```
!pip install tensorflow opencv-python

Requirement already satisfied: tensorflow in
/usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: opencv-python in
/usr/local/lib/python3.10/dist-packages (4.8.0.76)
Requirement already satisfied: absl-py>=1.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.5.4)
Requirement already satisfied: google-pasta>=0.1.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!
=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (3.20.3)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0
in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in
```

```
/usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0-
>tensorflow) (0.43.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (2.27.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (3.6)
Requirement already satisfied: requests<3,>=2.21.0 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (2.31.0)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0
in /usr/local/lib/python3.10/dist-packages (from
tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: werkzeug>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15-
>tensorflow) (3.0.3)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (5.3.3)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
/usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3-
>tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/usr/local/lib/python3.10/dist-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: charset-normalizer<4,>=2 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (2.0.7)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0-
>tensorboard<2.16,>=2.15->tensorflow) (2024.6.2)
Requirement already satisfied: MarkupSafe>=2.1.1 in
/usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1-
>tensorboard<2.16,>=2.15->tensorflow) (2.1.5)
```

```
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in
/usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1-
>google-auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.6.0)
Requirement already satisfied: oauthlib>=3.0.0 in
/usr/local/lib/python3.10/dist-packages (from requests-
oauthlib>=0.7.0->google-auth-oauthlib<2,>=0.5-
>tensorboard<2.16,>=2.15->tensorflow) (3.2.2)
```

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

```
Mounted at /content/drive
```

```
import os
import cv2
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

```
import os
import cv2
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the directories for input images, resized images, and
augmented images
input_dir = '/content/drive/MyDrive/Injured dog'
resized_dir = '/content/drive/MyDrive/resized_images'
augmented_dir = '/content/drive/MyDrive/augmented_images'

# Ensure the directories exist
os.makedirs(resized_dir, exist_ok=True)
os.makedirs(augmented_dir, exist_ok=True)

# Resize images to (200, 200)
for filename in os.listdir(input_dir):
    if filename.endswith(('.png', '.jpg', '.jpeg')):
        img_path = os.path.join(input_dir, filename)
        img = cv2.imread(img_path)

        if img is None:
            print(f"Error loading image {img_path}")
            continue

        resized_img = cv2.resize(img, (256, 256))
        cv2.imwrite(os.path.join(resized_dir, filename), resized_img)

# Initialize ImageDataGenerator for data augmentation
datagen = ImageDataGenerator(
    rotation_range=15,  # Reduced rotation range to keep wounds
recognizable
```

```
    width_shift_range=0.1,  # Reduced shifts to prevent wounds from
being moved out of view
    height_shift_range=0.1,
    shear_range=0.1,  # Reduced shear to prevent excessive distortion
    zoom_range=0.1,  # Reduced zoom to keep the context of the wound
    horizontal_flip=True,  # Horizontal flip can be useful
    vertical_flip=False,  # Vertical flip is usually not relevant for
wounds
    fill_mode='nearest',
    brightness_range=[0.8, 1.2],  # Vary brightness to simulate
different lighting conditions
    channel_shift_range=0.1  # Slight changes in color channels to
simulate lighting changes
)


# Load resized images
resized_images = []
for filename in os.listdir(resized_dir):
    if filename.endswith(('.png', '.jpg', '.jpeg')):
        img_path = os.path.join(resized_dir, filename)
        img = cv2.imread(img_path)

        if img is None:
            print(f"Error loading image {img_path}")
            continue

        resized_images.append(img)

resized_images = np.array(resized_images)

# Number of augmented images to generate
num_augmented_images = 12500  # Change this value to generate a
different number of images

# Apply data augmentation
i = 0
for batch in datagen.flow(resized_images, batch_size=1,
save_to_dir=augmented_dir, save_prefix='aug', save_format='jpg'):
    i += 1
    if i >= num_augmented_images:
        break

print(f'Resizing and data augmentation completed.
{num_augmented_images} images saved in {augmented_dir}.')
```

```
Error loading image /content/drive/MyDrive/Injured dog/frame_4103.png
Error loading image /content/drive/MyDrive/Injured dog/frame_4303.png
Error loading image /content/drive/MyDrive/Injured dog/frame_4503.png
```

```
Resizing and data augmentation completed. 12500 images saved in
/content/drive/MyDrive/augmented_images.
```

```
!mkdir -p ~/.kaggle
!cp kaggle.json ~/.kaggle/
```

```
cp: cannot stat 'kaggle.json': No such file or directory
```

```
!kaggle datasets download -d salader/dogs-vs-cats
```

```
Dataset URL: https://www.kaggle.com/datasets/salader/dogs-vs-cats
License(s): unknown
Downloading dogs-vs-cats.zip to /content
 99% 1.05G/1.06G [00:04<00:00, 299MB/s]
100% 1.06G/1.06G [00:04<00:00, 253MB/s]
```

```
import zipfile
zip_ref = zipfile.ZipFile("/content/dogs-vs-cats.zip","r")
zip_ref.extractall("/content")
zip_ref.close()
```

```
# import shutil

# # Path to the directory you want to remove
# directory = '/content/train/injured_dogs'

# # Remove the directory and its contents
# shutil.rmtree(directory)

# # Verify the directory has been removed
# print(f'Directory removed: {not os.path.exists(directory)}')
```

```
import os
import shutil
import random

# Define paths
augmented_dir = '/content/drive/MyDrive/augmented_images'
train_dir = '/content/train/injured_dogs'
test_dir = '/content/test/injured_dogs'

# Ensure the directories exist
os.makedirs(train_dir, exist_ok=True)
os.makedirs(test_dir, exist_ok=True)

# Get a list of all images in the augmented directory
augmented_images = [f for f in os.listdir(augmented_dir) if
os.path.isfile(os.path.join(augmented_dir, f))]

# Check if there are enough images
if len(augmented_images) < 12491:
```

```
    raise ValueError("Not enough images in the augmented directory to
perform the split.")

# Shuffle the images
random.shuffle(augmented_images)

# Split the images
train_images = augmented_images[:10000]
test_images = augmented_images[10000:12491]

# Function to move files
def move_files(files, src_dir, dest_dir, prefix):
    for i, img in enumerate(files):
        new_name = f"{prefix}_{i+1}.jpg"
        shutil.move(os.path.join(src_dir, img), os.path.join(dest_dir,
new_name))

# Move the images to the respective directories
move_files(train_images, augmented_dir, train_dir,
'injured_dog_train')
move_files(test_images, augmented_dir, test_dir, 'injured_dog_test')

print('10,000 images of injured dogs moved to train directory and
2,491 images of injured dogs moved to test directory.')
```

```
10,000 images of injured dogs moved to train directory and 2,491
images of injured dogs moved to test directory.
```

```
import os
import shutil

# Function to rename and move a directory
def rename_and_move(original_dir, new_dir_name, destination_dir):
    new_dir_path = os.path.join(os.path.dirname(original_dir),
new_dir_name)
    os.rename(original_dir, new_dir_path)
    os.makedirs(destination_dir, exist_ok=True)
    shutil.move(new_dir_path, destination_dir)
    print(f'Directory renamed to "{new_dir_name}" and moved to
"{destination_dir}"')

# Paths for the train directory
train_original_dir = '/content/dogs_vs_cats/train/dogs'
train_new_dir_name = 'healthy_dogs'
train_destination_dir = '/content/train'

# Paths for the test directory
test_original_dir = '/content/dogs_vs_cats/test/dogs'
test_new_dir_name = 'healthy_dogs'
test_destination_dir = '/content/test'
```

```
# Rename and move train directory
rename_and_move(train_original_dir, train_new_dir_name,
train_destination_dir)

# Rename and move test directory
rename_and_move(test_original_dir, test_new_dir_name,
test_destination_dir)
```

```
Directory renamed to "healthy_dogs" and moved to "/content/train"
Directory renamed to "healthy_dogs" and moved to "/content/test"
```

```
# Define paths for the folders to be removed
folders_to_remove = [
    '/content/test/cats',
    '/content/test/dogs',
    '/content/train/cats',
    '/content/train/dogs',
]

# Remove the folders
for folder in folders_to_remove:
    if os.path.exists(folder):
        shutil.rmtree(folder)
        print(f'Removed {folder}')
    else:
        print(f'{folder} does not exist')

print('Specified folders removed from train and test directories.')
```

```
Removed /content/test/cats
Removed /content/test/dogs
Removed /content/train/cats
Removed /content/train/dogs
Specified folders removed from train and test directories.
```

```
import os

def delete_files_from_folder(folder_path, num_files_to_delete):
    # List all files in the directory
    files = [f for f in os.listdir(folder_path) if
os.path.isfile(os.path.join(folder_path, f))]

    # Check if there are enough files to delete
    if len(files) < num_files_to_delete:
        print(f"Not enough files to delete. The folder contains only
{len(files)} files.")
        return
```

```
    # Delete the specified number of files
    for file in files[:num_files_to_delete]:
        file_path = os.path.join(folder_path, file)
        os.remove(file_path)
        print(f"Deleted file: {file_path}")

# Specify the folder path and number of files to delete
folder_path = '/content/test/healthy_dogs'
num_files_to_delete = 9      # To make this equal in the test
dataset...

# Delete the files
delete_files_from_folder(folder_path, num_files_to_delete)
```

```
Deleted file: /content/test/healthy_dogs/dog.4437.jpg
Deleted file: /content/test/healthy_dogs/dog.7815.jpg
Deleted file: /content/test/healthy_dogs/dog.7683.jpg
Deleted file: /content/test/healthy_dogs/dog.1345.jpg
Deleted file: /content/test/healthy_dogs/dog.9851.jpg
Deleted file: /content/test/healthy_dogs/dog.10129.jpg
Deleted file: /content/test/healthy_dogs/dog.8767.jpg
Deleted file: /content/test/healthy_dogs/dog.7020.jpg
Deleted file: /content/test/healthy_dogs/dog.3948.jpg
```

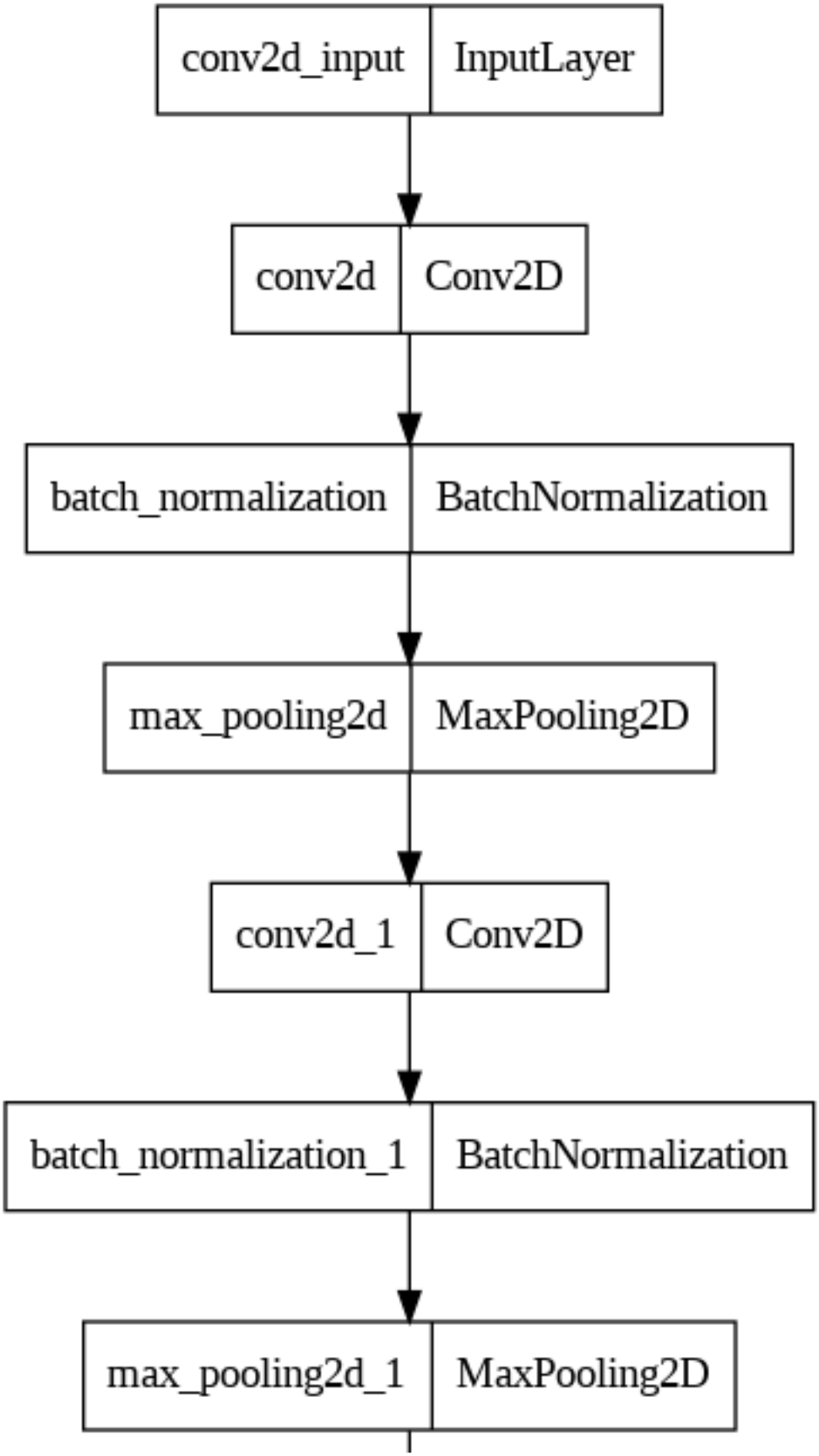
```
import os

def count_items_in_folder(folder_path):
    num_files = 0
    num_dirs = 0

    for root, dirs, files in os.walk(folder_path):
        num_files += len(files)
        num_dirs += len(dirs)

    return num_files, num_dirs

# Specify the list of folder paths
folder_paths = [
    '/content/train/healthy_dogs',  # Change this to your folder path
    '/content/train/injured_dogs',  # Add more paths as needed
    '/content/test/healthy_dogs',
    '/content/test/injured_dogs'
]

# Iterate through each folder and get the count of files and
directories
for folder_path in folder_paths:
    num_files, num_dirs = count_items_in_folder(folder_path)
    print(f"Folder: {folder_path}")
    print(f"Number of files: {num_files}")
```

```
    print(f"Number of directories: {num_dirs}")
    print('---')


Folder: /content/train/healthy_dogs
Number of files: 10000
Number of directories: 0
---
Folder: /content/train/injured_dogs
Number of files: 10000
Number of directories: 0
---
Folder: /content/test/healthy_dogs
Number of files: 2491
Number of directories: 0
---
Folder: /content/test/injured_dogs
Number of files: 2491
Number of directories: 0
---

import tensorflow as tf
from tensorflow import keras
from keras import Sequential
from keras.layers import
Dense,Conv2D,MaxPooling2D,Flatten,BatchNormalization,Dropout,
GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2

# generators   (To take all the images as input )

train_dataset = keras.utils.image_dataset_from_directory(
    directory ="/content/train",
    labels="inferred",
    label_mode="int",
    batch_size=32,
    image_size=(256, 256),
)


validation_dataset = keras.utils.image_dataset_from_directory(
    directory ="/content/test",
    labels="inferred",
    label_mode="int",
    batch_size=32,
    image_size=(256, 256),
)

Found 20000 files belonging to 2 classes.
Found 4982 files belonging to 2 classes.
```

```
# Normalization
def process(image,label):
  image =  tf.cast(image/255  , tf.float32)
  return image, label

train_dataset = train_dataset.map(process)
validation_dataset = validation_dataset.map(process)

# # Creating a CNN Model

# model = Sequential()

# # First Convolutional Layer
# model.add(Conv2D(32, kernel_size=(3, 3), padding="valid",
activation="relu", input_shape=(256, 256, 3),
kernel_regularizer=l2(0.001)))
# model.add(BatchNormalization())
# model.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding="valid"))

# # Second Convolutional Layer
# model.add(Conv2D(64, kernel_size=(3, 3), padding="valid",
activation="relu", kernel_regularizer=l2(0.001)))
# model.add(BatchNormalization())
# model.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding="valid"))

# # Third Convolutional Layer
# model.add(Conv2D(128, kernel_size=(3, 3), padding="valid",
activation="relu", kernel_regularizer=l2(0.001)))
# model.add(BatchNormalization())
# model.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding="valid"))

# # Flattening
# model.add(Flatten())

# # Fully Connected Layers
# model.add(Dense(128, activation="relu",
kernel_regularizer=l2(0.001)))
# model.add(Dropout(0.3))  # Increased dropout rate
# model.add(Dense(64, activation="relu",
kernel_regularizer=l2(0.001)))
# model.add(Dropout(0.3))  # Increased dropout rate

# # Output Layer
# model.add(Dense(1, activation="sigmoid"))

# # Compile the model
# model.compile(optimizer="adam", loss="binary_crossentropy",
```

```
metrics=["accuracy"])

#
------------------------------------------------------------------------
------------------------------------------------------------------

# model = Sequential()

# # First Convolutional Layer
# model.add(Conv2D(32, kernel_size=(3, 3), padding="valid",
activation="relu", input_shape=(256, 256, 3),
kernel_regularizer=l2(0.001)))
# model.add(BatchNormalization())
# model.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding="valid"))

# # Second Convolutional Layer
# model.add(Conv2D(64, kernel_size=(3, 3), padding="valid",
activation="relu", kernel_regularizer=l2(0.001)))
# model.add(BatchNormalization())
# model.add(MaxPooling2D(pool_size=(2, 2), strides=2,
padding="valid"))

# # Flattening
# model.add(Flatten())

# # Fully Connected Layers
# model.add(Dense(64, activation="relu",
kernel_regularizer=l2(0.001)))
# model.add(Dropout(0.3))  # Increased dropout rate

# # Output Layer
# model.add(Dense(1, activation="sigmoid"))

# # Compile the model
# model.compile(optimizer="adam", loss="binary_crossentropy",
metrics=["accuracy"])



#
========================================================================
==========================================

model = Sequential()

# First Convolutional Layer
model.add(Conv2D(32, kernel_size=(3, 3), padding="valid",
activation="relu", input_shape=(256, 256, 3),
kernel_regularizer=l2(0.001)))
```

```
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding="valid"))

# Second Convolutional Layer
model.add(Conv2D(64, kernel_size=(3, 3), padding="valid",
activation="relu", kernel_regularizer=l2(0.001)))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2), strides=2, padding="valid"))

# Global Average Pooling Layer
model.add(GlobalAveragePooling2D())

# Fully Connected Layers
model.add(Dense(64, activation="relu", kernel_regularizer=l2(0.001)))
model.add(Dropout(0.3))  # Increased dropout rate

# Output Layer
model.add(Dense(1, activation="sigmoid"))

# Compile the model
model.compile(optimizer=Adam(), loss="binary_crossentropy",
metrics=["accuracy"])

from keras.utils import plot_model
plot_model(model)
```


conv2d_input
InputLayer
conv2d
Conv2D
batch_normalization
BatchNormalization
max_pooling2d
MaxPooling2D
conv2d_1
Conv2D
batch_normalization_1
BatchNormalization
max_pooling2d_1
MaxPooling2D

```
model.summary()
```

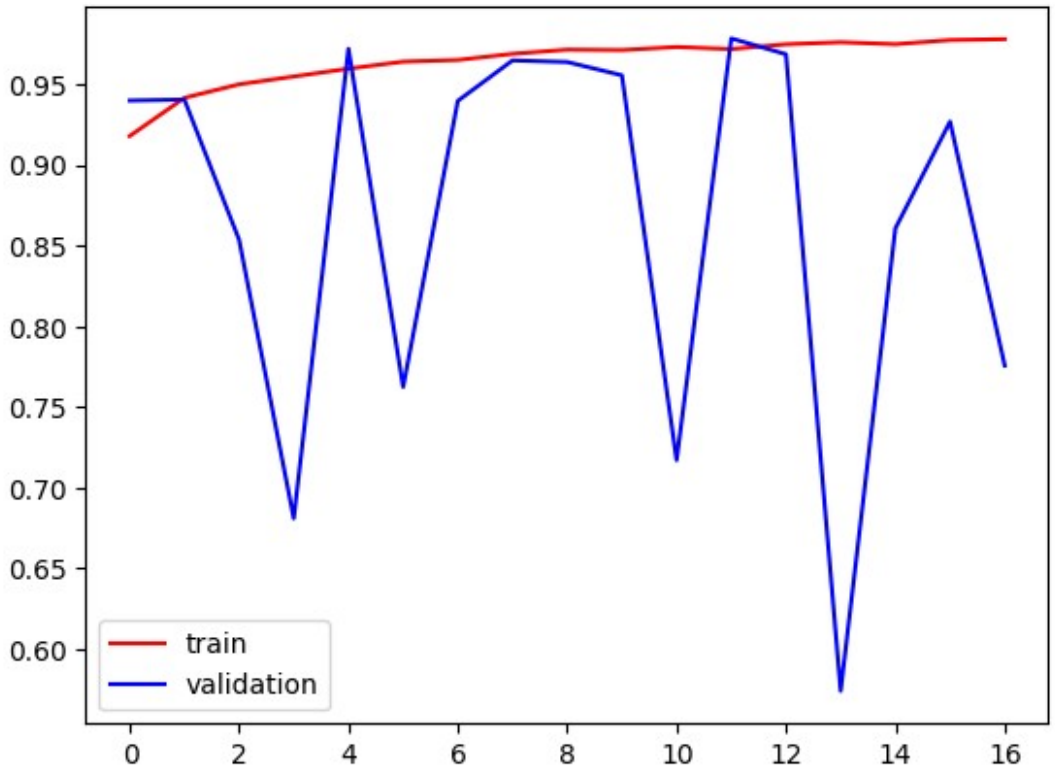
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 254, 254, 32)      896

 batch_normalization (Batch  (None, 254, 254, 32)      128
 Normalization)

 max_pooling2d (MaxPooling2  (None, 127, 127, 32)      0
 D)

 conv2d_1 (Conv2D)           (None, 125, 125, 64)      18496

 batch_normalization_1 (Bat  (None, 125, 125, 64)      256
 chNormalization)

 max_pooling2d_1 (MaxPoolin  (None, 62, 62, 64)        0
 g2D)

 global_average_pooling2d (  (None, 64)                0
 GlobalAveragePooling2D)

 dense (Dense)               (None, 64)                4160

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 1)                 65

=================================================================
Total params: 24001 (93.75 KB)
Trainable params: 23809 (93.00 KB)
Non-trainable params: 192 (768.00 Byte)
_________________________________________________________________
```

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

from tensorflow.keras.callbacks import EarlyStopping

early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)

history = model.fit(
    train_dataset,
    epochs=50,
    validation_data=validation_dataset,
```

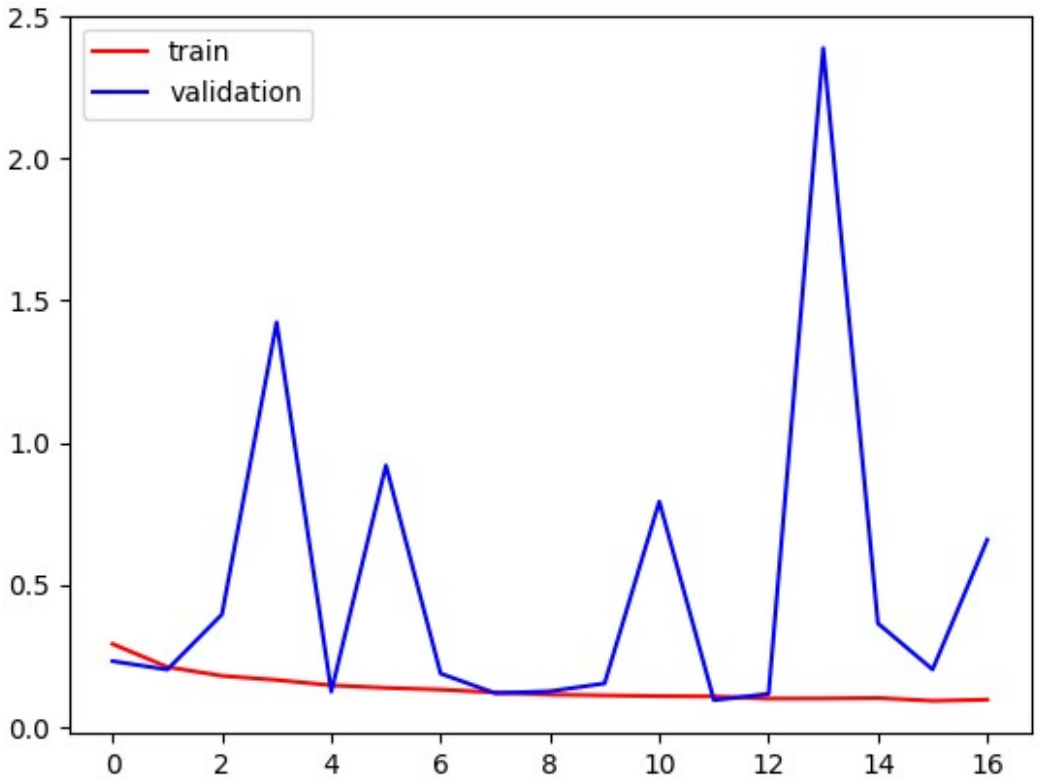
```
    callbacks=[early_stopping]
)

Epoch 1/50
625/625 [==============================] - 71s 92ms/step - loss:
0.2916 - accuracy: 0.9179 - val_loss: 0.2310 - val_accuracy: 0.9400
Epoch 2/50
625/625 [==============================] - 52s 83ms/step - loss:
0.2106 - accuracy: 0.9416 - val_loss: 0.2015 - val_accuracy: 0.9406
Epoch 3/50
625/625 [==============================] - 55s 86ms/step - loss:
0.1794 - accuracy: 0.9500 - val_loss: 0.3959 - val_accuracy: 0.8537
Epoch 4/50
625/625 [==============================] - 58s 92ms/step - loss:
0.1649 - accuracy: 0.9549 - val_loss: 1.4240 - val_accuracy: 0.6809
Epoch 5/50
625/625 [==============================] - 57s 91ms/step - loss:
0.1460 - accuracy: 0.9597 - val_loss: 0.1237 - val_accuracy: 0.9721
Epoch 6/50
625/625 [==============================] - 53s 85ms/step - loss:
0.1371 - accuracy: 0.9641 - val_loss: 0.9202 - val_accuracy: 0.7623
Epoch 7/50
625/625 [==============================] - 53s 85ms/step - loss:
0.1310 - accuracy: 0.9651 - val_loss: 0.1874 - val_accuracy: 0.9398
Epoch 8/50
625/625 [==============================] - 57s 91ms/step - loss:
0.1209 - accuracy: 0.9689 - val_loss: 0.1182 - val_accuracy: 0.9649
Epoch 9/50
625/625 [==============================] - 57s 91ms/step - loss:
0.1136 - accuracy: 0.9715 - val_loss: 0.1250 - val_accuracy: 0.9639
Epoch 10/50
625/625 [==============================] - 58s 92ms/step - loss:
0.1105 - accuracy: 0.9713 - val_loss: 0.1529 - val_accuracy: 0.9556
Epoch 11/50
625/625 [==============================] - 54s 86ms/step - loss:
0.1080 - accuracy: 0.9732 - val_loss: 0.7930 - val_accuracy: 0.7170
Epoch 12/50
625/625 [==============================] - 52s 83ms/step - loss:
0.1070 - accuracy: 0.9719 - val_loss: 0.0937 - val_accuracy: 0.9785
Epoch 13/50
625/625 [==============================] - 58s 91ms/step - loss:
0.0993 - accuracy: 0.9749 - val_loss: 0.1160 - val_accuracy: 0.9687
Epoch 14/50
625/625 [==============================] - 57s 91ms/step - loss:
0.0997 - accuracy: 0.9761 - val_loss: 2.3890 - val_accuracy: 0.5739
Epoch 15/50
625/625 [==============================] - 53s 85ms/step - loss:
0.1014 - accuracy: 0.9750 - val_loss: 0.3642 - val_accuracy: 0.8607
Epoch 16/50
625/625 [==============================] - 54s 87ms/step - loss:
```

```
0.0914 - accuracy: 0.9775 - val_loss: 0.2019 - val_accuracy: 0.9269
Epoch 17/50
625/625 [==============================] - 53s 85ms/step - loss:
0.0954 - accuracy: 0.9780 - val_loss: 0.6586 - val_accuracy: 0.7756
```

```
import matplotlib.pyplot as plt

plt.plot(history.history["accuracy"],color="red",label="train")
plt.plot(history.history["val_accuracy"],color="blue",label="validatio
n")
plt.legend()
plt.show()
```




```
import matplotlib.pyplot as plt

plt.plot(history.history["loss"],color="red",label="train")
plt.plot(history.history["val_loss"],color="blue",label="validation")
plt.legend()
plt.show()
```




```
# testing model on unseen images
import cv2
test_img_1 = cv2.imread("/content/100.jpeg")
plt.imshow(test_img_1)


test_img_1.shape

test_img_1  = cv2.resize(test_img_1,(256,256))

test_input = test_img_1.reshape((1,256,256,3)) # b/c we have only one
image as input

model.predict(test_input)

test_img = cv2.imread("/content/dod.jpg")
plt.imshow(test_img)

test_img.shape

test_img  = cv2.resize(test_img,(256,256))

test_input = test_img.reshape((1,256,256,3)) # b/c we have only one
image as input

model.predict(test_input)
```

```python
# Load and preprocess new image
from tensorflow.keras.preprocessing import image

img_path = '/content/dod.jpg'
img = image.load_img(img_path, target_size=(256, 256))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0) / 255.0  # Scale the
image

# Predict
prediction = model.predict(img_array)
print(prediction)

# Convert to class label
class_label = (prediction > 0.5).astype(int)
print("Predicted class:", "Injured Dog" if class_label == 1 else
"Healthy Dog")
```