Redes Neuronales Convolucionales

Predicción de Perros y Gatos

Introducción

El uso de la inteligencia artificial ha ido en aumento en los últimos años, esto debido a que ha facilitado la automatización de muchos tareas.

El reconocimiento de objetos suele ser una tarea fácil para el ser humano, sin embargo, para una inteligencia artificial no lo es, es por ello que el uso de imágenes captcha para la detección de bots ha sido bastante efectiva hasta hace poco.

Durante este proyecto se abordará la problemática de clasificación de mascotas, pues este es un sector en crecimiento y con aplicaciones muy diversas.

El proyecto tiene como punto de partida la clasificación de perros y gatos la cual tendrá muchas aplicaciones en el campo de la Veterinaria conforme el modelo vaya siendo más complejo como por ejemplo, la detección de razas, anomalías o diagnósticos.

Bibliotecas utilizadas

In [ ]:
```
# General librarys
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.image import imread
from os import listdir

# Tensorflow-keras librarys
import tensorflow as tf

# Keras models
from keras.models import Sequential

# Keras Layers
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from keras.layers import Dropout

# Keras preprocessing module
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array

# Sckit Learn Library
from sklearn.model_selection import train_test_split
```

Plot Sample Data

Impresión de algunas fotos de perros.

In [ ]:
```
# Ruta de acceso a las imágenes
folder = "train/"

plt.figure(figsize=(10,8))
for i in range(9):
    plt.subplot(330+1+i)
    filename = folder+"cat."+str(i)+".jpg"
    image = imread(filename)
    plt.imshow(image)
plt.show()
```

Transformación de datos

El set de datos a utilizar en este proyecto consta de 25,000 imágenes de perros y gatos divididas de manera equitativa (12,500 por cada uno). Para este proyecto solo se utilizará.

En este paso se realiza una transformación de datos, primero se redimensionan las imágenes a un tamaño fijo de 100 x 100 pixeles y se convertirán a una sola capa de colores (escala de grises), después se transforma el formato, pasando de imágenes en una carpeta a un arreglo de numpy que contiene todas las imágenes, las dimensiones de este arreglo es (25000, 100, 100, 1).

In [ ]:
```
folder = "train/"
photos, labels = list(), list()

for file in listdir(folder):
    output = 0.0
    if file.startswith("dog"):
        output = 1.0

    photo = load_img(folder+file, target_size=(100,100), color_mode="grayscale")
    photo = img_to_array(photo)

    photos.append(photo)
    labels.append(output)

photos = np.asarray(photos)
```

```
labels = np.asarray(labels)
print(photos.shape, labels.shape)

np.save("dogs_vs_cats_photos.npy", photos)
np.save("dogs_vs_cats_labels.npy", labels)
```

```
(25000, 100, 100, 1) (25000,)
```

Impresión de fotos transformadas

In [ ]:
```
plt.figure(figsize=(10, 8))
for i in range(9):
    plt.subplot(3,3,i+1)
    plt.imshow(photos[i], cmap="gray")
plt.show()
```

Cargando datos desde archivo numpy

Durante este apartado se carga el arreglo numpy, después se normaliza los valores diviendo entre 255 para que quede en un rango de 0 a 1.

- Se tomarán 10,000 imágenes como conjunto de entrenamiento, de las cuales 2,000 servirán como set de validación.
- El set de prueba consta de 5,000 imágenes, lo cual nos servirá para probar el rendimiento del modelo.

In [ ]:
```
# Loading data
photos = np.load("dogs_vs_cats_photos.npy")
labels = np.load("dogs_vs_cats_labels.npy")

# Normalizing data
photos = photos/255

# Split data in training and test
X_train, X_test, y_train, y_test = train_test_split(
    photos, labels, test_size=0.20, random_state=20)

X_train = X_train[:10000]
y_train = y_train[:10000]

print(photos.shape, labels.shape)
print(X_train.shape, X_test.shape)
```

```
(25000, 100, 100, 1) (25000,)
(10000, 100, 100, 1) (5000, 100, 100, 1)
```

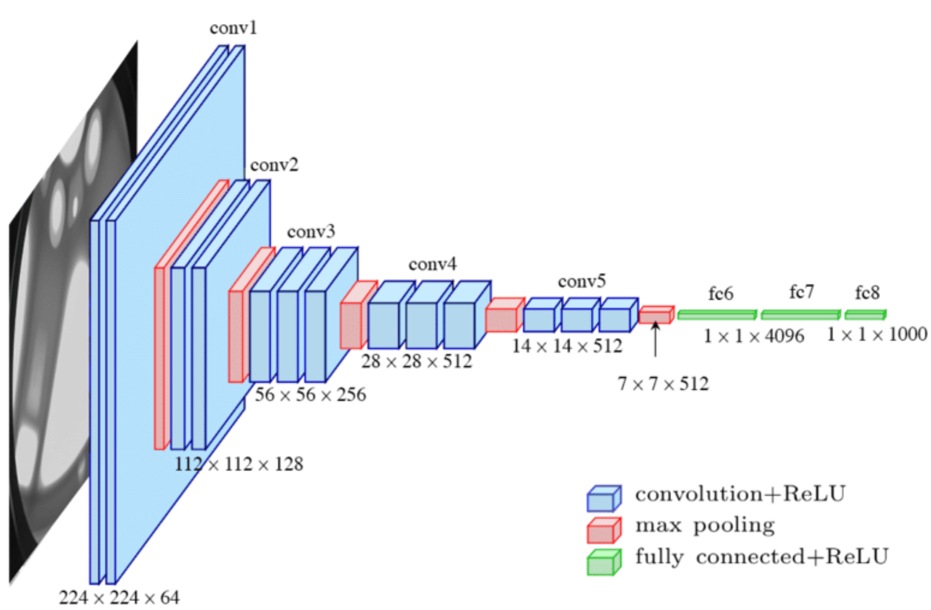
Redes Neuronales VGG

Se utilizará un modelo VGG (Visual Geometry Group) de Red Neural Convolucional. Éstos modelos han funcionado bien para el reconocimiento de imágenes de gran tamaño.

La profundidad de estas redes suele referirse al número de capas y bloques que compone la red:

- VGG-16: 16 capas convolucionales.
- VGG-19: 19 capas convolucionales.

Ejemplo de una arquitectura VGG-16



Funciones de la red

Función curvas de aprendizaje

Esta función nos permite graficar las curvas de aprendizaje (cross entropy y accuracy) para conocer el comportamiento de la Red Neuronal.

In [ ]:
```
def learning_curves(history):
    """Plot learning curves

    Args:
        history: history provided by fit method in keras
    """
    plt.figure(figsize=(12,5))
    plt.subplot(121)
    plt.title("Cross Entropy Loss")
    plt.plot(history.history["loss"], color="blue", label="train")
    plt.plot(history.history["val_loss"], color="orange", label="validation")
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
    plt.legend()

    plt.subplot(122)
    plt.title("Classification Accuracy")
    plt.plot(history.history["accuracy"], color="blue", label="train")
    plt.plot(history.history["val_accuracy"], color="orange", label="validation")
    plt.xlabel("Epochs")
    plt.ylabel("Accuracy")
    plt.legend()
    plt.show()
```

Fit and plot model

Con esta función correremos el modelo creado, aquí se pueden modificar los hiperparámetros de la red.

Para este proyecto el modelo se corrió tomando como hiperparámetros,

- Epocas = 10
- Tamaño de batch = 250
- Set de validación del 20% (2,000)
- Se habilitó el uso de múltiples núcleos

In [ ]:
```
def run_test(model: Sequential, X_train: np.array, y_train: np.array, X_test: np.array, y_test: np.array):
    """Run the training process and evaluate keras models.

    Args:
        model (Sequential): Sequential keras model.
        X_train (np.array): Features training set.
        y_train (np.array): Label training set.
        X_test (np.array): Features test set.
        y_test (np.array): Label test set.
    """
    # Fitting process
    print("Fitting model")
    history = model.fit(X_train,y_train,
                        epochs=50, batch_size=32,
                        validation_split=0.20,
                        workers=10, use_multiprocessing=True,
                        verbose=1)

    # Evaluating accuracy
    print("\nEvaluate model\n")
    _, acc = model.evaluate(X_test, y_test,verbose=1)
    print("Accuracy = %.3f" % (acc*100.0))
    return history, acc
```

Creando Modelos

Modelo VGG (base)

Utilizaremos un modelo VGG como modelo base para la comparación con otros modelos.

- Para cada uno de los modelos se utilizó accuracy como método de evaluación.

La configuración de la red se encuentra documentado en el código.

In [ ]:
```
def VGG():
    """Create a baseline model.

    Returns:
        model (Sequential): Baseline Sequential model for fit.
    """
    # Create sequential model
    model = Sequential()

    # Block 1
    model.add(Conv2D(32, (3, 3), activation="relu",
              kernel_initializer="he_uniform", padding="same", input_shape=(100, 100, 1)))
    model.add(MaxPooling2D((2, 2)))

    # Flatten and output
    model.add(Flatten())
    model.add(Dense(128, activation="relu", kernel_initializer="he_uniform"))
    model.add(Dense(1, activation="sigmoid"))

    # Compile model
    opt = tf.keras.optimizers.SGD(learning_rate=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss="binary_crossentropy",
                  metrics=["accuracy"])
    return model
```

Modelo VGG-2

Modelo VGG-2 con dos bloques.

In [ ]:
```
def VGG2():
    """Create a baseline model.

    Returns:
        model (Sequential): Baseline Sequential model for fit.
    """
    # Create sequential model
    model = Sequential()

    # Block 1
    model.add(Conv2D(32, (3, 3), activation="relu",
              kernel_initializer="he_uniform", padding="same", input_shape=(100, 100, 1)))
    model.add(MaxPooling2D((2, 2)))

    # Block 2
    model.add(Conv2D(64, (3, 3), activation="relu",
              kernel_initializer="he_uniform", padding="same", input_shape=(100, 100, 1)))
    model.add(MaxPooling2D((2, 2)))

    # Flatten and output
```

```
    model.add(Flatten())
    model.add(Dense(128, activation="relu", kernel_initializer="he_uniform"))
    model.add(Dense(1, activation="sigmoid"))

    # Compile model
    opt = tf.keras.optimizers.SGD(learning_rate=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss="binary_crossentropy",
                  metrics=["accuracy"])
    return model
```

Modelo VGG-3

Modelo VGG-3 con tres bloques.

In [ ]:
```
def VGG3():
    """Create a baseline model.

    Returns:
        model (Sequential): Baseline Sequential model for fit.
    """
    # Create sequential model
    model = Sequential()

    # Block 1
    model.add(Conv2D(32, (3, 3), activation="relu",
              kernel_initializer="he_uniform", padding="same", input_shape=(100, 100, 1)))
    model.add(MaxPooling2D((2, 2)))

    # Block 2
    model.add(Conv2D(64, (3, 3), activation="relu",
              kernel_initializer="he_uniform", padding="same", input_shape=(100, 100, 1)))
    model.add(MaxPooling2D((2, 2)))

    # Block 3
    model.add(Conv2D(128, (3, 3), activation="relu",
              kernel_initializer="he_uniform", padding="same", input_shape=(100, 100, 1)))
    model.add(MaxPooling2D((2, 2)))

    # Flatten and output
    model.add(Flatten())
    model.add(Dense(128, activation="relu", kernel_initializer="he_uniform"))
    model.add(Dense(1, activation="sigmoid"))

    # Compile model
    opt = tf.keras.optimizers.SGD(learning_rate=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss="binary_crossentropy",
                  metrics=["accuracy"])
    return model
```

Modelo Dropout

El siguiente modelo también es un modelo VGG-2, pero a éste se le ha aplicado la regularización Dropout.

In [ ]:
```
def vgg_dropout():
    model = Sequential()
    # Block 1
    model.add(Conv2D(32,(3,3), activation="relu", kernel_initializer="he_uniform",
                     padding="same", input_shape=(100,100,1)))
    model.add(MaxPooling2D((2,2)))
    model.add(Dropout(0.2))

    # Block 2
    model.add(Conv2D(64,(3,3), activation="relu", kernel_initializer="he_uniform",
                     padding="same", input_shape=(100,100,1)))
    model.add(MaxPooling2D((2,2)))
    model.add(Dropout(0.2))

    # Flatten and output
    model.add(Flatten())
    model.add(Dense(128, activation="relu", kernel_initializer="he_uniform"))
    model.add(Dropout(0.2))
    model.add(Dense(1, activation="sigmoid"))

    # Compile model
    opt = tf.keras.optimizers.SGD(learning_rate=0.001, momentum=0.9)
    model.compile(optimizer=opt, loss="binary_crossentropy",
                  metrics=["accuracy"])
    return model
```

Evaluando modelos

Modelo VGG (Base)

Compilación del modelo

In [ ]:
```
# Creating model
model = VGG()
model.summary()
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 100, 100, 32)      320

 max_pooling2d_4 (MaxPooling  (None, 50, 50, 32)       0
 2D)

 flatten_4 (Flatten)         (None, 80000)             0

 dense_8 (Dense)             (None, 128)               10240128

 dense_9 (Dense)             (None, 1)                 129

=================================================================
Total params: 10,240,577
Trainable params: 10,240,577
Non-trainable params: 0
_________________________________________________________________
```

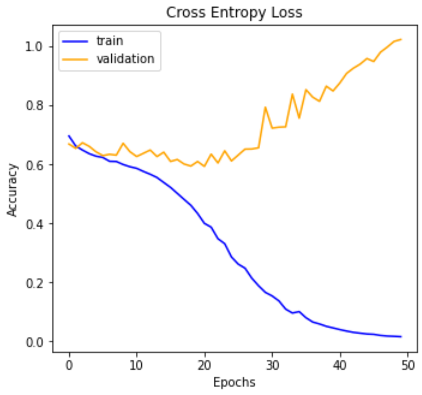
Entrenamiento del modelo

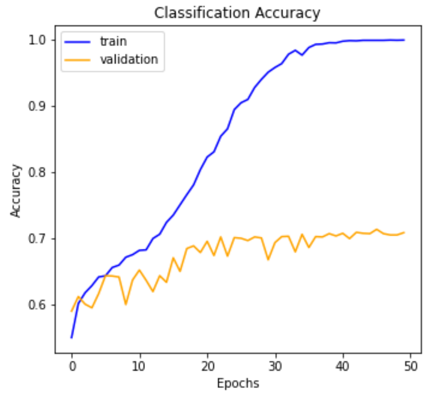
In [ ]:
```
history, acc = run_test(model, X_train, y_train, X_test, y_test)
```

```
Fitting model
Epoch 1/50
250/250 [==============================] - 34s 133ms/step - loss: 0.6968 - accuracy: 0.5505 - val_loss: 0.6696 - val_accuracy: 0.5905
Epoch 2/50
250/250 [==============================] - 33s 131ms/step - loss: 0.6629 - accuracy: 0.6020 - val_loss: 0.6548 - val_accuracy: 0.6125
Epoch 3/50
250/250 [==============================] - 33s 131ms/step - loss: 0.6491 - accuracy: 0.6183 - val_loss: 0.6739 - val_accuracy: 0.6010
Epoch 4/50
250/250 [==============================] - 33s 131ms/step - loss: 0.6367 - accuracy: 0.6290 - val_loss: 0.6615 - val_accuracy: 0.5955
Epoch 5/50
250/250 [==============================] - 33s 131ms/step - loss: 0.6284 - accuracy: 0.6421 - val_loss: 0.6431 - val_accuracy: 0.6170
Epoch 6/50
250/250 [==============================] - 33s 131ms/step - loss: 0.6239 - accuracy: 0.6438 - val_loss: 0.6303 - val_accuracy: 0.6440
Epoch 7/50
250/250 [==============================] - 33s 130ms/step - loss: 0.6110 - accuracy: 0.6564 - val_loss: 0.6347 - val_accuracy: 0.6435
Epoch 8/50
250/250 [==============================] - 33s 132ms/step - loss: 0.6103 - accuracy: 0.6600 - val_loss: 0.6319 - val_accuracy: 0.6420
Epoch 9/50
250/250 [==============================] - 32s 130ms/step - loss: 0.6001 - accuracy: 0.6719 - val_loss: 0.6720 - val_accuracy: 0.6005
Epoch 10/50
250/250 [==============================] - 34s 135ms/step - loss: 0.5926 - accuracy: 0.6758 - val_loss: 0.6436 - val_accuracy: 0.6375
Epoch 11/50
250/250 [==============================] - 33s 132ms/step - loss: 0.5871 - accuracy: 0.6823 - val_loss: 0.6272 - val_accuracy: 0.6525
Epoch 12/50
250/250 [==============================] - 34s 137ms/step - loss: 0.5766 - accuracy: 0.6830 - val_loss: 0.6381 - val_accuracy: 0.6370
Epoch 13/50
250/250 [==============================] - 33s 133ms/step - loss: 0.5671 - accuracy: 0.7001 - val_loss: 0.6492 - val_accuracy: 0.6200
Epoch 14/50
250/250 [==============================] - 33s 133ms/step - loss: 0.5560 - accuracy: 0.7066 - val_loss: 0.6270 - val_accuracy: 0.6440
Epoch 15/50
250/250 [==============================] - 33s 133ms/step - loss: 0.5392 - accuracy: 0.7245 - val_loss: 0.6419 - val_accuracy: 0.6340
Epoch 16/50
250/250 [==============================] - 33s 134ms/step - loss: 0.5225 - accuracy: 0.7355 - val_loss: 0.6105 - val_accuracy: 0.6710
Epoch 17/50
250/250 [==============================] - 34s 137ms/step - loss: 0.5023 - accuracy: 0.7510 - val_loss: 0.6171 - val_accuracy: 0.6505
Epoch 18/50
250/250 [==============================] - 34s 136ms/step - loss: 0.4819 - accuracy: 0.7664 - val_loss: 0.6018 - val_accuracy: 0.6850
Epoch 19/50
250/250 [==============================] - 35s 141ms/step - loss: 0.4618 - accuracy: 0.7811 - val_loss: 0.5948 - val_accuracy: 0.6890
Epoch 20/50
250/250 [==============================] - 32s 130ms/step - loss: 0.4341 - accuracy: 0.8043 - val_loss: 0.6103 - val_accuracy: 0.6790
Epoch 21/50
250/250 [==============================] - 32s 129ms/step - loss: 0.4007 - accuracy: 0.8231 - val_loss: 0.5937 - val_accuracy: 0.6960
Epoch 22/50
250/250 [==============================] - 33s 130ms/step - loss: 0.3876 - accuracy: 0.8310 - val_loss: 0.6353 - val_accuracy: 0.6745
Epoch 23/50
250/250 [==============================] - 32s 130ms/step - loss: 0.3486 - accuracy: 0.8545 - val_loss: 0.6052 - val_accuracy: 0.7025
Epoch 24/50
250/250 [==============================] - 33s 130ms/step - loss: 0.3316 - accuracy: 0.8658 - val_loss: 0.6466 - val_accuracy: 0.6735
Epoch 25/50
250/250 [==============================] - 32s 129ms/step - loss: 0.2866 - accuracy: 0.8947 - val_loss: 0.6121 - val_accuracy: 0.7015
Epoch 26/50
250/250 [==============================] - 32s 130ms/step - loss: 0.2623 - accuracy: 0.9051 - val_loss: 0.6322 - val_accuracy: 0.7005
Epoch 27/50
250/250 [==============================] - 33s 130ms/step - loss: 0.2487 - accuracy: 0.9101 - val_loss: 0.6522 - val_accuracy: 0.6970
Epoch 28/50
250/250 [==============================] - 33s 134ms/step - loss: 0.2148 - accuracy: 0.9281 - val_loss: 0.6527 - val_accuracy: 0.7025
Epoch 29/50
250/250 [==============================] - 32s 129ms/step - loss: 0.1891 - accuracy: 0.9404 - val_loss: 0.6567 - val_accuracy: 0.7010
Epoch 30/50
250/250 [==============================] - 32s 129ms/step - loss: 0.1667 - accuracy: 0.9513 - val_loss: 0.7944 - val_accuracy: 0.6680
Epoch 31/50
250/250 [==============================] - 32s 129ms/step - loss: 0.1546 - accuracy: 0.9584 - val_loss: 0.7234 - val_accuracy: 0.6940
Epoch 32/50
250/250 [==============================] - 32s 130ms/step - loss: 0.1379 - accuracy: 0.9643 - val_loss: 0.7266 - val_accuracy: 0.7030
Epoch 33/50
250/250 [==============================] - 32s 129ms/step - loss: 0.1099 - accuracy: 0.9784 - val_loss: 0.7277 - val_accuracy: 0.7035
Epoch 34/50
250/250 [==============================] - 32s 129ms/step - loss: 0.0965 - accuracy: 0.9843 - val_loss: 0.8388 - val_accuracy: 0.6800
Epoch 35/50
250/250 [==============================] - 32s 129ms/step - loss: 0.1015 - accuracy: 0.9770 - val_loss: 0.7571 - val_accuracy: 0.7065
Epoch 36/50
250/250 [==============================] - 32s 129ms/step - loss: 0.0807 - accuracy: 0.9885 - val_loss: 0.8538 - val_accuracy: 0.6865
Epoch 37/50
250/250 [==============================] - 32s 130ms/step - loss: 0.0661 - accuracy: 0.9933 - val_loss: 0.8284 - val_accuracy: 0.7030
Epoch 38/50
250/250 [==============================] - 33s 130ms/step - loss: 0.0595 - accuracy: 0.9937 - val_loss: 0.8143 - val_accuracy: 0.7025
Epoch 39/50
250/250 [==============================] - 30s 121ms/step - loss: 0.0516 - accuracy: 0.9958 - val_loss: 0.8656 - val_accuracy: 0.7075
Epoch 40/50
250/250 [==============================] - 30s 121ms/step - loss: 0.0461 - accuracy: 0.9955 - val_loss: 0.8494 - val_accuracy: 0.7040
Epoch 41/50
250/250 [==============================] - 31s 122ms/step - loss: 0.0405 - accuracy: 0.9980 - val_loss: 0.8765 - val_accuracy: 0.7080
Epoch 42/50
250/250 [==============================] - 30s 122ms/step - loss: 0.0354 - accuracy: 0.9989 - val_loss: 0.9090 - val_accuracy: 0.7000
Epoch 43/50
250/250 [==============================] - 30s 122ms/step - loss: 0.0311 - accuracy: 0.9986 - val_loss: 0.9265 - val_accuracy: 0.7095
Epoch 44/50
250/250 [==============================] - 31s 125ms/step - loss: 0.0283 - accuracy: 0.9994 - val_loss: 0.9404 - val_accuracy: 0.7080
Epoch 45/50
250/250 [==============================] - 33s 131ms/step - loss: 0.0256 - accuracy: 0.9994 - val_loss: 0.9594 - val_accuracy: 0.7075
Epoch 46/50
250/250 [==============================] - 32s 130ms/step - loss: 0.0244 - accuracy: 0.9994 - val_loss: 0.9493 - val_accuracy: 0.7140
Epoch 47/50
250/250 [==============================] - 33s 130ms/step - loss: 0.0207 - accuracy: 0.9994 - val_loss: 0.9807 - val_accuracy: 0.7075
Epoch 48/50
250/250 [==============================] - 33s 134ms/step - loss: 0.0186 - accuracy: 0.9998 - val_loss: 0.9981 - val_accuracy: 0.7055
Epoch 49/50
250/250 [==============================] - 33s 130ms/step - loss: 0.0176 - accuracy: 0.9995 - val_loss: 1.0173 - val_accuracy: 0.7055
Epoch 50/50
250/250 [==============================] - 34s 135ms/step - loss: 0.0164 - accuracy: 0.9998 - val_loss: 1.0238 - val_accuracy: 0.7090

Evaluate model

157/157 [==============================] - 5s 30ms/step - loss: 1.0142 - accuracy: 0.6988
Accuracy = 69.880
```

Curvas de aprendizaje

In [ ]:
```
learning_curves(history)
```





Modelo VGG-2

Compilación del modelo

In [ ]:
```
model = VGG2()
model.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_7 (Conv2D)           (None, 100, 100, 32)      320       
                                                                 
 max_pooling2d_7 (MaxPooling  (None, 50, 50, 32)       0         
 2D)                                                             
                                                                 
 conv2d_8 (Conv2D)           (None, 50, 50, 64)        18496     
                                                                 
 max_pooling2d_8 (MaxPooling  (None, 25, 25, 64)       0         
 2D)                                                             
                                                                 
 flatten_6 (Flatten)         (None, 40000)             0         
                                                                 
 dense_12 (Dense)            (None, 128)               5120128   
                                                                 
 dense_13 (Dense)            (None, 1)                 129       
                                                                 
=================================================================
Total params: 5,139,073
Trainable params: 5,139,073
Non-trainable params: 0
_________________________________________________________________
```

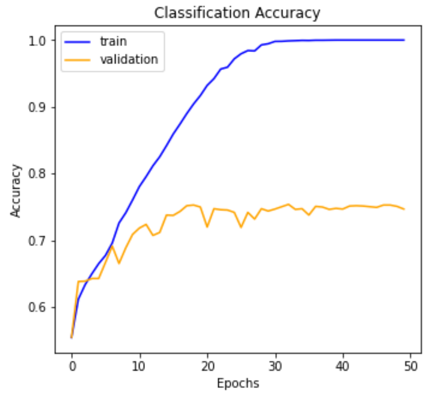
Entrenando del modelo

In [ ]:
```
history, acc = run_test(model, X_train, y_train, X_test, y_test)
```

```
Fitting model
Epoch 1/50
250/250 [==============================] - 49s 194ms/step - loss: 0.6890 - accuracy: 0.5545 - val_loss: 0.6755 - val_accuracy: 0.5550
Epoch 2/50
250/250 [==============================] - 51s 203ms/step - loss: 0.6554 - accuracy: 0.6120 - val_loss: 0.6397 - val_accuracy: 0.6385
Epoch 3/50
250/250 [==============================] - 51s 203ms/step - loss: 0.6389 - accuracy: 0.6338 - val_loss: 0.6304 - val_accuracy: 0.6390
Epoch 4/50
250/250 [==============================] - 51s 202ms/step - loss: 0.6246 - accuracy: 0.6501 - val_loss: 0.6221 - val_accuracy: 0.6430
Epoch 5/50
250/250 [==============================] - 51s 202ms/step - loss: 0.6100 - accuracy: 0.6653 - val_loss: 0.6238 - val_accuracy: 0.6430
Epoch 6/50
250/250 [==============================] - 50s 201ms/step - loss: 0.5916 - accuracy: 0.6775 - val_loss: 0.6039 - val_accuracy: 0.6670
Epoch 7/50
250/250 [==============================] - 49s 198ms/step - loss: 0.5760 - accuracy: 0.6961 - val_loss: 0.5827 - val_accuracy: 0.6915
Epoch 8/50
250/250 [==============================] - 49s 194ms/step - loss: 0.5498 - accuracy: 0.7261 - val_loss: 0.6043 - val_accuracy: 0.6655
Epoch 9/50
250/250 [==============================] - 49s 194ms/step - loss: 0.5280 - accuracy: 0.7415 - val_loss: 0.5859 - val_accuracy: 0.6890
Epoch 10/50
250/250 [==============================] - 50s 199ms/step - loss: 0.4963 - accuracy: 0.7606 - val_loss: 0.5510 - val_accuracy: 0.7090
Epoch 11/50
250/250 [==============================] - 50s 201ms/step - loss: 0.4704 - accuracy: 0.7806 - val_loss: 0.5451 - val_accuracy: 0.7185
Epoch 12/50
250/250 [==============================] - 50s 202ms/step - loss: 0.4464 - accuracy: 0.7956 - val_loss: 0.5311 - val_accuracy: 0.7240
Epoch 13/50
250/250 [==============================] - 49s 197ms/step - loss: 0.4259 - accuracy: 0.8117 - val_loss: 0.5687 - val_accuracy: 0.7075
Epoch 14/50
250/250 [==============================] - 50s 202ms/step - loss: 0.3971 - accuracy: 0.8253 - val_loss: 0.5789 - val_accuracy: 0.7120
Epoch 15/50
250/250 [==============================] - 51s 203ms/step - loss: 0.3720 - accuracy: 0.8419 - val_loss: 0.5347 - val_accuracy: 0.7380
Epoch 16/50
250/250 [==============================] - 50s 201ms/step - loss: 0.3379 - accuracy: 0.8594 - val_loss: 0.5371 - val_accuracy: 0.7375
Epoch 17/50
250/250 [==============================] - 51s 202ms/step - loss: 0.3121 - accuracy: 0.8744 - val_loss: 0.5274 - val_accuracy: 0.7435
Epoch 18/50
250/250 [==============================] - 51s 202ms/step - loss: 0.2819 - accuracy: 0.8900 - val_loss: 0.5372 - val_accuracy: 0.7520
Epoch 19/50
250/250 [==============================] - 51s 203ms/step - loss: 0.2541 - accuracy: 0.9044 - val_loss: 0.5393 - val_accuracy: 0.7530
Epoch 20/50
250/250 [==============================] - 50s 201ms/step - loss: 0.2335 - accuracy: 0.9170 - val_loss: 0.5540 - val_accuracy: 0.7500
Epoch 21/50
250/250 [==============================] - 51s 204ms/step - loss: 0.2010 - accuracy: 0.9321 - val_loss: 0.6362 - val_accuracy: 0.7200
Epoch 22/50
250/250 [==============================] - 50s 202ms/step - loss: 0.1824 - accuracy: 0.9423 - val_loss: 0.5744 - val_accuracy: 0.7475
Epoch 23/50
250/250 [==============================] - 50s 202ms/step - loss: 0.1543 - accuracy: 0.9566 - val_loss: 0.6055 - val_accuracy: 0.7460
Epoch 24/50
250/250 [==============================] - 49s 195ms/step - loss: 0.1418 - accuracy: 0.9594 - val_loss: 0.6045 - val_accuracy: 0.7455
Epoch 25/50
250/250 [==============================] - 49s 195ms/step - loss: 0.1211 - accuracy: 0.9719 - val_loss: 0.6419 - val_accuracy: 0.7420
Epoch 26/50
250/250 [==============================] - 49s 195ms/step - loss: 0.1024 - accuracy: 0.9795 - val_loss: 0.7360 - val_accuracy: 0.7195
Epoch 27/50
250/250 [==============================] - 49s 196ms/step - loss: 0.0890 - accuracy: 0.9843 - val_loss: 0.6774 - val_accuracy: 0.7420
Epoch 28/50
250/250 [==============================] - 49s 195ms/step - loss: 0.0833 - accuracy: 0.9837 - val_loss: 0.7475 - val_accuracy: 0.7320
Epoch 29/50
250/250 [==============================] - 50s 201ms/step - loss: 0.0617 - accuracy: 0.9927 - val_loss: 0.7158 - val_accuracy: 0.7475
Epoch 30/50
250/250 [==============================] - 51s 202ms/step - loss: 0.0511 - accuracy: 0.9945 - val_loss: 0.7219 - val_accuracy: 0.7440
Epoch 31/50
250/250 [==============================] - 51s 203ms/step - loss: 0.0426 - accuracy: 0.9980 - val_loss: 0.7410 - val_accuracy: 0.7470
Epoch 32/50
250/250 [==============================] - 51s 203ms/step - loss: 0.0369 - accuracy: 0.9981 - val_loss: 0.7762 - val_accuracy: 0.7505
Epoch 33/50
250/250 [==============================] - 50s 202ms/step - loss: 0.0305 - accuracy: 0.9987 - val_loss: 0.7787 - val_accuracy: 0.7540
Epoch 34/50
250/250 [==============================] - 51s 203ms/step - loss: 0.0268 - accuracy: 0.9991 - val_loss: 0.8055 - val_accuracy: 0.7465
Epoch 35/50
250/250 [==============================] - 51s 202ms/step - loss: 0.0223 - accuracy: 0.9995 - val_loss: 0.8278 - val_accuracy: 0.7475
Epoch 36/50
250/250 [==============================] - 51s 203ms/step - loss: 0.0210 - accuracy: 0.9994 - val_loss: 0.8707 - val_accuracy: 0.7380
Epoch 37/50
250/250 [==============================] - 51s 204ms/step - loss: 0.0190 - accuracy: 0.9998 - val_loss: 0.8529 - val_accuracy: 0.7510
Epoch 38/50
250/250 [==============================] - 51s 205ms/step - loss: 0.0158 - accuracy: 0.9998 - val_loss: 0.8770 - val_accuracy: 0.7500
Epoch 39/50
250/250 [==============================] - 52s 208ms/step - loss: 0.0136 - accuracy: 0.9999 - val_loss: 0.8852 - val_accuracy: 0.7465
Epoch 40/50
250/250 [==============================] - 50s 200ms/step - loss: 0.0123 - accuracy: 1.0000 - val_loss: 0.8968 - val_accuracy: 0.7480
Epoch 41/50
250/250 [==============================] - 49s 196ms/step - loss: 0.0111 - accuracy: 1.0000 - val_loss: 0.9274 - val_accuracy: 0.7470
Epoch 42/50
250/250 [==============================] - 50s 198ms/step - loss: 0.0107 - accuracy: 1.0000 - val_loss: 0.9259 - val_accuracy: 0.7515
Epoch 43/50
250/250 [==============================] - 52s 206ms/step - loss: 0.0098 - accuracy: 1.0000 - val_loss: 0.9274 - val_accuracy: 0.7520
Epoch 44/50
250/250 [==============================] - 50s 201ms/step - loss: 0.0087 - accuracy: 1.0000 - val_loss: 0.9421 - val_accuracy: 0.7515
Epoch 45/50
250/250 [==============================] - 51s 202ms/step - loss: 0.0079 - accuracy: 1.0000 - val_loss: 0.9702 - val_accuracy: 0.7505
Epoch 46/50
250/250 [==============================] - 50s 200ms/step - loss: 0.0078 - accuracy: 1.0000 - val_loss: 0.9594 - val_accuracy: 0.7495
Epoch 47/50
250/250 [==============================] - 51s 206ms/step - loss: 0.0069 - accuracy: 1.0000 - val_loss: 0.9700 - val_accuracy: 0.7530
Epoch 48/50
250/250 [==============================] - 50s 199ms/step - loss: 0.0065 - accuracy: 1.0000 - val_loss: 0.9728 - val_accuracy: 0.7530
Epoch 49/50
250/250 [==============================] - 51s 203ms/step - loss: 0.0061 - accuracy: 1.0000 - val_loss: 0.9867 - val_accuracy: 0.7510
Epoch 50/50
250/250 [==============================] - 51s 204ms/step - loss: 0.0057 - accuracy: 1.0000 - val_loss: 0.9980 - val_accuracy: 0.7470

Evaluate model

157/157 [==============================] - 7s 41ms/step - loss: 1.0470 - accuracy: 0.7444
Accuracy = 74.440
```

Curvas de aprendizaje

In [ ]:
```
learning_curves(history)
```



Guardar Modelo

In [ ]:
```
model.save(r"export_model\vgg2_model_weights.h5")
```

Modelo VGG-3

Compilación del modelo

In [ ]:
```
model = VGG3()
model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_12 (Conv2D)          (None, 100, 100, 32)      320       
                                                                 
 max_pooling2d_12 (MaxPoolin  (None, 50, 50, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_13 (Conv2D)          (None, 50, 50, 64)        18496     
                                                                 
 max_pooling2d_13 (MaxPoolin  (None, 25, 25, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_14 (Conv2D)          (None, 25, 25, 128)       73856     
                                                                 
 max_pooling2d_14 (MaxPoolin  (None, 12, 12, 128)      0         
 g2D)                                                            
                                                                 
 flatten_8 (Flatten)         (None, 18432)             0         
                                                                 
 dense_16 (Dense)            (None, 128)               2359424   
                                                                 
 dense_17 (Dense)            (None, 1)                 129       
                                                                 
=================================================================
Total params: 2,452,225
Trainable params: 2,452,225
Non-trainable params: 0
_________________________________________________________________
```

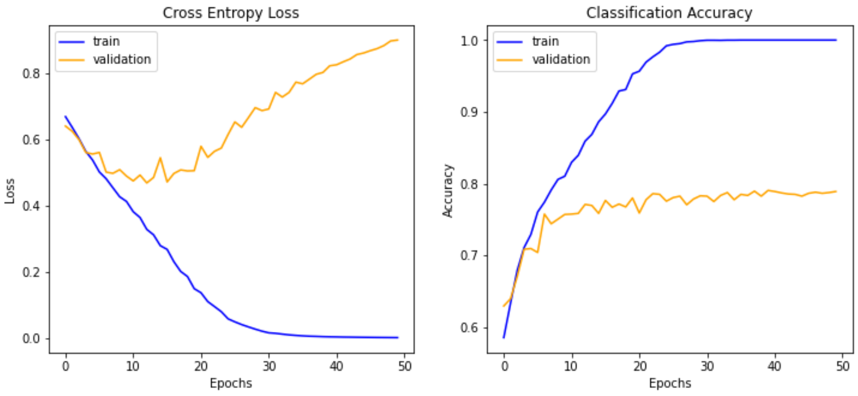
Entrenamiento del modelo

In [ ]:
```
history, acc = run_test(model, X_train, y_train, X_test, y_test)
```

```
Fitting model
Epoch 1/50
250/250 [==============================] - 61s 243ms/step - loss: 0.6679 - accuracy: 0.5854 - val_loss: 0.6395 - val_accuracy: 0.6295
Epoch 2/50
250/250 [==============================] - 61s 243ms/step - loss: 0.6355 - accuracy: 0.6335 - val_loss: 0.6237 - val_accuracy: 0.6395
Epoch 3/50
250/250 [==============================] - 62s 248ms/step - loss: 0.6016 - accuracy: 0.6795 - val_loss: 0.5986 - val_accuracy: 0.6710
Epoch 4/50
250/250 [==============================] - 63s 253ms/step - loss: 0.5632 - accuracy: 0.7109 - val_loss: 0.5600 - val_accuracy: 0.7085
Epoch 5/50
250/250 [==============================] - 63s 253ms/step - loss: 0.5372 - accuracy: 0.7291 - val_loss: 0.5554 - val_accuracy: 0.7095
Epoch 6/50
250/250 [==============================] - 65s 258ms/step - loss: 0.5013 - accuracy: 0.7604 - val_loss: 0.5602 - val_accuracy: 0.7040
Epoch 7/50
250/250 [==============================] - 63s 252ms/step - loss: 0.4810 - accuracy: 0.7741 - val_loss: 0.5009 - val_accuracy: 0.7575
Epoch 8/50
250/250 [==============================] - 65s 258ms/step - loss: 0.4537 - accuracy: 0.7911 - val_loss: 0.4973 - val_accuracy: 0.7440
Epoch 9/50
250/250 [==============================] - 62s 249ms/step - loss: 0.4267 - accuracy: 0.8059 - val_loss: 0.5083 - val_accuracy: 0.7505
Epoch 10/50
250/250 [==============================] - 62s 248ms/step - loss: 0.4126 - accuracy: 0.8102 - val_loss: 0.4890 - val_accuracy: 0.7570
Epoch 11/50
250/250 [==============================] - 62s 249ms/step - loss: 0.3816 - accuracy: 0.8296 - val_loss: 0.4744 - val_accuracy: 0.7575
Epoch 12/50
250/250 [==============================] - 64s 256ms/step - loss: 0.3642 - accuracy: 0.8395 - val_loss: 0.4922 - val_accuracy: 0.7585
Epoch 13/50
250/250 [==============================] - 61s 245ms/step - loss: 0.3285 - accuracy: 0.8591 - val_loss: 0.4683 - val_accuracy: 0.7710
Epoch 14/50
250/250 [==============================] - 60s 242ms/step - loss: 0.3120 - accuracy: 0.8686 - val_loss: 0.4848 - val_accuracy: 0.7695
Epoch 15/50
250/250 [==============================] - 64s 256ms/step - loss: 0.2794 - accuracy: 0.8861 - val_loss: 0.5442 - val_accuracy: 0.7585
Epoch 16/50
250/250 [==============================] - 64s 255ms/step - loss: 0.2683 - accuracy: 0.8970 - val_loss: 0.4713 - val_accuracy: 0.7765
Epoch 17/50
250/250 [==============================] - 67s 267ms/step - loss: 0.2311 - accuracy: 0.9118 - val_loss: 0.4968 - val_accuracy: 0.7670
Epoch 18/50
250/250 [==============================] - 62s 248ms/step - loss: 0.2019 - accuracy: 0.9290 - val_loss: 0.5075 - val_accuracy: 0.7715
Epoch 19/50
250/250 [==============================] - 62s 248ms/step - loss: 0.1866 - accuracy: 0.9311 - val_loss: 0.5045 - val_accuracy: 0.7675
Epoch 20/50
250/250 [==============================] - 62s 249ms/step - loss: 0.1500 - accuracy: 0.9529 - val_loss: 0.5052 - val_accuracy: 0.7800
Epoch 21/50
250/250 [==============================] - 65s 260ms/step - loss: 0.1376 - accuracy: 0.9566 - val_loss: 0.5786 - val_accuracy: 0.7590
Epoch 22/50
250/250 [==============================] - 61s 243ms/step - loss: 0.1110 - accuracy: 0.9693 - val_loss: 0.5453 - val_accuracy: 0.7775
Epoch 23/50
250/250 [==============================] - 60s 240ms/step - loss: 0.0960 - accuracy: 0.9766 - val_loss: 0.5636 - val_accuracy: 0.7860
Epoch 24/50
250/250 [==============================] - 61s 242ms/step - loss: 0.0806 - accuracy: 0.9831 - val_loss: 0.5734 - val_accuracy: 0.7850
Epoch 25/50
250/250 [==============================] - 66s 263ms/step - loss: 0.0591 - accuracy: 0.9920 - val_loss: 0.6144 - val_accuracy: 0.7755
Epoch 26/50
250/250 [==============================] - 67s 267ms/step - loss: 0.0500 - accuracy: 0.9940 - val_loss: 0.6521 - val_accuracy: 0.7805
Epoch 27/50
250/250 [==============================] - 67s 267ms/step - loss: 0.0418 - accuracy: 0.9951 - val_loss: 0.6359 - val_accuracy: 0.7825
Epoch 28/50
250/250 [==============================] - 67s 268ms/step - loss: 0.0348 - accuracy: 0.9975 - val_loss: 0.6651 - val_accuracy: 0.7705
Epoch 29/50
250/250 [==============================] - 67s 270ms/step - loss: 0.0281 - accuracy: 0.9980 - val_loss: 0.6951 - val_accuracy: 0.7785
Epoch 30/50
250/250 [==============================] - 67s 268ms/step - loss: 0.0220 - accuracy: 0.9991 - val_loss: 0.6861 - val_accuracy: 0.7830
Epoch 31/50
250/250 [==============================] - 66s 265ms/step - loss: 0.0170 - accuracy: 0.9998 - val_loss: 0.6910 - val_accuracy: 0.7825
Epoch 32/50
250/250 [==============================] - 67s 269ms/step - loss: 0.0156 - accuracy: 0.9998 - val_loss: 0.7409 - val_accuracy: 0.7750
Epoch 33/50
250/250 [==============================] - 67s 268ms/step - loss: 0.0131 - accuracy: 0.9996 - val_loss: 0.7269 - val_accuracy: 0.7835
Epoch 34/50
250/250 [==============================] - 66s 266ms/step - loss: 0.0108 - accuracy: 0.9999 - val_loss: 0.7410 - val_accuracy: 0.7875
Epoch 35/50
250/250 [==============================] - 59s 238ms/step - loss: 0.0092 - accuracy: 0.9999 - val_loss: 0.7718 - val_accuracy: 0.7775
Epoch 36/50
250/250 [==============================] - 66s 264ms/step - loss: 0.0079 - accuracy: 1.0000 - val_loss: 0.7668 - val_accuracy: 0.7850
Epoch 37/50
250/250 [==============================] - 67s 267ms/step - loss: 0.0068 - accuracy: 1.0000 - val_loss: 0.7810 - val_accuracy: 0.7835
Epoch 38/50
250/250 [==============================] - 60s 238ms/step - loss: 0.0062 - accuracy: 1.0000 - val_loss: 0.7954 - val_accuracy: 0.7895
Epoch 39/50
250/250 [==============================] - 59s 237ms/step - loss: 0.0055 - accuracy: 1.0000 - val_loss: 0.8011 - val_accuracy: 0.7825
Epoch 40/50
250/250 [==============================] - 62s 248ms/step - loss: 0.0050 - accuracy: 1.0000 - val_loss: 0.8211 - val_accuracy: 0.7905
Epoch 41/50
250/250 [==============================] - 64s 257ms/step - loss: 0.0047 - accuracy: 1.0000 - val_loss: 0.8241 - val_accuracy: 0.7890
Epoch 42/50
250/250 [==============================] - 75s 301ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.8331 - val_accuracy: 0.7870
Epoch 43/50
250/250 [==============================] - 76s 305ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.8416 - val_accuracy: 0.7855
Epoch 44/50
250/250 [==============================] - 64s 257ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.8549 - val_accuracy: 0.7850
Epoch 45/50
250/250 [==============================] - 62s 250ms/step - loss: 0.0034 - accuracy: 1.0000 - val_loss: 0.8595 - val_accuracy: 0.7825
Epoch 46/50
250/250 [==============================] - 62s 248ms/step - loss: 0.0032 - accuracy: 1.0000 - val_loss: 0.8667 - val_accuracy: 0.7865
Epoch 47/50
250/250 [==============================] - 62s 249ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.8729 - val_accuracy: 0.7880
Epoch 48/50
250/250 [==============================] - 68s 273ms/step - loss: 0.0029 - accuracy: 1.0000 - val_loss: 0.8821 - val_accuracy: 0.7865
Epoch 49/50
250/250 [==============================] - 68s 272ms/step - loss: 0.0027 - accuracy: 1.0000 - val_loss: 0.8959 - val_accuracy: 0.7875
Epoch 50/50
250/250 [==============================] - 67s 267ms/step - loss: 0.0026 - accuracy: 1.0000 - val_loss: 0.8985 - val_accuracy: 0.7890

Evaluate model

157/157 [==============================] - 8s 52ms/step - loss: 1.0222 - accuracy: 0.7732
Accuracy = 77.320
```

Curvas de aprendizaje

In [ ]:
```
learning_curves(history)
```



Guardar Modelo

In [ ]:
```
model.save(r"export_model\vgg3_model_weights.h5")
```

Modelo VGG2-Dropout

Compilación del modelo

In [ ]:
```
model = vgg_dropout()
model.summary()
```

```
Model: "sequential_11"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_19 (Conv2D)          (None, 100, 100, 32)      320       
                                                                 
 max_pooling2d_19 (MaxPoolin  (None, 50, 50, 32)       0         
 g2D)                                                            
                                                                 
 dropout_6 (Dropout)         (None, 50, 50, 32)        0         
                                                                 
 conv2d_20 (Conv2D)          (None, 50, 50, 64)        18496     
                                                                 
 max_pooling2d_20 (MaxPoolin  (None, 25, 25, 64)       0         
 g2D)                                                            
                                                                 
 dropout_7 (Dropout)         (None, 25, 25, 64)        0         
                                                                 
 flatten_11 (Flatten)        (None, 40000)             0         
                                                                 
 dense_22 (Dense)            (None, 128)               5120128   
                                                                 
 dropout_8 (Dropout)         (None, 128)               0         
                                                                 
 dense_23 (Dense)            (None, 1)                 129       
                                                                 
=================================================================
Total params: 5,139,073
Trainable params: 5,139,073
Non-trainable params: 0
_________________________________________________________________
```

Entrenamiento del modelo

In [ ]:
```
history, acc = run_test(model, X_train, y_train, X_test, y_test)
```

```
Fitting model
Epoch 1/50
250/250 [==============================] - 57s 227ms/step - loss: 0.7052 - accuracy: 0.5336 - val_loss: 0.6827 - val_accuracy: 0.5810
Epoch 2/50
250/250 [==============================] - 54s 216ms/step - loss: 0.6807 - accuracy: 0.5605 - val_loss: 0.6772 - val_accuracy: 0.5800
Epoch 3/50
250/250 [==============================] - 54s 216ms/step - loss: 0.6727 - accuracy: 0.5830 - val_loss: 0.6684 - val_accuracy: 0.6030
Epoch 4/50
250/250 [==============================] - 54s 216ms/step - loss: 0.6701 - accuracy: 0.5801 - val_loss: 0.6640 - val_accuracy: 0.6285
Epoch 5/50
250/250 [==============================] - 54s 215ms/step - loss: 0.6636 - accuracy: 0.5907 - val_loss: 0.6570 - val_accuracy: 0.6295
Epoch 6/50
250/250 [==============================] - 54s 216ms/step - loss: 0.6558 - accuracy: 0.6118 - val_loss: 0.6549 - val_accuracy: 0.6300
Epoch 7/50
250/250 [==============================] - 54s 217ms/step - loss: 0.6454 - accuracy: 0.6183 - val_loss: 0.6550 - val_accuracy: 0.5970
Epoch 8/50
250/250 [==============================] - 55s 219ms/step - loss: 0.6465 - accuracy: 0.6151 - val_loss: 0.6478 - val_accuracy: 0.6395
Epoch 9/50
250/250 [==============================] - 57s 226ms/step - loss: 0.6376 - accuracy: 0.6315 - val_loss: 0.6488 - val_accuracy: 0.6105
Epoch 10/50
250/250 [==============================] - 56s 226ms/step - loss: 0.6347 - accuracy: 0.6385 - val_loss: 0.6377 - val_accuracy: 0.6395
Epoch 11/50
250/250 [==============================] - 57s 226ms/step - loss: 0.6271 - accuracy: 0.6415 - val_loss: 0.6313 - val_accuracy: 0.6440
Epoch 12/50
250/250 [==============================] - 57s 226ms/step - loss: 0.6280 - accuracy: 0.6395 - val_loss: 0.6391 - val_accuracy: 0.6215
Epoch 13/50
250/250 [==============================] - 56s 225ms/step - loss: 0.6273 - accuracy: 0.6445 - val_loss: 0.6244 - val_accuracy: 0.6575
Epoch 14/50
250/250 [==============================] - 56s 226ms/step - loss: 0.6220 - accuracy: 0.6464 - val_loss: 0.6386 - val_accuracy: 0.6140
Epoch 15/50
250/250 [==============================] - 57s 227ms/step - loss: 0.6219 - accuracy: 0.6555 - val_loss: 0.6336 - val_accuracy: 0.6545
Epoch 16/50
250/250 [==============================] - 57s 226ms/step - loss: 0.6122 - accuracy: 0.6603 - val_loss: 0.6314 - val_accuracy: 0.6320
Epoch 17/50
250/250 [==============================] - 57s 227ms/step - loss: 0.6034 - accuracy: 0.6649 - val_loss: 0.6179 - val_accuracy: 0.6600
Epoch 18/50
250/250 [==============================] - 57s 226ms/step - loss: 0.6027 - accuracy: 0.6711 - val_loss: 0.6271 - val_accuracy: 0.6265
Epoch 19/50
250/250 [==============================] - 58s 231ms/step - loss: 0.5997 - accuracy: 0.6719 - val_loss: 0.6294 - val_accuracy: 0.6230
Epoch 20/50
250/250 [==============================] - 57s 229ms/step - loss: 0.5911 - accuracy: 0.6814 - val_loss: 0.6198 - val_accuracy: 0.6455
Epoch 21/50
250/250 [==============================] - 57s 229ms/step - loss: 0.5808 - accuracy: 0.6883 - val_loss: 0.6235 - val_accuracy: 0.6305
Epoch 22/50
250/250 [==============================] - 58s 231ms/step - loss: 0.5755 - accuracy: 0.6911 - val_loss: 0.6142 - val_accuracy: 0.6560
Epoch 23/50
250/250 [==============================] - 60s 242ms/step - loss: 0.5645 - accuracy: 0.7038 - val_loss: 0.6138 - val_accuracy: 0.6515
Epoch 24/50
250/250 [==============================] - 61s 244ms/step - loss: 0.5576 - accuracy: 0.7075 - val_loss: 0.6094 - val_accuracy: 0.6565
Epoch 25/50
250/250 [==============================] - 61s 246ms/step - loss: 0.5506 - accuracy: 0.7138 - val_loss: 0.6172 - val_accuracy: 0.6420
Epoch 26/50
250/250 [==============================] - 61s 243ms/step - loss: 0.5368 - accuracy: 0.7184 - val_loss: 0.6015 - val_accuracy: 0.6655
Epoch 27/50
250/250 [==============================] - 61s 243ms/step - loss: 0.5334 - accuracy: 0.7294 - val_loss: 0.6213 - val_accuracy: 0.6430
Epoch 28/50
250/250 [==============================] - 60s 242ms/step - loss: 0.5196 - accuracy: 0.7406 - val_loss: 0.5875 - val_accuracy: 0.6855
Epoch 29/50
250/250 [==============================] - 60s 242ms/step - loss: 0.5046 - accuracy: 0.7500 - val_loss: 0.5758 - val_accuracy: 0.6970
Epoch 30/50
250/250 [==============================] - 60s 241ms/step - loss: 0.5017 - accuracy: 0.7524 - val_loss: 0.5853 - val_accuracy: 0.6795
Epoch 31/50
250/250 [==============================] - 60s 241ms/step - loss: 0.4842 - accuracy: 0.7653 - val_loss: 0.5593 - val_accuracy: 0.7120
Epoch 32/50
250/250 [==============================] - 56s 226ms/step - loss: 0.4705 - accuracy: 0.7729 - val_loss: 0.5649 - val_accuracy: 0.6995
Epoch 33/50
250/250 [==============================] - 57s 229ms/step - loss: 0.4615 - accuracy: 0.7799 - val_loss: 0.5518 - val_accuracy: 0.7135
Epoch 34/50
250/250 [==============================] - 57s 228ms/step - loss: 0.4472 - accuracy: 0.7875 - val_loss: 0.5725 - val_accuracy: 0.6995
Epoch 35/50
250/250 [==============================] - 57s 228ms/step - loss: 0.4353 - accuracy: 0.7959 - val_loss: 0.5445 - val_accuracy: 0.7145
Epoch 36/50
250/250 [==============================] - 57s 227ms/step - loss: 0.4272 - accuracy: 0.8021 - val_loss: 0.5427 - val_accuracy: 0.7140
Epoch 37/50
250/250 [==============================] - 57s 228ms/step - loss: 0.4054 - accuracy: 0.8146 - val_loss: 0.5372 - val_accuracy: 0.7275
Epoch 38/50
250/250 [==============================] - 57s 230ms/step - loss: 0.3934 - accuracy: 0.8185 - val_loss: 0.5533 - val_accuracy: 0.7145
Epoch 39/50
250/250 [==============================] - 57s 227ms/step - loss: 0.3786 - accuracy: 0.8319 - val_loss: 0.5394 - val_accuracy: 0.7200
Epoch 40/50
250/250 [==============================] - 57s 227ms/step - loss: 0.3669 - accuracy: 0.8389 - val_loss: 0.5354 - val_accuracy: 0.7240
Epoch 41/50
250/250 [==============================] - 57s 227ms/step - loss: 0.3525 - accuracy: 0.8406 - val_loss: 0.5287 - val_accuracy: 0.7270
Epoch 42/50
250/250 [==============================] - 57s 227ms/step - loss: 0.3373 - accuracy: 0.8550 - val_loss: 0.5438 - val_accuracy: 0.7230
Epoch 43/50
250/250 [==============================] - 57s 226ms/step - loss: 0.3235 - accuracy: 0.8591 - val_loss: 0.5331 - val_accuracy: 0.7315
Epoch 44/50
250/250 [==============================] - 57s 228ms/step - loss: 0.3153 - accuracy: 0.8666 - val_loss: 0.5340 - val_accuracy: 0.7330
Epoch 45/50
250/250 [==============================] - 56s 226ms/step - loss: 0.3031 - accuracy: 0.8709 - val_loss: 0.5438 - val_accuracy: 0.7245
Epoch 46/50
250/250 [==============================] - 56s 226ms/step - loss: 0.2772 - accuracy: 0.8894 - val_loss: 0.5395 - val_accuracy: 0.7330
Epoch 47/50
250/250 [==============================] - 56s 226ms/step - loss: 0.2663 - accuracy: 0.8904 - val_loss: 0.5679 - val_accuracy: 0.7280
Epoch 48/50
250/250 [==============================] - 57s 226ms/step - loss: 0.2584 - accuracy: 0.8965 - val_loss: 0.5524 - val_accuracy: 0.7345
Epoch 49/50
250/250 [==============================] - 57s 226ms/step - loss: 0.2512 - accuracy: 0.8971 - val_loss: 0.5593 - val_accuracy: 0.7380
Epoch 50/50
250/250 [==============================] - 56s 226ms/step - loss: 0.2343 - accuracy: 0.9075 - val_loss: 0.5470 - val_accuracy: 0.7365

Evaluate model

157/157 [==============================] - 7s 42ms/step - loss: 0.5721 - accuracy: 0.7348
Accuracy = 73.480
```

Curvas de aprendizaje

In [ ]:
```
learning_curves(history)
```

Exportar Modelo

```
In [ ]: model.save(r"export_model\vgg2_dropout_model.h5")
```

Model VGG3 + Data Augmentation

Compilación del modelo

```
In [ ]: model = VGG3()
        model.summary()
```

```
Model: "sequential_13"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d_24 (Conv2D)          (None, 100, 100, 32)      320       
                                                                 
 max_pooling2d_24 (MaxPoolin  (None, 50, 50, 32)       0         
 g2D)                                                            
                                                                 
 conv2d_25 (Conv2D)          (None, 50, 50, 64)        18496     
                                                                 
 max_pooling2d_25 (MaxPoolin  (None, 25, 25, 64)       0         
 g2D)                                                            
                                                                 
 conv2d_26 (Conv2D)          (None, 25, 25, 128)       73856     
                                                                 
 max_pooling2d_26 (MaxPoolin  (None, 12, 12, 128)      0         
 g2D)                                                            
                                                                 
 flatten_13 (Flatten)        (None, 18432)             0         
                                                                 
 dense_26 (Dense)            (None, 128)               2359424   
                                                                 
 dense_27 (Dense)            (None, 1)                 129       
                                                                 
=================================================================
Total params: 2,452,225
Trainable params: 2,452,225
Non-trainable params: 0
_________________________________________________________________
```

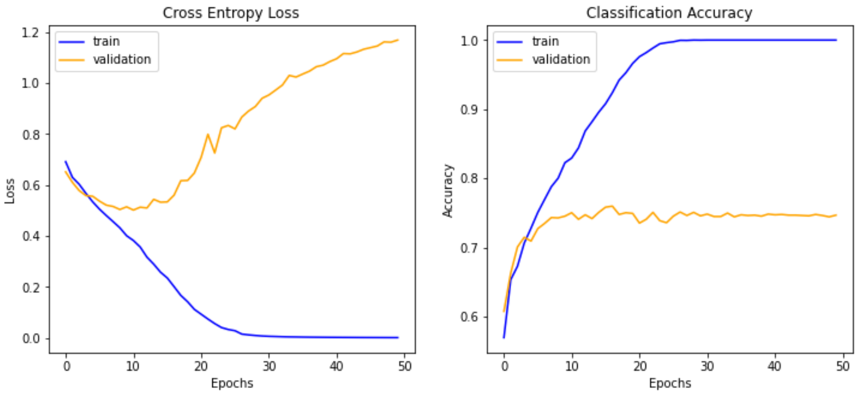
Entrenamiento del modelo

```
In [ ]: datagen = ImageDataGenerator(width_shift_range=0.1,
                                     height_shift_range=0.1,
                                     horizontal_flip=True)

        it = datagen.flow(X_train, y_train)

        history, acc = run_test(model, it.x, it.y, X_test, y_test)
```

```
Fitting model
Epoch 1/50
250/250 [==============================] - 63s 251ms/step - loss: 0.6912 - accuracy: 0.5700 - val_loss: 0.6512 - val_accuracy: 0.6080
Epoch 2/50
250/250 [==============================] - 61s 243ms/step - loss: 0.6298 - accuracy: 0.6536 - val_loss: 0.6102 - val_accuracy: 0.6630
Epoch 3/50
250/250 [==============================] - 59s 235ms/step - loss: 0.6025 - accuracy: 0.6731 - val_loss: 0.5780 - val_accuracy: 0.7010
Epoch 4/50
250/250 [==============================] - 59s 235ms/step - loss: 0.5667 - accuracy: 0.7066 - val_loss: 0.5580 - val_accuracy: 0.7150
Epoch 5/50
250/250 [==============================] - 59s 235ms/step - loss: 0.5336 - accuracy: 0.7283 - val_loss: 0.5555 - val_accuracy: 0.7095
Epoch 6/50
250/250 [==============================] - 59s 237ms/step - loss: 0.5050 - accuracy: 0.7509 - val_loss: 0.5364 - val_accuracy: 0.7275
Epoch 7/50
250/250 [==============================] - 59s 236ms/step - loss: 0.4801 - accuracy: 0.7696 - val_loss: 0.5208 - val_accuracy: 0.7350
Epoch 8/50
250/250 [==============================] - 59s 235ms/step - loss: 0.4563 - accuracy: 0.7881 - val_loss: 0.5156 - val_accuracy: 0.7435
Epoch 9/50
250/250 [==============================] - 59s 236ms/step - loss: 0.4320 - accuracy: 0.8004 - val_loss: 0.5037 - val_accuracy: 0.7430
Epoch 10/50
250/250 [==============================] - 59s 236ms/step - loss: 0.4008 - accuracy: 0.8227 - val_loss: 0.5142 - val_accuracy: 0.7455
Epoch 11/50
250/250 [==============================] - 59s 235ms/step - loss: 0.3821 - accuracy: 0.8295 - val_loss: 0.5015 - val_accuracy: 0.7505
Epoch 12/50
250/250 [==============================] - 58s 231ms/step - loss: 0.3567 - accuracy: 0.8441 - val_loss: 0.5127 - val_accuracy: 0.7410
Epoch 13/50
250/250 [==============================] - 55s 220ms/step - loss: 0.3170 - accuracy: 0.8686 - val_loss: 0.5097 - val_accuracy: 0.7475
Epoch 14/50
250/250 [==============================] - 55s 220ms/step - loss: 0.2894 - accuracy: 0.8820 - val_loss: 0.5437 - val_accuracy: 0.7420
Epoch 15/50
250/250 [==============================] - 55s 221ms/step - loss: 0.2579 - accuracy: 0.8959 - val_loss: 0.5324 - val_accuracy: 0.7510
Epoch 16/50
250/250 [==============================] - 55s 220ms/step - loss: 0.2352 - accuracy: 0.9080 - val_loss: 0.5335 - val_accuracy: 0.7585
Epoch 17/50
250/250 [==============================] - 55s 220ms/step - loss: 0.2013 - accuracy: 0.9239 - val_loss: 0.5599 - val_accuracy: 0.7600
Epoch 18/50
250/250 [==============================] - 55s 220ms/step - loss: 0.1668 - accuracy: 0.9421 - val_loss: 0.6167 - val_accuracy: 0.7480
Epoch 19/50
250/250 [==============================] - 55s 220ms/step - loss: 0.1421 - accuracy: 0.9529 - val_loss: 0.6177 - val_accuracy: 0.7505
Epoch 20/50
250/250 [==============================] - 55s 220ms/step - loss: 0.1121 - accuracy: 0.9663 - val_loss: 0.6469 - val_accuracy: 0.7495
Epoch 21/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0931 - accuracy: 0.9760 - val_loss: 0.7084 - val_accuracy: 0.7355
Epoch 22/50
250/250 [==============================] - 55s 219ms/step - loss: 0.0743 - accuracy: 0.9819 - val_loss: 0.7988 - val_accuracy: 0.7410
Epoch 23/50
250/250 [==============================] - 55s 222ms/step - loss: 0.0564 - accuracy: 0.9885 - val_loss: 0.7253 - val_accuracy: 0.7510
Epoch 24/50
250/250 [==============================] - 55s 219ms/step - loss: 0.0409 - accuracy: 0.9946 - val_loss: 0.8241 - val_accuracy: 0.7390
Epoch 25/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0332 - accuracy: 0.9962 - val_loss: 0.8333 - val_accuracy: 0.7360
Epoch 26/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0283 - accuracy: 0.9975 - val_loss: 0.8197 - val_accuracy: 0.7455
Epoch 27/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0152 - accuracy: 0.9996 - val_loss: 0.8662 - val_accuracy: 0.7515
Epoch 28/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0121 - accuracy: 0.9995 - val_loss: 0.8898 - val_accuracy: 0.7465
Epoch 29/50
250/250 [==============================] - 55s 221ms/step - loss: 0.0097 - accuracy: 1.0000 - val_loss: 0.9075 - val_accuracy: 0.7510
Epoch 30/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0077 - accuracy: 0.9999 - val_loss: 0.9398 - val_accuracy: 0.7460
Epoch 31/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0064 - accuracy: 1.0000 - val_loss: 0.9528 - val_accuracy: 0.7485
Epoch 32/50
250/250 [==============================] - 55s 222ms/step - loss: 0.0057 - accuracy: 1.0000 - val_loss: 0.9721 - val_accuracy: 0.7450
Epoch 33/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0049 - accuracy: 1.0000 - val_loss: 0.9919 - val_accuracy: 0.7450
Epoch 34/50
250/250 [==============================] - 55s 221ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 1.0295 - val_accuracy: 0.7500
Epoch 35/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 1.0230 - val_accuracy: 0.7445
Epoch 36/50
250/250 [==============================] - 55s 220ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 1.0350 - val_accuracy: 0.7475
Epoch 37/50
250/250 [==============================] - 55s 221ms/step - loss: 0.0032 - accuracy: 1.0000 - val_loss: 1.0464 - val_accuracy: 0.7465
Epoch 38/50
250/250 [==============================] - 55s 221ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 1.0637 - val_accuracy: 0.7470
Epoch 39/50
250/250 [==============================] - 56s 224ms/step - loss: 0.0027 - accuracy: 1.0000 - val_loss: 1.0697 - val_accuracy: 0.7455
Epoch 40/50
250/250 [==============================] - 59s 237ms/step - loss: 0.0025 - accuracy: 1.0000 - val_loss: 1.0838 - val_accuracy: 0.7485
Epoch 41/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0023 - accuracy: 1.0000 - val_loss: 1.0949 - val_accuracy: 0.7475
Epoch 42/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 1.1152 - val_accuracy: 0.7480
Epoch 43/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0020 - accuracy: 1.0000 - val_loss: 1.1141 - val_accuracy: 0.7470
Epoch 44/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0019 - accuracy: 1.0000 - val_loss: 1.1215 - val_accuracy: 0.7470
Epoch 45/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0018 - accuracy: 1.0000 - val_loss: 1.1323 - val_accuracy: 0.7465
Epoch 46/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0017 - accuracy: 1.0000 - val_loss: 1.1383 - val_accuracy: 0.7460
Epoch 47/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0016 - accuracy: 1.0000 - val_loss: 1.1448 - val_accuracy: 0.7480
Epoch 48/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0015 - accuracy: 1.0000 - val_loss: 1.1610 - val_accuracy: 0.7465
Epoch 49/50
250/250 [==============================] - 59s 235ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 1.1597 - val_accuracy: 0.7445
Epoch 50/50
250/250 [==============================] - 57s 228ms/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 1.1680 - val_accuracy: 0.7470

Evaluate model

157/157 [==============================] - 7s 44ms/step - loss: 1.2278 - accuracy: 0.7518
Accuracy = 75.180
```

Curvas de aprendizaje

In [ ]:
```python
learning_curves(history)
```



Guardar modelo

```
In [ ]: model.save(r"export_model\vgg3_augmentation_model.h5")
```

Conclusiones

A continuación se muestra una tabla comparativa de los distintos modelos probados durante este proyecto.

Modelo	Tiempo	Validation Accuracy	Accuracy
VGG Base	27:15	69.88 %	69.88 %
VGG-2	41:56	72.40 %	74.44 %
VGG-3	53:39	78.90 %	77.32 %
VGG-2 Dropout	47:42	73.65 %	73.48 %
VGG-3 Augmentation	47:30	74.70 %	75.18 %

- Se puede observar que el modelo que presenta mayor rendimiendo es el modelo VGG de tres bloques, por otra parte, es el que tardó más en entrenarse.
- Tanto el modelo VGG-3 y VGG-3 Augmentation, tienen el mínimo de error con respecto al set de validación, esto puede deberse a que la aumentación de datos no fue lo suficientemente variada, y de ahí que los comportamientos y resultados sean similares.
- El uso del dropout como medida de regularización no presentó una mejoría con respecto a la red VGG-2 sin dropout, esto puede someterse a más pruebas variando el porcentaje de desactivación de neuronas.

El uso de modelos VGG para la clasificación de imágenes de gran escala, son una herramienta bastante efectiva para muchos casos prácticos, sin embargo, el costo computacional es bastante algo, razón por la cual no pudo llevarse a cabo mayor variedad de modelos con su respectiva optimización de parámetros.