Universidad de San Carlos de Guatemala

Facultad de Ingeniería

Escuela de Ciencias y Sistemas

Inteligencia Artificial I

Proyecto 1

GRUPO #20

MascotaMatch

Carnet	Nombre
201602942	Mario Yonathan Tun Quino
201800987	Luis Diego de Leon Sanchez

MANUAL TECNICO

Se debe construir una aplicación para dispositivos Android que pueda detectar una imagen a través de la cámara y la reconozca, clasificando en una especie de animal. Luego debe colocar encima de la fotografía una figura representativa de la clase detectada.

Herramienta utilizada:

Es una plataforma gratuita basada en la nube proporcionada por Google que permite a los usuarios escribir y ejecutar código de Python en un entorno de bloc de notas colaborativo.

Google Colab es especialmente útil para tareas de ciencia de datos, aprendizaje automático, educación y colaboración en proyectos de programación. Su accesibilidad y potentes características hacen que sea una opción popular para aquellos que desean ejecutar código de Python en la nube de manera colaborativa y gratuita.

Entre algunas de sus caracteristicas se destaca:

- Entorno de Bloc de Notas Colaborativo
- Basado en la Nube
- Uso Gratuito de Recursos de GPU
- Integración con Google Drive

Funcionamiento:

Primero se debe especificar un rango de imagenes sobre las cuales se entrenara el modelo para poder lograr que se reconozca lo que se desea, en este caso se cargo un dataset enfocado en perros y gatos los cuales contienen al rededor de 23k imagenes

```
import tensorflow as tf
import tensorflow_datasets as tfds
datos, metadatos = tfds.load('cats_vs_dogs', as_supervised=True, with_info=True)
```

Luego de eso, se procede a guardar en un arreglo, todos los datos de pixeles referentes a las imagenes junto con sus etiquetas correspondientes

```
import cv2

datos_entrenamiento = []
imagen_tm = 100

for i, (imagen, etiqueta) in enumerate(datos['train']):
  imagen = cv2.resize(imagen.numpy(), (imagen_tm, imagen_tm))
  imagen = cv2.cvtColor(imagen, cv2.COLOR_BGR2GRAY)
  imagen = imagen.reshape(imagen_tm, imagen_tm, 1)

  datos_entrenamiento.append([imagen, etiqueta])
```

Ahora se necesita separar lo que son los valores de las imagenes y las etiquetas para despues poder organizarlas debidamente:

```
X = []
Y = []

for imagen, etiqueta in datos_entrenamiento:
  X.append(imagen)
  Y.append(etiqueta)
```

Luego la parte mas importante, para poder lograr una precision en la lectura de datos adecuada, se necesita que se ajusten las imagenes para que asi todas sigan un estandar, y el entrenamiento pueda ser mas efectivo y preciso

```
[ ]  import numpy as np

     X = np.array(X).astype(float) / 255
     Y = np.array(Y)
```

```
     from tensorflow.keras.preprocessing.image import ImageDataGenerator

     datagen = ImageDataGenerator(
         rotation_range = 30,
         width_shift_range = 0.2,
         height_shift_range = 0.2,
         shear_range=15,
         zoom_range=[0.7, 1.4]
     )

     datagen.fit(X)

     plt.figure(figsize=(20,8))

     for imagen, etiqueta in datagen.flow(X, Y, batch_size=10, shuffle=False):
       for i in range(10):
         plt.subplot(2, 5, i+1)
         plt.imshow(imagen[i].reshape(100, 100), cmap="gray")
       break
```

Adjunto a ello, se debe crear un modelo secuencial el cual a base de las neuronas y las funciones de activacion que se le especifique, entrenata en base a las imagenes subidas, luego se compila y por ultimo se separan la cantidad de imagenes para entrenamiento y tambien para validaciones

```
[ ]  model = tf.keras.models.Sequential([
         tf.keras.layers.Conv2D(32, (3, 3) , activation='relu', input_shape=(100,100,1)),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
         tf.keras.layers.MaxPooling2D(2, 2),
         tf.keras.layers.Conv2D(128, (3, 3), activation='relu'),
         tf.keras.layers.MaxPooling2D(2, 2),

         tf.keras.layers.Dropout(0.5),
         tf.keras.layers.Flatten(),
         tf.keras.layers.Dense(250, activation='relu'),
         tf.keras.layers.Dense(1, activation='sigmoid'),
     ])
```

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

```
[ ]  X_entrenamiento = X[:19700]
     X_validacion = X[19700:]
```

```
[ ]  Y_entrenamiento = Y[:19700]
     Y_validacion = Y[19700:]
```

```
[ ]  data_gen_entrenamiento = datagen.flow(X_entrenamiento, Y_entrenamiento, batch_size=32)
```

Se entrena el modelo con la cantidad de epochs deseadas para entrenar de mejor manera el modelo:

```
model.fit(data_gen_entrenamiento, batch_size=32, validation_data=(X_validacion, Y_validacion),
          steps_per_epoch=int(np.ceil(len(X_entrenamiento)/ float(32))), validation_steps=int(np.ceil(len(X_validacion)/ float(32))),
          epochs=100)
```

```
Epoch 1/100
616/616 [==============================] - 29s 46ms/step - loss: 0.2629 - accuracy: 0.8879 - val_loss: 0.2208 - val_accuracy: 0.9043
Epoch 2/100
616/616 [==============================] - 27s 43ms/step - loss: 0.2550 - accuracy: 0.8896 - val_loss: 0.2258 - val_accuracy: 0.9015
Epoch 3/100
616/616 [==============================] - 26s 42ms/step - loss: 0.2587 - accuracy: 0.8889 - val_loss: 0.1862 - val_accuracy: 0.9256
Epoch 4/100
616/616 [==============================] - 26s 42ms/step - loss: 0.2593 - accuracy: 0.8891 - val_loss: 0.1924 - val_accuracy: 0.9186
Epoch 5/100
616/616 [==============================] - 27s 43ms/step - loss: 0.2535 - accuracy: 0.8924 - val_loss: 0.2777 - val_accuracy: 0.8739
```

Y por ultimo cuando ya se llegue a un estandar de precision adecuado y coherente, se extrae el modelo para poder compilarlo en una app propia para dispositivos

```
[ ] model.save('modelo-perros-gatos.h5')
```

```
/usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via
  saving_api.save_model(
```

```
[ ] !pip install tensorflowjs
```

```
[ ] !mkdir exportacion_modelo
```

```
[ ] !tensorflowjs_converter --input_format keras modelo-perros-gatos.h5 exportacion_modelo
```

```
2023-12-20 02:16:46.791394: E external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register cuDNN factory: Attempti
2023-12-20 02:16:46.791445: E external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register cuFFT factory: Attemptir
2023-12-20 02:16:46.792695: E external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to register cuBLAS factory: Attemp
2023-12-20 02:16:47.796533: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not find TensorRT
```

```
[ ]
```