Double-click (or enter) to edit

VGG 16 using te Keras applications

```
from keras.applications.vgg16 import VGG16
model = VGG16()
model.summary()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ━━━━━━━━━━━━━━━━━━━━ 7s 0us/step
Model: "vgg16"
```

```
Start coding or generate with AI.
```

```
Start coding or generate with AI.
```

```
from keras.preprocessing.image import load_img
```

```
from keras.applications.imagenet_utils import preprocess_input
image=load_img('phone.jpg',target_size=(224,224))
image=np.array(image)
image=image.reshape((1,image.shape[0],image.shape[1],image.shape[2]))
image=preprocess_input(image)
my_image=imread('phone.jpg')
imshow(my_image)
```

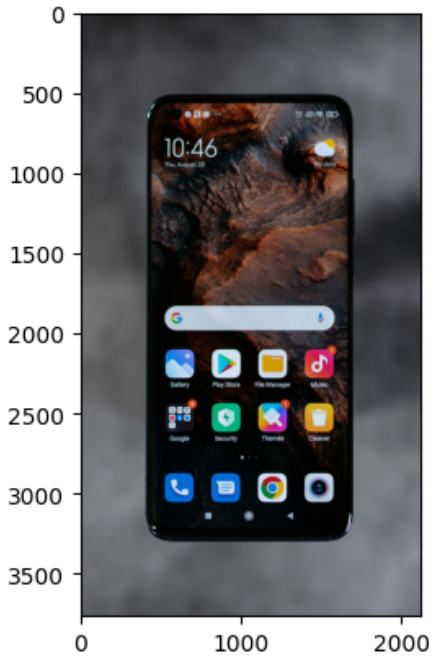
```
NameError                                 Traceback (most recent call last)
/tmp/ipython-input-2-3301202073.py in <cell line: 0>()
      1 from keras.applications.imagenet_utils import preprocess_input
----> 2 image=load_img('phone.jpg',target_size=(224,224))
      3 image=np.array(image)
      4 image=image.reshape((1,image.shape[0],image.shape[1],image.shape[2]))
      5 image=preprocess_input(image)

NameError: name 'load_img' is not defined
```

using Keras applications and VGG16 classification and prediciaction of image .

```
from keras.applications.vgg16 import VGG16
from keras.preprocessing.image import load_img
from keras.applications.imagenet_utils import preprocess_input
import numpy as np
from matplotlib.pyplot import imshow
```

```
from skimage.io import imread

# Load the VGG16 model
model = VGG16()

# Load and preprocess the image
# Make sure you have an image named 'phone.jpg' in your working directory or provide the correct path
image = load_img('phone.jpg', target_size=(224, 224))
image = np.array(image)
image = image.reshape((1, image.shape[0], image.shape[1], image.shape[2]))
image = preprocess_input(image)

# Display the image (optional)
my_image = imread('phone.jpg')
imshow(my_image)

# Now the 'image' variable contains the preprocessed image ready to be used as input for the VGG16 model.
```

```
<matplotlib.image.AxesImage at 0x7e652674ac10>
```



```
from keras.applications.vgg16 import preprocess_input,decode_predictions
predictions=model.predict(image)
label=decode_predictions(predictions)
label=label[0][0]
print('%s (%.2f%%)' % (label[1],label[2]*100))
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 597ms/step
iPod (68.25%)
```

implemation of VGG16 using tensorflow .

```
import tensorflow as tf
from tensorflow.keras import layers, models

def build_vgg(input_shape=(224,224,3), num_classes=10):
    model = models.Sequential()

    # Block 1
    model.add(layers.Conv2D(64, (3,3), activation='relu', padding='same', input_shape=input_shape))
    model.add(layers.Conv2D(64, (3,3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2,2)))

    # Block 2
    model.add(layers.Conv2D(128, (3,3), activation='relu', padding='same'))
    model.add(layers.Conv2D(128, (3,3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2,2)))

    # Block 3
    model.add(layers.Conv2D(256, (3,3), activation='relu', padding='same'))
    model.add(layers.Conv2D(256, (3,3), activation='relu', padding='same'))
    model.add(layers.Conv2D(256, (3,3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2,2)))

    # Block 4
    model.add(layers.Conv2D(512, (3,3), activation='relu', padding='same'))
```

```
    model.add(layers.Conv2D(512, (3,3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3,3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2,2)))

    # Block 5
    model.add(layers.Conv2D(512, (3,3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3,3), activation='relu', padding='same'))
    model.add(layers.Conv2D(512, (3,3), activation='relu', padding='same'))
    model.add(layers.MaxPooling2D((2,2)))

    # Fully connected layers
    model.add(layers.Flatten())
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(4096, activation='relu'))
    model.add(layers.Dropout(0.5))
    model.add(layers.Dense(num_classes, activation='softmax'))

    return model

# Create the model
model = build_vgg(input_shape=(224,224,3), num_classes=10)

# Compile (still no data needed)
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Show summary
model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_1 (Conv2D)	(None, 224, 224, 64)	36,928

Resnet -50 using Keras

```
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model

# Load the ResNet50 model without the top classification layer
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Freeze the base model (so we don't train these weights)
base_model.trainable = False

# Add custom layers on top
x = base_model.output
x = GlobalAveragePooling2D()(x)    # Global average pooling instead of flatten
x = Dense(1024, activation='relu')(x)
predictions = Dense(3, activation='softmax')(x)   # Assuming 3 classes

# Create the final model
model = Model(inputs=base_model.input, outputs=predictions)

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

# Print summary
model.summary()
```

dropout_1 (Dropout)	(None, 4096)	0

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 ━━━━━━━━━━━━━━━━━━━━ 1s 0us/step
```

Model: "functional_24"

Layer (type)	Output Shape	Param #	Connected to
input_layer_3 (InputLayer)	(None, 224, 224, 3)	0	-
conv1_pad (ZeroPadding2D)	(None, 230, 230, 3)	0	input_layer_3[0]…
conv1_conv (Conv2D)	(None, 112, 112, 64)	9,472	conv1_pad[0][0]
conv1_bn (BatchNormalizatio…	(None, 112, 112, 64)	256	conv1_conv[0][0]
conv1_relu (Activation)	(None, 112, 112, 64)	0	conv1_bn[0][0]
pool1_pad (ZeroPadding2D)	(None, 114, 114, 64)	0	conv1_relu[0][0]
pool1_pool (MaxPooling2D)	(None, 56, 56, 64)	0	pool1_pad[0][0]
conv2_block1_1_conv (Conv2D)	(None, 56, 56, 64)	4,160	pool1_pool[0][0]
conv2_block1_1_bn (BatchNormalizatio…	(None, 56, 56, 64)	256	conv2_block1_1_c…
conv2_block1_1_relu	(None, 56, 56,	0	conv2_block1_1_b…

Total params: 134,301,514 (512.32 MB)
Trainable params: 134,301,514 (512.32 MB)
Non-trainable params: 0 (0.00 B)