Francisco Javier Castro Zuñiga
A01281649

Image classification

Dataset

Las imágenes fueron obtenidas de Kaggle del siguiente Link:
https://www.kaggle.com/kmader/food41

Proporcionan cerca de 100 platillos, pero yo escogí los siguientes: hamburguesa, pizza, filete, hot dog, y papas a la francesa.

Preparar carpetas

Se crean 3 carpetas: train, valid, test.
Cada una de ellas contendra otra carpeta correspondiente a cada clase.

```
food = ['steak', 'hamburger', 'french_fries', 'hot_dog', 'pizza']
folders = ['train', 'valid', 'test']

for j in folders:
    for i in food:
        mydir = 'food/' + j + '/' + i
        print(mydir)
        try:
            os.makedirs(mydir)
            break
        except:
            pass
```

Poblar carpetas

```
#train
for i in food:
    for _ in range(600):
        dest = 'food/train/' + i
        src = 'food/' + i
        img = random.choice(os.listdir(src))
        shutil.move(src + '/' + img, dest)

#test
for i in food:
    for _ in range(100):
        dest = 'food/test/' + i
        src = 'food/' + i
        img = random.choice(os.listdir(src))
        shutil.move(src + '/' + img, dest)

#validation
for i in food:
    for _ in range(100):
        dest = 'food/valid/' + i
        src = 'food/' + i
        img = random.choice(os.listdir(src))
        shutil.move(src + '/' + img, dest)
```

Se comienzan a llenar las carpetas de cada clase dentro de cada sección (train, valid, test) tomando muestras random.

Entrenamiento

Se utiliza un modelo de red neuronal vgg16
`vgg16_model = tf.keras.applications.vgg16.VGG16()`

Se agrega cada capa del modelo al nuestro, además de una extra de 5 unidades ya que son 5 clases a clasificar, con la funcion softmax como activación. Y se crearon copias del modelo para experimentar variando las épocas y las muestras a analizar por época.

```
model = Sequential()
for layer in vgg16_model.layers[:-1]:
    model.add(layer)

model.add(Dense(units=5, activation='softmax'))
```

```
modelDos = model
modelTres = model
```

```
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
model.fit(x=train_batches,
          steps_per_epoch=10,
          validation_data=valid_batches,
          validation_steps=5,
          epochs=30,
          verbose=1
)
```

Se compila y comientza el entrenamiento con las especificaciones mostradas en la imagen. Con el accuracy como métrica de desempeño conseguimos un 70%

```
modelDos.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])
```

```
modelDos.fit(x=train_batches,
          steps_per_epoch=5,
          validation_data=valid_batches,
          validation_steps=5,
          epochs=100,
          verbose=1
)
```

Para el siguiente modelo se agregaron epocas y se disminuyeron los pasos por épocas. Para mejorar el rendimiento a un 80% de accuracy.

Clasificación

```
def plotImages(images_arr):
    fig, axes = plt.subplots(1, 10, figsize=(20,20))
    axes = axes.flatten()
    for img, ax in zip( images_arr, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()
```

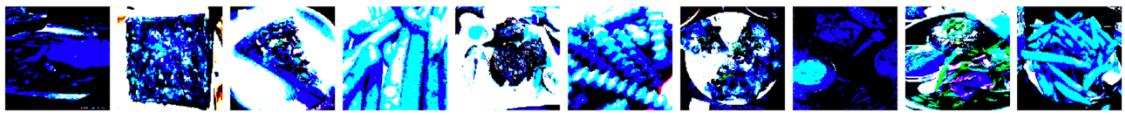
```
def printLabel(labels):
    for l in labels:
        i = np.where(l == 1)
        print(food[i[0][0]])
```

Se crean estas funciones que ayudan a mostrar la imagen y el label correspondiente.

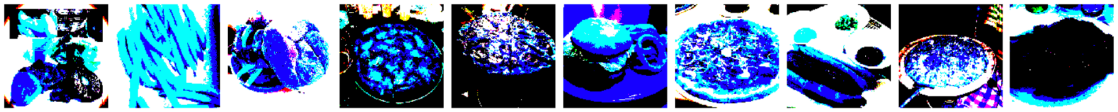
Nota: cuando se seleccionaron las muestras, se activo la opción: shuffle para analizar imágenes mezcladas y al realizar las predicciones no mostrara el mismo tipo.

```
imgs, labels = next(test_batches)
plotImages(imgs)
printLabel(labels)
```

Y los resultados son estos:



```
hamburger
pizza
pizza
french_fries
steak
french_fries
pizza
hamburger
hamburger
french_fries
```



```
steak
french_fries
hamburger
pizza
pizza
hamburger
pizza
hot_dog
pizza
steak
```