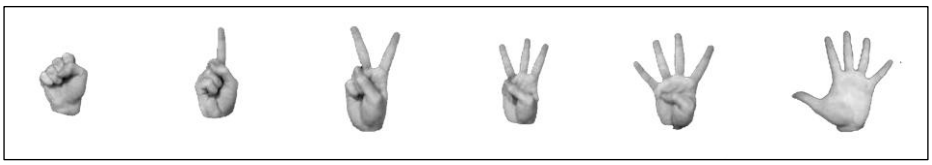
Esto indica que, si el valor del accuracy sobre los datos de validación no mejora después de 5 épocas, el entrenamiento finaliza. Puede usarse el parámetro min_delta para indicar cuando la diferencia entre dos accuracy se considerará significativa. Luego agregue este objeto en el momento del entrenamiento por medio del párametro callbacks

```
H = model.fit(x = X_train, y = Y_train, batch_size = LOTES,
        validation_data = (X_test, Y_test), epochs=4000, callbacks=[es])
```

Capa Convolucional Conv2D				Max Pooling con filtro de tamaño 2x2 y stride=2	Total de parámetros	Épocas	Accuracy en Train	Accuracy en Test
Cant. de filtros	Tamaño del kernel o filtro	Stride	Función de activación					
4	3x3	1	ReLU	Si				
16	3x3	1	ReLU	Si				
64	3x3	1	ReLU	Si				
4	7x7	1	ReLU	Si				
16	7x7	1	ReLU	Si				
64	7x7	1	ReLU	Si				
64	3x3	2	ReLU	No				
64	3x3	3	ReLU	No				
64	3x3	1	TanH	Si				
64	3x3	1	Sigmoide	Si				

Ejercicio 3

Para resolver este ejercicio utilice un modelo de red neuronal convolucional que reconozca la cantidad de dedos extendidos en cada mano de las imágenes que conforman el juego de datos ***"Fingers"***.



La versión original de este de estas imágenes se encuentra en https://www.kaggle.com/koryakinp/fingers.

Puede hallar una versión reducida de estas imágenes en el Moodle del curso, en la misma sección donde se encuentra este enunciado de práctica. También encontrará allí ejemplos sobre cómo cargar estas imágenes y cómo procesarlas con una red neuronal convolucional.

a) Entrene y pruebe un modelo utilizando los datos de las carpetas ***test*** y ***train***, midiendo ***accuracy***
b) Genere una versión del ***dataset*** para ***test*** agregando transformaciones al azar sobres imágenes originales. Haga ***rotaciones*** entre -45 y 45 grados, repita el test y mida el accuracy.
c) Genere una versión del ***dataset train*** como la realizada en b) y repita entrenamiento y prueba de a) con los datasets de modificados.

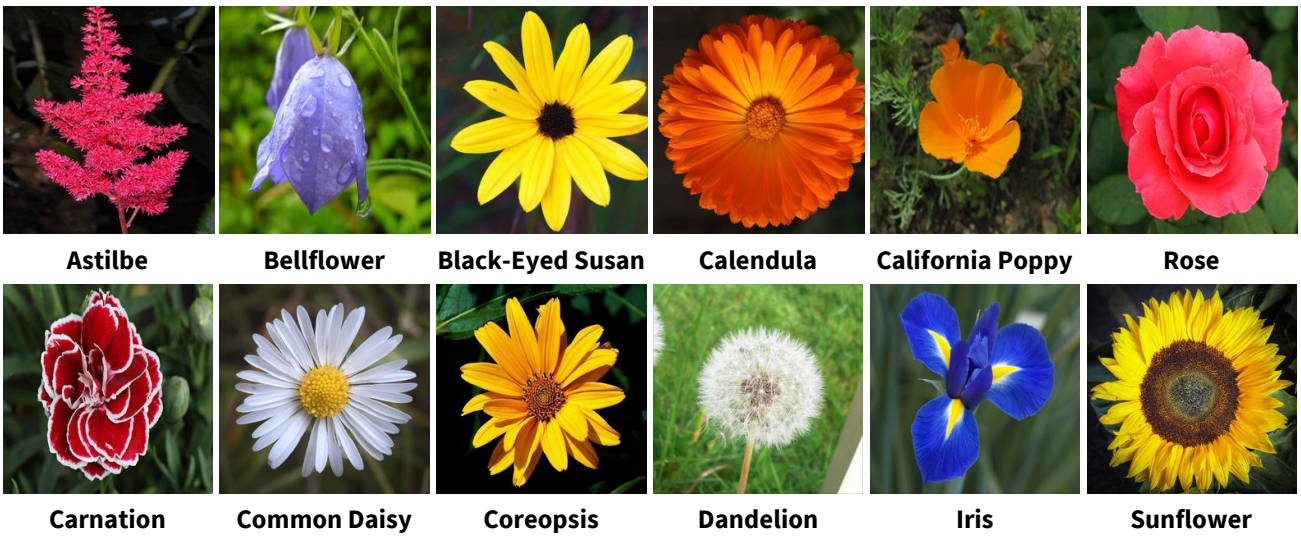
Ejercicio 4

Resuelva el punto c) del ejercicio anterior con las imágenes originales de las carpetas ***train*** y ***test*** y utilice:

a) el objeto ***ImageDataGenerator*** del módulo ***tensorflow.keras.preprocessing.image*** para generar de forma automática una versión aumentada de los datos con las características del punto b). Utilice el método ***flow_from_file*** del objeto ***ImageDataGenerator*** para utilizar directamente las imágenes de las carpetas en vez de cargarlas en memoria.
b) repita el punto a) utilizando la función ***tf.keras.utils.image_dataset_from_directory*** para generar el dataset a partir de carpetas en combinación con funciones para aplicar el preprocesamiento/data augmentation.

Ejercicio 5

"Flowers" es un juego de datos compuesto por 16 categorías de flores con unas 1000 imágenes de RGB de 256 x 256 píxeles disponible en Kaggle en el enlace https://www.kaggle.com/datasets/l3llff/flowers.
En la carpeta de datos del Moodle del curso hay una versión reducida de 96 x 96 píxeles. Utilice este dataset para entrenar varios modelos de redes convolucionales que permitan clasificar las 16 flores según las especificaciones que siguen a continuación.



Para cada modelo a entrenar contabilice la cantidad de ***épocas***, ***accuracy*** y ***tiempo*** promedio por época (utilice la clase ***EpochTiming*** definida en el paquete ***tf_utils*** de la carpeta ***Fuentes*** como ***callback*** para el entrenamiento). Utilice 250 épocas para los entrenamientos y agregue una parada temprana para evitar overfitting.

a) Entrene un modelo que utilice solamente capas ***Conv2D***, ***MaxPooling2D***, ***Flatten*** y ***Dense***.
b) Modifique la arquitectura anterior agregando capas de ***BatchNormalization*** luego las capas de ***MaxPooling2D*** y entrene un nuevo modelo.
c) Compare las mediciones de los entrenamientos de los modelos de a) y b ¿Qué puede concluir? ¿Por qué?

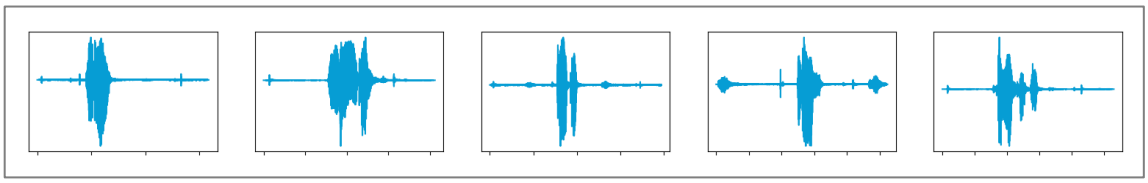
Ejercicio 6

Re-implemente el ejercicio 5 utilizando las capas convolucionales del modelo Mobilenet V3 (clase ***MobileNetV3Small*** del paquete ***tensorflow.keras.applications***) para extraer los patrones característicos y agregue una o más capas densas para entrenar una nueva red que permita reconocer los 16 tipos de flores del dataset ***"Flowers"***. Recuerde que solo debe entrenar las capas densas.

Ejercicio 7

Mozilla Common Voice es un proyecto (https://commonvoice.mozilla.org/es/datasets) que desarrolla una base de datos de voces, abierta y multi idioma, que cualquiera puede usar para entrenar aplicaciones que utilicen la voz como interfaz.

En particular cuenta con un pequeño corpus denominado **Single Word** que contiene las palabras habladas SI, NO, Hey, Firefox y dígitos del cero al nueve.



a) Utilizando los ejemplos de una versión seleccionada de estos audios en el Moodle del curso, transformarlos en imágenes para que puedan ser procesados por la red siguiendo los siguientes criterios:
   - Eliminar los silencios iniciales y finales. La comparación de los silencios en 2 audios produce una falsa coincidencia.
   - Elegir un tamaño fijo para los audios. Todas las imágenes deben tener el mismo tamaño. Como las palabras del corpus son breves un tamaño de 0.75 segundos es razonable.
   - Achicar/agrandar los audios manteniendo el tono de la voz
   - Pasar el audio del dominio del tiempo al domino de la frecuencia. Dividir el audio en pequeños intervalos y obtener un conjunto de valores que determinen las frecuencias presentes. Para esto puede utilizarse la transformada de Fourier (FFT) o los coeficientes cepstrales en frecuencia mel (MFCC) que aproximan la sensibilidad del oído humano.
   - Finalmente convertir los coeficientes en una imagen con compresión sin pérdida como PNG.

b) Utilizando las imágenes generadas en a), entrenar un modelo de red neuronal convolucional que permita reconocer las palabras del corpus Single Word.