Small dataset with similar data input

```
import tensorflow as tf
from tensorflow.keras import Sequential,Model
from tensorflow.keras.layers import Dense,Dropout,Flatten,BatchNormalization
from tensorflow.keras.applications.vgg16 import VGG16,preprocess_input
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

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

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

```
img_width=50
img_height=50
batch_size=500
```

```
# Initializing training dataset
train_data_dir="/content/drive/My Drive/fruitvsvegetable/train"
datagen = ImageDataGenerator(rescale=1./255)
train_generator = datagen.flow_from_directory(directory=train_data_dir,
                                              target_size = (img_width, img_height),
                                              class_mode = 'binary',
                                              batch_size=batch_size)
```

```
Found 505 images belonging to 2 classes.
```

```
# Initializing validation dataset
validation_dir="/content/drive/My Drive/validation"
val_generator=datagen.flow_from_directory(validation_dir,target_size=(img_width,img_height),classes=['cat','dog'],batch_size
```

```
Found 40 images belonging to 2 classes.
```

```
# importing VGG16 model
vgg_arch=VGG16(input_shape=(img_width,img_height,3),weights="imagenet",include_top=False) #include_top= False represents tha
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering
58892288/58889256 [==============================] - 0s 0us/step
```

```
# This says that not to use dense values mentioned in VGG (Not to use VGG layers)
for layers in vgg_arch.layers:
  layers.trainable=False
```

```
# Training the model
model=Sequential()
model.add(vgg_arch)
model.add(Flatten())
model.add(Dense(128,activation='relu',))
model.add(Dropout(0.5))
model.add(BatchNormalization())
model.add(Dense(1,activation="sigmoid"))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Functional)           (None, 1, 1, 512)         14714688
```

```
_________________________________________________________________
flatten (Flatten)            (None, 512)               0         
_________________________________________________________________
dense (Dense)                (None, 128)               65664     
_________________________________________________________________
dropout (Dropout)            (None, 128)               0         
_________________________________________________________________
batch_normalization (BatchNo (None, 128)               512       
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 129       
=================================================================
Total params: 14,780,993
Trainable params: 66,049
Non-trainable params: 14,714,944
_________________________________________________________________
```

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

```
history=model.fit_generator(generator=train_generator, steps_per_epoch=len(train_generator), epochs = 10,
                            validation_data=val_generator, validation_steps=len(val_generator)
                            , verbose = 1)
```

```
WARNING:tensorflow:From <ipython-input-17-d0d9008c083e>:3: Model.fit_generator (from tensorflow.python.keras.engine.tr
Instructions for updating:
Please use Model.fit, which supports generators.
Epoch 1/10
2/2 [==============================] - 44s 22s/step - loss: 0.8472 - accuracy: 0.5129 - val_loss: 0.6922 - val_accurac
Epoch 2/10
2/2 [==============================] - 28s 14s/step - loss: 0.7611 - accuracy: 0.5545 - val_loss: 0.6802 - val_accurac
Epoch 3/10
2/2 [==============================] - 1s 631ms/step - loss: 0.7431 - accuracy: 0.5564 - val_loss: 0.6723 - val_accura
Epoch 4/10
2/2 [==============================] - 1s 625ms/step - loss: 0.6752 - accuracy: 0.6356 - val_loss: 0.6702 - val_accura
Epoch 5/10
2/2 [==============================] - 1s 635ms/step - loss: 0.6374 - accuracy: 0.6475 - val_loss: 0.6703 - val_accura
Epoch 6/10
2/2 [==============================] - 1s 630ms/step - loss: 0.6223 - accuracy: 0.7010 - val_loss: 0.6678 - val_accura
Epoch 7/10
2/2 [==============================] - 1s 634ms/step - loss: 0.6473 - accuracy: 0.6515 - val_loss: 0.6655 - val_accura
```

```
Epoch 8/10
2/2 [==============================] - 28s 14s/step - loss: 0.6338 - accuracy: 0.6851 - val_loss: 0.6656 - val_accurac
Epoch 9/10
2/2 [==============================] - 1s 638ms/step - loss: 0.5852 - accuracy: 0.7129 - val_loss: 0.6674 - val_accura
Epoch 10/10
2/2 [==============================] - 1s 641ms/step - loss: 0.5664 - accuracy: 0.7010 - val_loss: 0.6682 - val_accura
```

```
from tensorflow.keras.preprocessing import image



# Predicting the image
img=image.load_img("/content/256.jpg",target_size=(img_width,img_height))
plt.imshow(img)
img=image.img_to_array(img)
img=img/255.0
img = np.expand_dims(img, axis=0)
img_class = np.argmax(model.predict(img),axis=1)

if(model.predict(img)<=0.5):
  print('fruits')

else:
  print('vegetable')
```

```
fruits
```

```
# Predicting the image
img=image.load_img("/content/261.jpg",target_size=(img_width,img_height))
plt.imshow(img)
img=image.img_to_array(img)
img=img/255.0
img = np.expand_dims(img, axis=0)
img_class = np.argmax(model.predict(img),axis=1)

if(model.predict(img)<=0.5):
  print('fruits')

else:
  print('vegetable')
```

```
vegetable
```