Memoria de prácticas de Fundamentos de los Sistemas Inteligentes

Alejandro Medina Díaz & Aitor Ventura Delgado

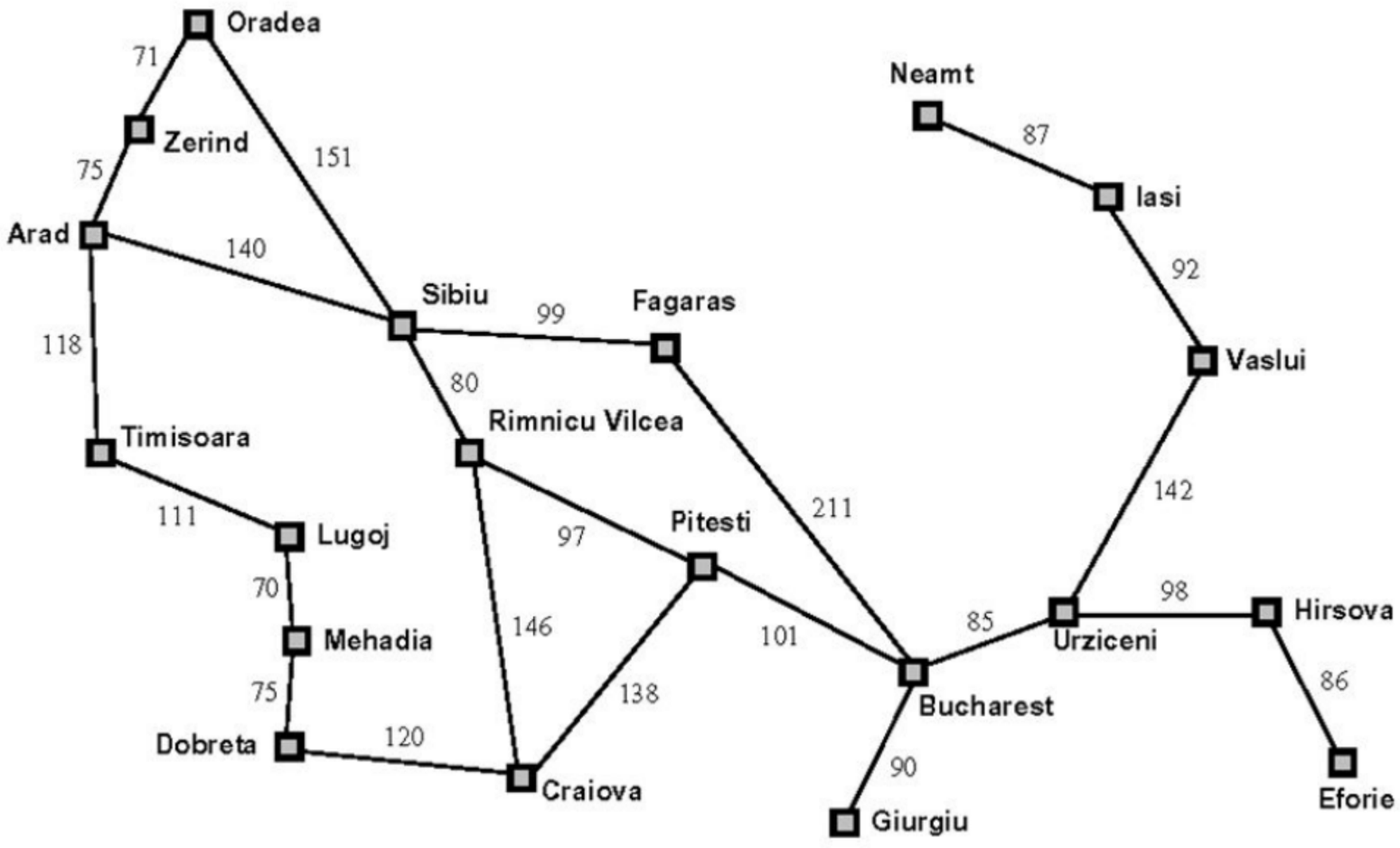
Curso 2020/2021 - Escuela de Ingeniería Informática

¿Quiénes somos?

Somos el **equipo 06** de prácticas de la asignatura **40825 - Fundamentos de los Sistemas Inteligentes** del curso 2020/2021 de la Escuela de Ingeniería Informática, en la Universidad de Las Palmas de Gran Canaria.

Las prácticas de este año han sido dos: la primera, que se centra en encontrar y desarrollar diferentes estrategias de búsqueda en grafos; y la segunda, que pide desarrollar una red neuronal para la clasificación de un conjunto de imágenes.

Primera práctica - Búsqueda en grafos.



Código base

[ ] ↳ *3 celdas ocultas*

Primera parte

La estrategia de búsqueda de ramificación y acotación pertenece a las estrategias de búsqueda no informada. Su principio de funcionamiento se basa en ordenar la lista abierta tomando como criterio el coste acumulado de cada trayectoria parcial, representada ésta por cada nodo de la lista abierta. Por tanto, el primer nodo de la lista irá expandiéndose y generando nuevas trayectorias por ramificación de sus hijos.

Tareas:

- *A partir del código base entregado, se deberá programar el método de ramificación y acotación. Utilícese como problema el grafo de las ciudades de Rumanía presente en el código.*
- *Comparar la cantidad de nodos expandidos por este método con relación a los métodos de búsqueda primero en anchura y primero en profundidad.*
- *Realizar a mano la traza de una búsqueda (opcional).*

Realización de los alumnos

Decidimos crear un nuevo tipo de cola, que se llamaría *OrderedQueue*. Esta cola devuelve los elementos en orden creciente dependiendo de los *path_cost* de los nodos. Luego, para la búsqueda de ramificación y acotación, llamamos al método *graph_search* con esta cola.

```
class OrderedQueue(Queue):
    """A queue that pops its elements in order of least cost."""

    def __init__(self):
        self.A = []
        self.dirty = False

    def append(self, item):
        self.A.append(item)
        self.dirty = True

    def __len__(self):
        return len(self.A)

    def extend(self, items):
        self.A.extend(items)
        self.dirty = True

    def pop(self):
        if self.dirty:
            self.A = sorted(self.A, key=lambda node: node.path_cost, reverse=True)

        self.dirty = False

        return self.A.pop()
```

```
def branch_and_bound_search (problem):
    """Search using least cost node in the fringe """
    return graph_search(problem, OrderedQueue())
```

```
print(branch_and_bound_search(ab).path())
```

```
[<Node B>, <Node P>, <Node R>, <Node S>, <Node A>]
```

Segunda parte

La estrategia de búsqueda ramificación y acotación con subestimación pertenece a las estrategias de búsqueda informada. En este caso, además de utilizar el coste acumulado de un camino desde el estado inicial hasta un cierto estado del grafo, se utiliza una estimación heurística hasta el estado final para ordenar la lista abierta. De esta forma, dado un determinado nodo n del árbol de búsqueda, la expresión de coste estimado f(n) será:

```
f(n) = g(n) + h(n)
```

Donde g(n) representa el coste acumulado y h(n) la heurística utilizada. Para que el camino encontrado sea óptimo, la heurística debe ser consistente. Es decir, ha de cumplir que para cada nodo n y cada nodo hijo n' alcanzado mediante la acción a, el valor heurístico h(n) debe ser siempre menor o igual al valor heurístico h(n') más el coste del nodo n al n' mediante la acción a.

```
h(n) <= c(n, a, n') + h(n')
```

Tareas:

- *A partir del código base entregado, se deberá programar el método de búsqueda de ramificación y acotación con subestimación. Utilícese como problema el grafo de las ciudades de Rumanía presente en el código. Como heurística se utilizará la distancia en línea recta entre cada estado y el estado final.*
- *Comparar la cantidad de nodos expandidos por este método con relación al método de ramificación y acotación.*
- *Demostrar con un ejemplo hecho a mano que si la heurística no fuera consistente no se aseguraría el carácter óptimo de la búsqueda (opcional).*

Realización de los alumnos

Decidimos modificar la cola previamente creada, *OrderedQueue*, cuyo cambio fue que decidimos ordenar la lista según insertáramos los elementos, en vez de a la hora de sacarlos. Además, añadimos un parámetro lambda (f), que define nuestra función heurística.

```
class OrderedQueue(Queue):
    """A queue that pops its elements in order of least cost using heuristics."""

    def __init__(self, f=lambda x: 0):
        self.A = []
        self.f = f

    def append(self, item):
        bisect.insort(self.A, (item.path_cost + self.f(item), item))

    def __len__(self):
        return len(self.A)

    def pop(self):
        return self.A.pop(0)[1]
```

Luego, en el apartado de búsquedas, editamos la clase *Node*, en concreto el método *path*, para mostrar el coste acumulado, y desarrollamos el método *less than*, que compara dos nodos por su estado. Este último nos servirá para desambigüar los casos de empate en nuestra cola por orden alfabético. Modificamos el método *graph_search* para mostrar los nodos visitados, y desarrollamos los métodos *branch_and_bound_search* y *best_first_graph_search*. Ambas utilizarían esta nueva cola que habíamos desarrollado previamente.

```
class Node:
    """A node in a search tree. Contains a pointer to the parent (the node
    that this is a successor of) and to the actual state for this node. Note
    that if a state is arrived at by two paths, then there are two nodes with
    the same state.  Also includes the action that got us to this state, and
    the total path_cost (also known as g) to reach the node.  Other functions
    may add an f and h value; see best_first_graph_search and astar_search for
    an explanation of how the f and h values are handled. You will not need to
    subclass this class."""

    def __init__(self, state, parent=None, action=None, path_cost=0):
        """Create a search tree Node, derived from a parent by an action."""
        update(self, state=state, parent=parent, action=action,
               path_cost=path_cost, depth=0)
        if parent:
            self.depth = parent.depth + 1

    def __repr__(self):
        return "<Node %s(%s)>" % (self.state,self.path_cost)

    def __lt__(self, other):
        return self.state < other.state

    def path(self):
        """Create a list of nodes from the root to this node."""
        x, result = self, [self]
        while x.parent:
            result.append(x.parent)
            x = x.parent
        return result

    def expand(self, problem):
        """Return a list of nodes reachable from this node. [Fig. 3.8]"""
        return [Node(next, self, act,
                     problem.path_cost(self.path_cost, self.state, act, next))
                for (act, next) in problem.successor(self.state)]


# _____________________________________________________________________________
## Uninformed Search algorithms

```

```
def graph_search(problem, fringe):
    """Search through the successors of a problem to find a goal.
    The argument fringe should be an empty queue.
    If two paths reach a state, only use the best one. [Fig. 3.18]"""
    closed = {}
    stats = {}
    stats["visited"] = 0
    fringe.append(Node(problem.initial))
    while fringe:
        node = fringe.pop()
        stats["visited"] += 1
        if problem.goal_test(node.state):
            return (node, stats)
        if node.state not in closed:
            closed[node.state] = True
            fringe.extend(node.expand(problem))
    return (None, stats)


def breadth_first_graph_search(problem):
    """Search the shallowest nodes in the search tree first. [p 74]"""
    return graph_search(problem, FIFOQueue())  # FIFOQueue -> fringe


def depth_first_graph_search(problem):
    """Search the deepest nodes in the search tree first. [p 74]"""
    return graph_search(problem, Stack())

def branch_and_bound_search (problem):
    """Search using least cost node in the fringe """
    return graph_search(problem, OrderedQueue())

## Informed Search Algorithms
def branch_and_bound_heuristic_search(problem):
    return graph_search(problem, OrderedQueue(problem.h))
```

Y finalmente lo llamábamos en run.py, habiéndonos desarrollado el método visualizar para mostrar por pantalla el coste acumulado hasta el momento, así como los nodos visitados. Además, parametrizamos los nodos inicio y final, para interactuar de forma dinámica con el programa.

```
inicio = 'A' #@param ['A', 'B','C', 'D','E','F','G','H','I', 'L','N','O','P','R','S','T'
fin    = 'B' #@param ['A', 'B','C', 'D','E','F','G','H','I', 'L','N','O','P','R','S','T'

ab = GPSProblem(inicio, fin, romania)

def visualizar(a, name):
    print(name+"\t", a[1], "\t", a[0].path())

visualizar(breadth_first_graph_search(ab), "# Breadth First " + inicio + fin + "\t")
visualizar(depth_first_graph_search(ab), "# Depth First " + inicio + fin + "\t")
visualizar(branch_and_bound_search(ab), "# Branch and Bound " + inicio + fin + "\t")
visualizar(branch_and_bound_heuristic_search(ab), "# Branch and Bound H " + inicio + fin
```

inicio: A

fin: B

```
# Breadth First AB              {'visited': 16}         [<Node B(450)>, <Node F(239)>, <Node S(140)>, <Node A(0)>]
# Depth First AB                {'visited': 10}         [<Node B(733)>, <Node P(632)>, <Node C(494)>, <Node D(374)>, <Node M(299)>, <Node L(229)>, <Node T(118)>, <Node A(0)>]
# Branch and Bound AB           {'visited': 24}         [<Node B(418)>, <Node P(317)>, <Node R(220)>, <Node S(140)>, <Node A(0)>]
# Branch and Bound H AB         {'visited': 6}          [<Node B(418)>, <Node P(317)>, <Node R(220)>, <Node S(140)>, <Node A(0)>]
```

Segunda práctica - Redes neuronales.

Se pide desarrollar una red neuronal para la clasificación de un conjunto de imágenes. Este conjunto puede ser creado por el propio alumno hacinedo fotos de diferentes objetos en diferentes localizaciones y condiciones. Por ejemplo, libros, cubiertos de cocina, monedas, prendas de ropa... rollos de papel higiénico o lo que se tercie. El número recomendable de clases distintas será entre 4 y 7. Cada clase deberá tener un mínimo de 20 imágenes (cuantas más, mejor) para el conjunto de entrenamiento y 5 para el conjunto de validación. Una vez creado el dataset haremos lo siguiente:

- *Entrenar la red y visualizar gráficamente el progreso del accuracy tanto del conjunto de entrenamiento como del conjunto de validación. Para ello, se puede hacer uso del objeto history que devuelve el método fit_generator del modelo.*
- *Hacer data augmentation sobre el conjunto de entrenamiento.*
- *Probar con distintas configuraciones de hiperparámetros para escoger la que mejor resultados ofrezca.*
- *Categorical cross entropy es una función de pérdida similar a la ya estudiada suma de diferencias al cuadrado. Explica cómo y por qué funciona esta función de pérdida.*

Condiciones de la entrega:

- *La práctica se hará en grupo, de dos alumnos. No se permiten los grupos de tres o más alumnos. Tampoco se permitirá su realización de forma individual.*
- *Los resultados deberán presentarse contenidos en una memoria en PDF (no otro formato).*
- *El código y su funcionamiento se presentarán en una defensa por videoconferencia. Para ello será imprescindible activar la cámara del ordenador o, en su defecto, cámara del móvil. También será obligatorio mostrar el DNI.*
- *En cuanto al procedimiento de entrega y plazos para las defensas, se comunicarán más adelante.*

Realización de los alumnos

Establecimiento del entorno

En esta parte nos encargamos de establecer la conexión con Google Drive y cargar los datos en la máquina virtual. Además, definimos las funciones que utilizaremos más adelante para guardar los resultados de las diferentes pruebas. Creamos el fichero *parameters.json* en caso de que no exista, con el contenido apropiado, y si existe lo notificamos.

```
from google.colab import drive
drive.mount('/content/gdrive')

#@markdown ##Conectar a Google Drive { display-mode: "both" }
#@markdown Este trozo de código nos permite conectar nuestro drive, de donde sacaremos la
#@markdown Ejecutar siempre que empecemos una sesión.
```

```
Mounted at /content/gdrive
```

Conectar a Google Drive

Este trozo de código nos permite conectar nuestro drive, de donde sacaremos la información de los datasets y guardaremos el estado de nuestros modelos. Ejecutar siempre que empecemos una sesión.

Descargar DataSet

Esta parte del código determina la ruta en GDrive hasta el comprimido del DataSet. El ficheor se descargará y su contenido estará disponible en

/content/datasets/uncompressed

```
#@markdown ##Descargar Dataset
#@markdown Determina la ruta en GDrive hasta el comprimido de tu dataset:
datasetZip_path = "/ColabNotebooks/datasets/DogDataset.zip" #@param {type:"string"}
models_path = "/ColabNotebooks/models" #@param {type:"string"}
#@markdown El fichero se descargará y su contenido estará disponible en `/content/datase

#creamos carpetas si no estan
!mkdir /content/datasets
!mkdir ./datasets/compressed
!mkdir ./datasets/uncompressed

drive_ds_zip = "gdrive/MyDrive"+datasetZip_path
#drive_mod_dir = "gdrive/MyDrive"+models_path

#traemos el dataset alojado en drive a la maquina virtual y lo expandimos
!rsync -ah --progress $drive_ds_zip datasets/compressed/ds.zip
#descomprimimos
!unzip ./datasets/compressed/ds.zip -d ./datasets/uncompressed

```

Descargar Dataset

Determina la ruta en GDrive hasta el comprimido de tu dataset:

datasetZip_path: " /ColabNotebooks/datasets/DogDataset.zip "

models_path: " /ColabNotebooks/models "

El fichero se descargará y su contenido estará disponible en `/content/datasets/uncompressed`

```
sending incremental file list
DogDataset.zip
        54.62M 100%   21.45MB/s    0:00:02 (xfr#1, to-chk=0/1)

Archive:  ./datasets/compressed/ds.zip
   creating: ./datasets/uncompressed/test/
   creating: ./datasets/uncompressed/test/n02088364-beagle/
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_15315.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_15370.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_15690.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_15787.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_15877.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16060.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16065.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16165.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16207.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16210.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16339.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16493.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16502.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16508.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16519.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16588.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16635.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16689.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16695.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16704.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16721.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16791.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16881.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_16985.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17167.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17170.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17258.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17294.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17314.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17406.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17473.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17474.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17479.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17530.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17534.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17553.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17671.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17689.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17766.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_17935.jpg
  inflating: ./datasets/uncompressed/test/n02088364-beagle/n02088364_18403.jpg
   creating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4181.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4198.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4214.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4248.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4399.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4588.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4624.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4710.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4916.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_4990.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_5155.jpg
  inflating: ./datasets/uncompressed/test/n02094433-Yorkshire_terrier/n02094433_5176.jpg
```

Funciones de persistencia de modelos

Funciones útiles para guardar y cargar los datos de los modelos. Necesario antes de cualquiera de las siguientes celdas.

```
import json
import datetime;

#@markdown ##Funciones de persistencia de modelos
#@markdown Funciones útiles para guardar y cargar los datos de los modelos. Necesario antes de cualquiera de las siguientes celdas
route_parameters_json = 'gdrive/MyDrive/ColabNotebooks/models/parameters.json' #@param
route = 'gdrive/MyDrive/ColabNotebooks/models/' #@param

import os.path
if os.path.isfile(route_parameters_json):
    print ("File exists")
else:
    initial_data = {}
    initial_data['modelNames'] = []
    initial_data['modelTrainingData'] = {}
    with open(route_parameters_json, 'w') as outfile:
        json.dump(initial_data, outfile)
    with open(route_parameters_json) as json_file:
      print(json.load(json_file))

def load_configuration_json():
  with open(route_parameters_json) as json_file:
    return json.load(json_file)

def save_modelname_to_json(name):
  with open(route_parameters_json) as json_file:
    data = json.load(json_file)
    modelNames = data['modelNames']
    if name not in modelNames:
      modelNames.append(name)
```

```
        data['modelTrainingData'][name] = history_initial_data()
        print (data)
        with open(route_parameters_json, 'w') as outfile:
          json.dump(data, outfile)

def history_initial_data():
  history={}
  history['accuracy']=[]
  history['loss']=[]
  history['val_accuracy']=[]
  history['val_loss']=[]
  data={}
  data['history']= history
  data['epochs']=0
  return data

def save_model(model_storage,model ):
  ct = datetime.datetime.now().strftime("_%Y%m%d_%H%M%S")
  model.save(model_storage + model.name + ct )
  save_modelname_to_json(model.name)

def load_models(tf):
  conf = load_configuration_json()
  loaded_models={}
  for model in conf['modelNames']:
    model_route = route + model +"*"
    model_versions = !ls $model_route -d
    loaded_models[model] = tf.keras.models.load_model(model_versions[-1])
  return loaded_models, conf

def save_config_to_json(data):
  with open(route_parameters_json, 'w') as outfile:
    json.dump(data, outfile)

print("Ultima ejecución: "+ datetime.datetime.now().strftime("%Y%m%d_%H%M%S"))
```

```
File exists
Ultima ejecución: 20210602_102247
```

Definición de modelos

En cada celda de este apartado se define un modelo en cada caso, con hiperparámetros diferentes. Los modelos son guardados en el Drive, en un estado inicial sin entrenamiento. En la fase de entrenamiento, cargaremos estos modelos desde el Drive y, tras entrenarlos, se guardará una versión entrenada de cada uno de ellos.

```
%tensorflow_version 2.x
import tensorflow as tf
import keras
#@markdown ##Nombra el Modelo actual
model_name = "Modelo001-Papafrita" #@param
model_storage = route
with tf.device('/device:GPU:0'):
  model = tf.keras.models.Sequential(name=model_name)
  model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
                   activation='relu',
                   input_shape=(150, 150, 3)))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(3, 3)))
  model.add(tf.keras.layers.Conv2D(64, (5, 5), activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(3, 3)))
  model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Dropout(0.25))
  model.add(tf.keras.layers.Flatten())
  model.add(tf.keras.layers.Dense(128, activation='relu'))
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(tf.keras.layers.Dense(9, activation='softmax'))

  model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=keras.optimizers.Adadelta(),
                metrics=['accuracy'])
  model.build( input_shape =(150, 150, 3))

  model.summary()

  save_model(model_storage, model)
```

Nombra el Modelo actual

model_name: "Modelo001-Papafrita"

```
Model: "Modelo001-Papafrita"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_15 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 49, 49, 32)        0
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 45, 45, 64)        51264
_________________________________________________________________
max_pooling2d_16 (MaxPooling (None, 15, 15, 64)        0
_________________________________________________________________
conv2d_17 (Conv2D)           (None, 13, 13, 64)        36928
_________________________________________________________________
max_pooling2d_17 (MaxPooling (None, 6, 6, 64)          0
_________________________________________________________________
dropout_12 (Dropout)         (None, 6, 6, 64)          0
_________________________________________________________________
flatten_6 (Flatten)          (None, 2304)              0
_________________________________________________________________
dense_12 (Dense)             (None, 128)               295040
_________________________________________________________________
dropout_13 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_13 (Dense)             (None, 9)                 1161
=================================================================
Total params: 385,289
Trainable params: 385,289
Non-trainable params: 0
_________________________________________________________________
INFO:tensorflow:Assets written to: gdrive/MyDrive/Colab Notebooks/models/Modelo001-Papafrita_20210529_001327/assets
{'modelNames': ['Modelo001-Papafrita'], 'modelTrainingData': {'Modelo001-Papafrita': {'history': {'accuracy': [], 'loss': [], 'val_accuracy': [], 'val_loss': []}, 'epochs': 0}}}
```

```
%tensorflow_version 2.x
import tensorflow as tf
import keras
#@markdown ##Nombra el Modelo actual
model_name = "Modelo002-ReyLeonsio" #@param
model_storage = route
```

Nombra el Modelo actual

model_name: "Modelo002-ReyLeonsio"

```
model_storage = route
with tf.device('/device:GPU:0'):
  model = tf.keras.models.Sequential(name=model_name)
  model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
                  activation='relu',
                  input_shape=(150, 150, 3)))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Conv2D(64, (3, 3), activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Dropout(0.25))
  model.add(tf.keras.layers.Flatten())
  model.add(tf.keras.layers.Dense(128, activation='relu'))
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(tf.keras.layers.Dense(9, activation='softmax'))

  model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=keras.optimizers.Adadelta(),
                metrics=['accuracy'])
  model.build( input_shape =(150, 150, 3))

  model.summary()

  save_model(model_storage, model)
```

```
Model: "Modelo002-ReyLeonsio"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_18 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_18 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_19 (Conv2D)           (None, 72, 72, 64)        18496
_________________________________________________________________
max_pooling2d_19 (MaxPooling (None, 36, 36, 64)        0
_________________________________________________________________
conv2d_20 (Conv2D)           (None, 34, 34, 64)        36928
_________________________________________________________________
max_pooling2d_20 (MaxPooling (None, 17, 17, 64)        0
_________________________________________________________________
dropout_14 (Dropout)         (None, 17, 17, 64)        0
_________________________________________________________________
flatten_7 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_14 (Dense)             (None, 128)               2367616
_________________________________________________________________
dropout_15 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_15 (Dense)             (None, 9)                 1161
=================================================================
Total params: 2,425,097
Trainable params: 2,425,097
Non-trainable params: 0
_________________________________________________________________
INFO:tensorflow:Assets written to: gdrive/MyDrive/Colab Notebooks/models/Modelo002-ReyLeonsio_20210529_001329/assets
{'modelNames': ['Modelo001-Papafrita', 'Modelo002-ReyLeonsio'], 'modelTrainingData': {'Modelo001-Papafrita': {'history': {'accuracy': [], 'loss': [], 'val_accuracy': [], 'val_loss': []},
```

```
%tensorflow_version 2.x
import tensorflow as tf
import keras
#@markdown ##Crear modelo
model_name = "Modelo003-JuanSinMiedo" #@param
model_storage = route
with tf.device('/device:GPU:0'):
  model = tf.keras.models.Sequential(name=model_name)
  model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
                  activation='relu',
                  input_shape=(150, 150, 3)))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Dropout(0.25))
  model.add(tf.keras.layers.Flatten())
  model.add(tf.keras.layers.Dense(128, activation='relu'))
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(tf.keras.layers.Dense(9, activation='softmax'))

  model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=keras.optimizers.Adadelta(),
                metrics=['accuracy'])
  model.build( input_shape =(150, 150, 3))

  model.summary()

  save_model(model_storage, model)
```

Crear modelo

model_name: "Modelo003-JuanSinMiedo"

```
Model: "Modelo003-JuanSinMiedo"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_21 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_21 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_22 (Conv2D)           (None, 72, 72, 128)       36992
_________________________________________________________________
max_pooling2d_22 (MaxPooling (None, 36, 36, 128)       0
_________________________________________________________________
dropout_16 (Dropout)         (None, 36, 36, 128)       0
_________________________________________________________________
flatten_8 (Flatten)          (None, 165888)            0
_________________________________________________________________
dense_16 (Dense)             (None, 128)               21233792
_________________________________________________________________
dropout_17 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_17 (Dense)             (None, 9)                 1161
=================================================================
Total params: 21,272,841
Trainable params: 21,272,841
Non-trainable params: 0
_________________________________________________________________
INFO:tensorflow:Assets written to: gdrive/MyDrive/Colab Notebooks/models/Modelo003-JuanSinMiedo_20210529_001334/assets
{'modelNames': ['Modelo001-Papafrita', 'Modelo002-ReyLeonsio', 'Modelo003-JuanSinMiedo'], 'modelTrainingData': {'Modelo001-Papafrita': {'history': {'accuracy': [], 'loss': [], 'val_accura
```

```
%tensorflow_version 2.x
```

```
import tensorflow as tf
import keras
#@markdown ##Crear modelo
model_name = "Modelo004-Calamardo" #@param
model_storage = route
with tf.device('/device:GPU:0'):
  model = tf.keras.models.Sequential(name=model_name)
  model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
                  activation='sigmoid',
                  input_shape=(150, 150, 3)))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Dropout(0.25))
  model.add(tf.keras.layers.Flatten())
  model.add(tf.keras.layers.Dense(256, activation='relu'))
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(tf.keras.layers.Dense(9, activation='softmax'))

  model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=keras.optimizers.Adadelta(),
                metrics=['accuracy'])
  model.build( input_shape =(150, 150, 3))

  model.summary()

  save_model(model_storage, model)
```

Crear modelo

model_name: "Modelo004-Calamardo"

```
Model: "Modelo004-Calamardo"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_23 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_23 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_24 (Conv2D)           (None, 72, 72, 128)       36992
_________________________________________________________________
max_pooling2d_24 (MaxPooling (None, 36, 36, 128)       0
_________________________________________________________________
dropout_18 (Dropout)         (None, 36, 36, 128)       0
_________________________________________________________________
flatten_9 (Flatten)          (None, 165888)            0
_________________________________________________________________
dense_18 (Dense)             (None, 256)               42467584
_________________________________________________________________
dropout_19 (Dropout)         (None, 256)               0
_________________________________________________________________
dense_19 (Dense)             (None, 9)                 2313
=================================================================
Total params: 42,507,785
Trainable params: 42,507,785
Non-trainable params: 0
_________________________________________________________________
INFO:tensorflow:Assets written to: gdrive/MyDrive/Colab Notebooks/models/Modelo004-Calamardo_20210529_001340/assets
{'modelNames': ['Modelo001-Papafrita', 'Modelo002-ReyLeonsio', 'Modelo003-JuanSinMiedo', 'Modelo004-Calamardo'], 'modelTrainingData': {'Modelo001-Papafrita': {'history': {'accuracy': [],
```

```
%tensorflow_version 2.x
import tensorflow as tf
import keras
#@markdown ##Crear modelo
model_name = "Modelo005-VendoOpelCorsa" #@param
model_storage = route
with tf.device('/device:GPU:0'):
  model = tf.keras.models.Sequential(name=model_name)
  model.add(tf.keras.layers.Conv2D(32, kernel_size=(3, 3),
                  activation='relu',
                  input_shape=(150, 150, 3)))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Conv2D(128, (3, 3), activation='relu'))
  model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
  model.add(tf.keras.layers.Dropout(0.25))
  model.add(tf.keras.layers.Flatten())
  model.add(tf.keras.layers.Dense(128, activation='relu'))
  model.add(tf.keras.layers.Dropout(0.5))
  model.add(tf.keras.layers.Dense(9, activation='softmax'))

  model.compile(loss=keras.losses.categorical_crossentropy,
                optimizer=keras.optimizers.SGD(),
                metrics=['accuracy'])
  model.build( input_shape =(150, 150, 3))

  model.summary()

  save_model(model_storage, model)
```

Crear modelo

model_name: "Modelo005-VendoOpelCorsa"

```
Model: "Modelo005-VendoOpelCorsa"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_25 (Conv2D)           (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d_25 (MaxPooling (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_26 (Conv2D)           (None, 72, 72, 128)       36992
_________________________________________________________________
max_pooling2d_26 (MaxPooling (None, 36, 36, 128)       0
_________________________________________________________________
dropout_20 (Dropout)         (None, 36, 36, 128)       0
_________________________________________________________________
flatten_10 (Flatten)         (None, 165888)            0
_________________________________________________________________
dense_20 (Dense)             (None, 128)               21233792
_________________________________________________________________
dropout_21 (Dropout)         (None, 128)               0
_________________________________________________________________
dense_21 (Dense)             (None, 9)                 1161
=================================================================
Total params: 21,272,841
Trainable params: 21,272,841
Non-trainable params: 0
_________________________________________________________________
INFO:tensorflow:Assets written to: gdrive/MyDrive/Colab Notebooks/models/Modelo005-VendoOpelCorsa_20210529_001343/assets
{'modelNames': ['Modelo001-Papafrita', 'Modelo002-ReyLeonsio', 'Modelo003-JuanSinMiedo', 'Modelo004-Calamardo', 'Modelo005-VendoOpelCorsa'], 'modelTrainingData': {'Modelo001-Papafrita': {
```

Diferencia entre modelos

	Conv2D	MaxPooling2D	Conv2D	MaxPooling2D	Conv2D	MaxPooling2D	Dropout	Flatten	Dense	Dropout	Dense
Papafrita	32; 3x3; relu	3x3	64; 5x5; relu	3x3	64; 3x3; relu	2x2	0.25	X	128; relu	0.5	9; softmax
ReyLeonsio	32; 3x3; relu	2x2	64; 3x3: relu	2x2	64; 3x3: relu	2x2	0.25	X	128; relu	0.5	9; softmax
JuanSinMiedo	32; 3x3; relu	2x2	128; 3x3; relu	2x2			0.25	x	128; relu	0.5	9; softmax
Calamardo	32; 3x3; sigmoid	2x2	128; 3x3; relu	2x2			0.25	X	256; relu	0.5	9; softmax
VendoOpelCorsa	32; 3x3; relu	2x2	128; 3x3; relu	2x2			0.25	X	128; relu	0.5	9; softmax

Entrenamiento de modelos

Agregamos *X* épocas de entrenamiento (epochs) a cada uno de los modelos. Por defecto se escogerán a todos los modelos para entrenarlos, sin embargo, hemos desarrollado una estrategia para excluir a ciertos modelos que se necesitarán especificar por su nombre (exclude_from_training, en forma de array de Strings).

```
#@markdown ##Entrenamiento de los modelos creados

%tensorflow_version 2.x
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator
from keras.models import Sequential
from keras.layers import Dense, Dropout
from keras.optimizers import RMSprop
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D, MaxPooling2D

from keras.callbacks import EarlyStopping
from keras import backend as K
import keras
from time import time

# DATA SOURCE ----------------------------------------------

batch_size = 20

train_data_dir = '/content/datasets/uncompressed/train/'
validation_data_dir = '/content/datasets/uncompressed/test/'
model_storage = route
train_datagen = ImageDataGenerator(
        rescale=1./255,
        rotation_range=15,
        zoom_range=0.1)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_data_dir,
        target_size=(150, 150),
        batch_size=batch_size,
        class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(150, 150),
        batch_size=batch_size,
        class_mode='categorical')

# MODEL ----------------------------------------------
loaded_models, config = load_models(tf)

# TRAINING ----------------------------------------------
epochs = 1#@param
exclude_from_training = ['Modelo001-Papafrita', 'Modelo002-ReyLeonsio', 'Modelo003-JuanS
for model_str in loaded_models:
  model = loaded_models[model_str]

  if model_str in exclude_from_training:
    continue
  model.summary()

  model_history = config['modelTrainingData'][model_str]
  H = model.fit(
          train_generator,
          epochs=epochs,
          validation_data=validation_generator
  )

# SAVING ----------------------------------------------
  model_history['epochs'] += epochs
  model_history['history']['accuracy'].extend( H.history["accuracy"] )
  model_history['history']['val_accuracy'].extend( H.history["val_accuracy"] )
  model_history['history']['loss'].extend( H.history["loss"] )
  model_history['history']['val_loss'].extend( H.history["val_loss"] )

  save_model(model_storage,model )
  save_config_to_json(config)
```

Entrenamiento de los modelos creados

epochs: 1

exclude_from_training: ['Modelo001-Papafrita', 'Modelo002-ReyLeonsio', 'Modelo003-JuanSir

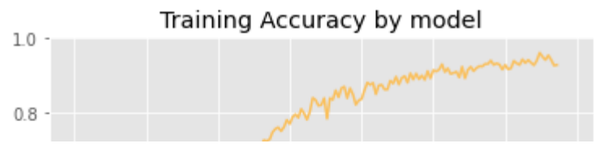
```
Found 1203 images belonging to 9 classes.
Found 274 images belonging to 9 classes.
Model: "Modelo005-VendoOpelCorsa"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 128)       36992
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 128)       0
_________________________________________________________________
dropout (Dropout)            (None, 36, 36, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 165888)            0
_________________________________________________________________
dense (Dense)                (None, 128)               21233792
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 9)                 1161
=================================================================
Total params: 21,272,841
```

```
Trainable params: 21,272,841
Non-trainable params: 0
_________________________________________________________________
61/61 [==============================] - 11s 170ms/step - loss: 0.1610 - accuracy: 0.9418 - val_loss: 3.6412 - val_accuracy: 0.3285
INFO:tensorflow:Assets written to: gdrive/MyDrive/ColabNotebooks/models/Modelo005-VendoOpelCorsa_20210602_111532/assets
```

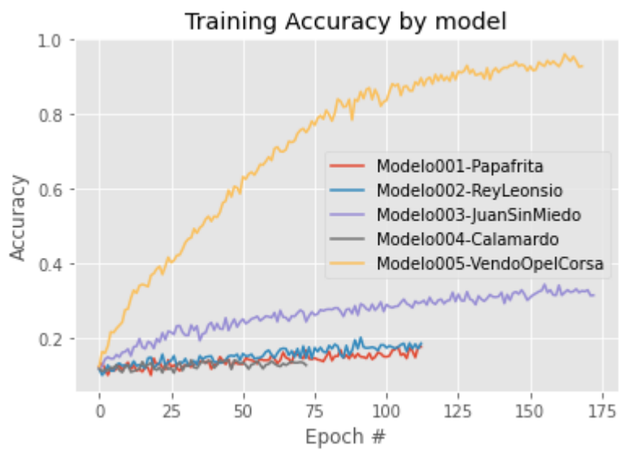
Análisis de resultados de forma gráfica

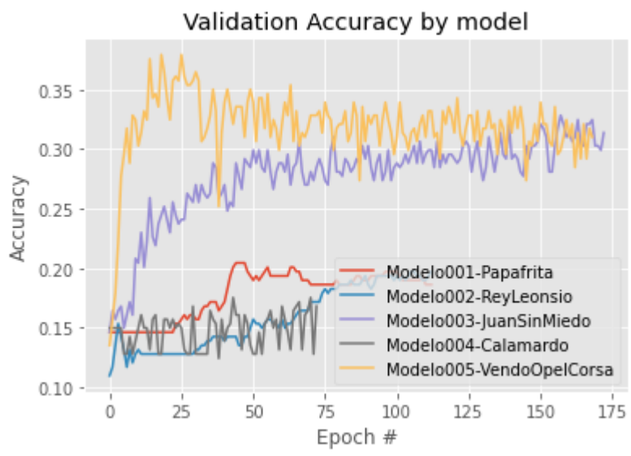
Creamos cuatro gráficas que ilustran la progresión de los diferentes modelos utilizando las métricas *accuracy* y *loss* tanto de entrenamiento como de validación.

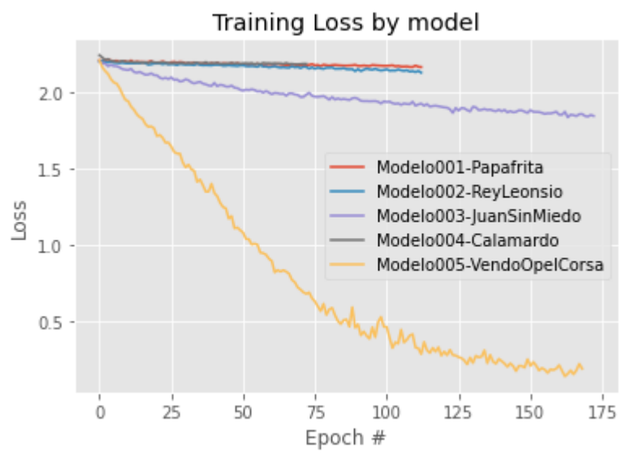
```
import matplotlib.pyplot as plt
import numpy as np

config = load_configuration_json()
plt.style.use("ggplot")

plt.figure()
for model_str in config['modelNames']:
  h = config['modelTrainingData'][model_str]
  plt.plot(np.arange(0, h['epochs']), h['history']["accuracy"], label=model_str)
  plt.title("Training Accuracy by model")
  plt.xlabel("Epoch #")
  plt.ylabel("Accuracy")
  plt.legend(loc="best")
  plt.savefig("accuracy.png")

plt.figure()
for model_str in config['modelNames']:
  h = config['modelTrainingData'][model_str]
  plt.plot(np.arange(0, h['epochs']), h['history']["val_accuracy"], label=model_str)
  plt.title("Validation Accuracy by model")
  plt.xlabel("Epoch #")
  plt.ylabel("Accuracy")
  plt.legend(loc="best")
  plt.savefig("val_accuracy.png")

plt.figure()
for model_str in config['modelNames']:
  h = config['modelTrainingData'][model_str]
  plt.plot(np.arange(0, h['epochs']), h['history']["loss"], label=model_str)
  plt.title("Training Loss by model")
  plt.xlabel("Epoch #")
  plt.ylabel("Loss")
  plt.legend(loc="best")
  plt.savefig("loss.png")

plt.figure()
for model_str in config['modelNames']:
  h = config['modelTrainingData'][model_str]
  plt.plot(np.arange(0, h['epochs']), h['history']["val_loss"], label=model_str)
  plt.title("Validation Loss by model")
  plt.xlabel("Epoch #")
  plt.ylabel("Loss")
  plt.legend(loc="best")
  plt.savefig("val_loss.png")
```

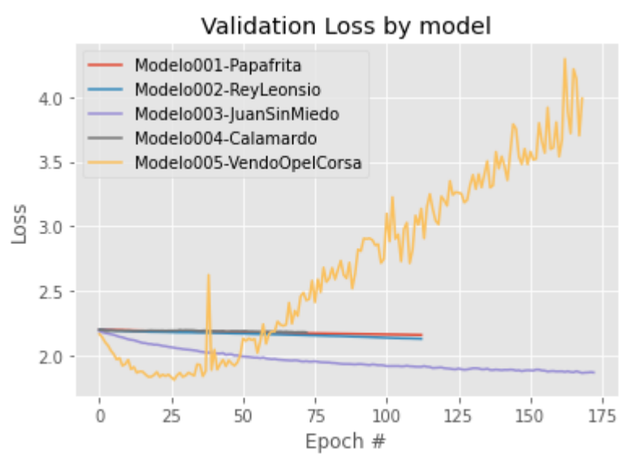


Conclusiones









De las gráficas anteriores interpretamos que, a pesar de que el modelo VendoOpelCorsa identifica muy bien las imágenes de entrenamiento, no es capaz de generalizar correctamente a la hora de ver imágenes que no ha visto anteriormente. También observamos el impacto del algoritmo optimizador utilizado (Adadelta para JuanSinMiedo, SGD para VendoOpelCorsa), donde aunque el GSD parece el más prometedor al principio, Adadelta termina obteniendo el mejor resultado. En vista de los resultados, a partir de aquí lo conveniente sería iterar sobre el modelo JuanSinMiedo para el diseño del siguiente modelo.

Matriz de confusión

En este apartado mostramos la matriz de confusión de uno de nuestros modelos. Para especificar el modelo, éste se debe introducir de manera manual.

```
modelName = 'Modelo005-VendoOpelCorsa' #@param ['Modelo001-Papafrita', 'Modelo002-ReyLeo
mostrarPredicciones = True #@param {type:"boolean"}
matrix_as_text = True #@param {type:"boolean"}
model = loaded_models[modelName]

validation_generator = test_datagen.flow_from_directory(
        validation_data_dir,
        target_size=(150, 150),
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False)
predictions = model.predict(validation_generator)

import numpy as np
y_pred = np.argmax(predictions, axis=1)
y_real = validation_generator.classes
if mostrarPredicciones:
  print("Valores de predicción:\n", y_pred)
  print("Valores reales:\n", y_real)

from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_real,y_pred)
if matrix_as_text:
  print("Matriz de confusión:\n", conf_matrix)

import seaborn as sns
import matplotlib.pyplot as plt

ax= plt.subplot()

```

modelName: Modelo005-VendoOpelCorsa

mostrarPredicciones: ☑

matrix_as_text: ☑

```
g = sns.heatmap(conf_matrix, annot=True, fmt='g', ax=ax); #annot=True to annotate cells,
g.set_yticklabels(g.get_yticklabels(), rotation = 0, fontsize = 8)
g.set_xticklabels(g.get_xticklabels(), rotation = 80, fontsize = 8,)

labels =!ls datasets/uncompressed/test/* -d
trimmed_labels =[]
for label in labels:
  trimmed_labels.append('-'.join(label.split('-')[1:]))

ax.set_xlabel('Predicted labels');ax.set_ylabel('True labels');
ax.set_title('Confusion Matrix: ' + modelName);
ax.xaxis.set_ticklabels(trimmed_labels);
ax.yaxis.set_ticklabels(trimmed_labels);
```

```
Found 274 images belonging to 9 classes.
Valores de predicción:
 [8 0 1 0 7 5 7 3 0 0 3 8 0 4 0 0 0 8 3 7 3 8 2 2 0 0 1 3 4 4 3 0 0 8 7 3 2
 0 8 8 8 0 7 1 6 5 4 4 1 3 1 0 3 4 6 3 3 4 2 8 3 1 8 0 4 7 3 6 3 8 1 3 1 4
 3 2 2 2 2 3 4 2 8 0 8 3 2 2 2 6 8 2 2 8 8 8 8 3 3 3 2 8 2 4 1 4 1 0 2 1 4
 1 3 3 7 3 3 4 0 3 3 3 3 3 4 3 5 1 7 8 5 2 5 3 8 0 2 7 3 3 5 7 6 0 1 3 3 4
 3 1 1 4 3 4 3 8 3 3 5 5 5 7 1 7 5 5 6 5 2 5 6 0 6 5 0 5 8 5 4 5 6 5 5 7 0
 1 8 1 3 6 3 1 7 6 3 5 2 5 8 0 8 2 7 7 3 6 3 7 7 3 8 0 8 3 1 0 7 7 1 7 7 1
 8 6 3 0 8 7 2 0 7 4 3 8 4 7 3 7 7 8 8 8 0 6 4 2 8 6 7 8 8 3 8 7 6 8 8 0 8
 8 4 1 7 8 8 0 8 8 7 8 2 7 7 8]
Valores reales:
 [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3
 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4
 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6
 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 6 7 7 7 7 7 7 7 7 7 7 7 7 7 7
 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 7 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8
 8 8 8 8 8 8 8 8 8 8 8 8 8 8 8]
Matriz de confusión:
 [[13  2  3  7  3  1  0  4  8]
 [ 3  6  1  8  6  1  3  2  3]
 [ 1  0 12  6  1  0  1  0  8]
 [ 2  4  1 10  5  0  0  1  0]
 [ 2  4  2 10  3  5  1  3  3]
 [ 2  1  1  0  1 12  4  3  1]
 [ 2  3  2  5  0  2  3  5  3]
 [ 4  3  1  5  2  0  1  9  5]
 [ 3  1  2  1  2  0  3  6 17]]
```