

### 1) algorithm-related

- 1-1: load pre-trained model (Ex: VGG16)
- 1-2: add new classifier layer
- 1-3: define the loaded pre-trained model

```
In [7]: #load model
from keras.applications.vgg16 import VGG16
from keras.models import Model
from keras.layers import Dense
from keras.layers import Flatten

In [8]: # Define the model
model = VGG16(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

In [13]: # Add new classifier layer
flat1 = Flatten()(model.layers[-1].output)
class1 = Dense(256, activation='relu')(flat1)
output = Dense(1, activation='sigmoid')(class1)
```

### 2) image processing

- 2-1: load image
- 2-2: resize the image to fit model requirement
- 2-3: define image characteristics

```
In [3]: # Load data
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    width_shift_range=0.05,
    height_shift_range=0.15,
    rotation_range=10,
    zoom_range=0.1,
)
validation_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

In [4]: # train_generator = train_datagen.flow_from_directory(
    'D:/CRUIING ROI 101/train',
    target_size=(224,224),
    batch_size=32,
    class_mode='binary',
    #color_mode='grayscale',
    shuffle=True,
)
```

Found 3480 Images belonging to 2 classes.

### 3) model fitting and performance evaluation.

#### 3-1: model fitting

```
history = model.fit(train_generator, epochs=20, validation_data=validation_generator)

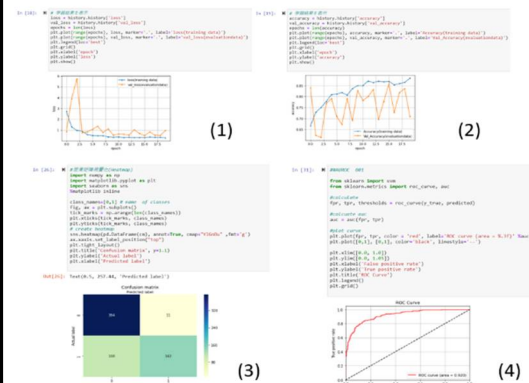
WARNING:tensorflow:sample_weight modes were coerced from:
...
['...']
WARNING:tensorflow:sample_weight modes were coerced from:
...
['...']

Train for 100 steps, validate for 7 steps
Epoch 1/20
100/100 [#####] 100s 11s/step - loss: 2.7333 - accuracy: 0.6664 - binary_crossentropy: 2.7385 -
precision: 0.5647 - recall: 0.5601 - val_loss: 0.8515 - val_accuracy: 0.8489 - val_binary_crossentropy: 0.8283 - val_precision:
0.8587 - val_recall: 0.8471
```

### 3) model fitting and performance evaluation.

#### 3-2: performance evaluation

- (1). LOSS curve (2) Accuracy curve (3) confusion matrix (4) AUROC



Model: "vgg16"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	(None, 224, 224, 3)	0
block1_conv1 (Conv2D)	(None, 224, 224, 64)	1792
block1_conv2 (Conv2D)	(None, 224, 224, 64)	36928
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	0
block2_conv1 (Conv2D)	(None, 112, 112, 128)	73856
block2_conv2 (Conv2D)	(None, 112, 112, 128)	147584
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	0
block3_conv1 (Conv2D)	(None, 56, 56, 256)	295168
block3_conv2 (Conv2D)	(None, 56, 56, 256)	590080
block3_conv3 (Conv2D)	(None, 56, 56, 256)	590080
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	0
block4_conv1 (Conv2D)	(None, 28, 28, 512)	1180160
block4_conv2 (Conv2D)	(None, 28, 28, 512)	2359808
block4_conv3 (Conv2D)	(None, 28, 28, 512)	2359808
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	0
block5_conv1 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv2 (Conv2D)	(None, 14, 14, 512)	2359808
block5_conv3 (Conv2D)	(None, 14, 14, 512)	2359808
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	0
Total params: 14,714,688		
Trainable params: 14,714,688		
Non-trainable params: 0		

**Figure S2.** Summary of VGG16 model developed in this study.

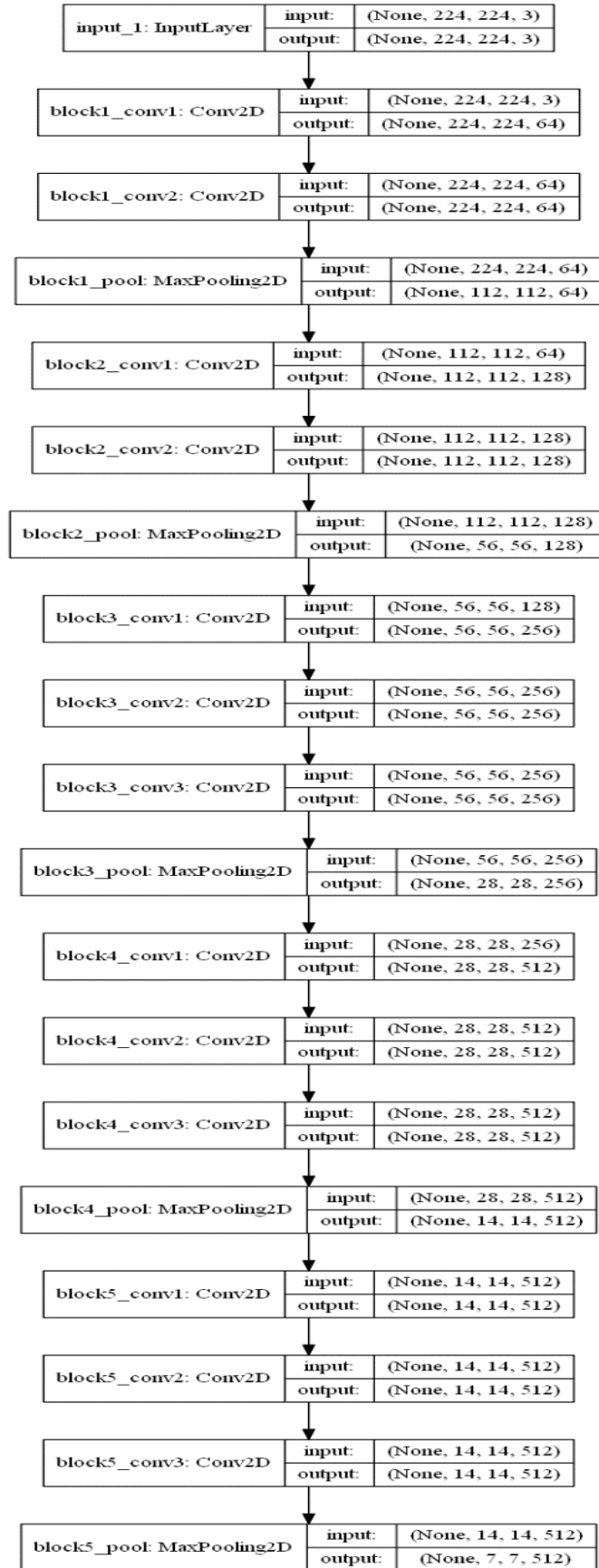


Figure S3. Model architecture of VGG16 model developed in this study.