Archivos de la clase

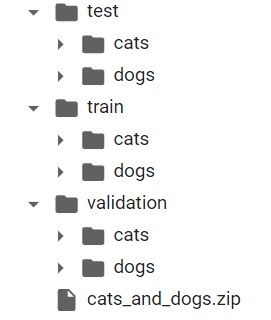
perros-vs-gatos.ipynb

https://colab.research.google.com/drive/1u0na7ep-IjFSkDZ-IfkLjdW1AfXiiiNP#scrollTo=mBIO0BVuDNaB

Subir datos

Carga el zip y luego

!unzip cats\_and\_dogs.zip



Perros vs. gatos | Kaggle

<https://www.kaggle.com/alarcon7a/perros-vs-gatos>

**CAMBIE RUNTIME A GTU**

import tensorflow as tf

from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

from  tensorflow.keras import models, optimizers, regularizers

**# Cada imagen de 150x150 a colores 150,150,3**

# Filtros 32 – 64 – 128 – 128 – 512

lote = 32

**MODELO**

model = models.Sequential()

model.add(Conv2D(**l6te**, (3,3), activation= 'relu', input\_shape=(150,150,3)))

model.add(MaxPooling2D((2,2))) **# Reducir ruido**

model.add(Conv2D(**lote\*2**, (3,3), activation = 'relu'))

model.add(MaxPooling2D((2,2)))

model.add(Conv2D(**lote\*4**, (3,3), activation = 'relu'))

model.add(MaxPooling2D((2,2)))

model.add(Conv2D(**lote\*4**, (3,3), activation = 'relu'))

model.add(MaxPooling2D((2,2)))

model.add(Flatten()) **#convertir matriz a vector**

model.add(Dropout(0.5)) **#reducir overfitting**

model.add(Dense(**lote\*16**, activation='relu'))

**# 1 filtro, salida, sigmoid, es gato o es perro, 2 valores, binary**

model.add(Dense(1, activation='sigmoid'))

model.summary()

**DATA AUGMENTATION**

train\_datagen = ImageDataGenerator(

**rescale=1./255**,

    rotation\_range=40,

    width\_shift\_range=0.2,

    height\_shift\_range=0.2,

    shear\_range=0.2,

    zoom\_range=0.2,

    horizontal\_flip=True

    )

test\_datagen = ImageDataGenerator(**rescale=1./255**)

train\_generator = train\_datagen.flow\_from\_directory('train',

                                 target\_size=(150,150),

                                 batch\_size= **lote**,

                                 class\_mode='binary' **#Gato o Perro**

                                 )

validation\_generator = test\_datagen.flow\_from\_directory('validation',

                                 target\_size=(150,150),

                                 batch\_size= **lote**,

                                 class\_mode='binary' **#Gato o Perro**

                                 )

**CALLBACK**

checkpoint = ModelCheckpoint('modelo.hdf5',

monitor='val\_accuracy',

verbose= 1,

save\_best\_only=True)

**COMPILANDO**

model.compile(loss='binary\_crossentropy',

optimizer =optimizers.Adam(),

              metrics=['accuracy'])

**ENTRENANDO**

hist = model.fit(train\_generator, steps\_per\_epoch=2000//32,

                epochs=100,

                validation\_data=validation\_generator,

                validation\_steps= 1000//32,

                callbacks=[checkpoint])

**RESULTADOS**

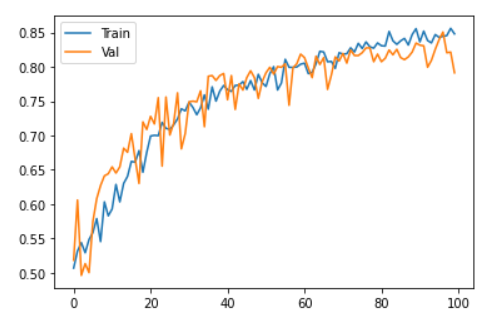
import matplotlib.pyplot as plt

plt.plot(hist.history['accuracy'], label = 'Train')

plt.plot(hist.history['val\_accuracy'], label = 'Val')

plt.legend()

plt.show()



test\_generator = test\_datagen.flow\_from\_directory('test',

                                 target\_size=(150,150),

                                 batch\_size= **lote**,

                                 class\_mode='binary'

                                 )

model2 = model

model2.load\_weights('./modelo.hdf5')

model2.evaluate(test\_generator)

32/32 [==============================] - 4s 111ms/step - loss: 0.4444 - accuracy: 0.8140

[0.444, 0.81400]