Implement the VGG-19 architectureusing a deep learning library such as TensorFlow or PyTorch to build a sequential model

In [ ]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

The VGG-19 architecture consists of 16 convolutional layers and 3 fully connected layers, making a total of 19 layers.

The Input Layer:

- Input size: 224 × 224 × 3 (Height × Width × Channels).

Convolutional Layers:

- Useing 3×3 filters for all convolutional layers.
- Applying ReLU activation after each convolutional operation.
- Grouping convolutional layers into blocks:
  - Block 1: 2 convolutional layers with 64 filters each.
  - Block 2: 2 convolutional layers with 128 filters each.
  - Block 3: 4 convolutional layers with 256 filters each.
  - Block 4: 4 convolutional layers with 512 filters each.
  - Block 5: 4 convolutional layers with 512 filters each.

Adding Max Pooling Layers:

- After each block, a MaxPooling layer is added with a 2×2 pool size and a stride of 2 to reduce spatial dimensions.

Flatten the Output: [0.5]

- At the end of the convolutional blocks, the 3D feature maps are flattened into a 1D vector.

Add Fully Connected Layers:

- Adding two dense (fully connected) layers with 4096 neurons each and ReLU activation.

- Adding a final dense layer with N neurons (where N is the number of classes in the dataset) and using softmax activation.
- Ploting the accuracy and saving the model.

Link: 'https://www.kaggle.com/datasets/carlosrunner/pizza-not-pizza?select=pizza_not_pizza'

In [ ]:
```
import os
import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array



pizza_path= "/kaggle/input/pizza-not-pizza/pizza_not_pizza/pizza"
notPizza_path= "/kaggle/input/pizza-not-pizza/pizza_not_pizza/not_pizza"


img_height = 224
img_width = 224
batch_size = 16

datagen = ImageDataGenerator(
    rescale=1.0/255,
    validation_split=0.2
)

train_data = datagen.flow_from_directory(
    "/kaggle/input/pizza-not-pizza/pizza_not_pizza",
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

val_data = datagen.flow_from_directory(
    "/kaggle/input/pizza-not-pizza/pizza_not_pizza",
    target_size=(img_height, img_width),
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)
```

```
Found 1574 images belonging to 2 classes.
Found 392 images belonging to 2 classes.
```

In [ ]:
```
model = Sequential()
model.add(Conv2D(64, (3,3), activation='relu', padding='same',
                 input_shape=(224, 224, 3)))
model.add(Conv2D(64, (3,3), activation='relu', padding='same'))
```

```
model.add(MaxPooling2D(pool_size=(2,2), strides=2))


model.add(Conv2D(128, (3,3), activation='relu', padding='same'))
model.add(Conv2D(128, (3,3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2,2), strides=2))


model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
model.add(Conv2D(256, (3,3), activation='relu', padding='same'))
model.add(MaxPooling2D(pool_size=(2,2), strides=2))


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


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


model.add(Flatten())
model.add(Dense(4096, activation='relu'))
model.add(Dense(4096, activation='relu'))
model.add(Dense(2, activation='softmax'))

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

model.summary()
```

Model: "sequential_1"

Layer (type)	Output Shape	Param #
conv2d_16 (Conv2D)	(None, 224, 224, 64)	1,792
conv2d_17 (Conv2D)	(None, 224, 224, 64)	36,928
max_pooling2d_5 (MaxPooling2D)	(None, 112, 112, 64)	0
conv2d_18 (Conv2D)	(None, 112, 112, 128)	73,856
conv2d_19 (Conv2D)	(None, 112, 112, 128)	147,584
max_pooling2d_6 (MaxPooling2D)	(None, 56, 56, 128)	0
conv2d_20 (Conv2D)	(None, 56, 56, 256)	295,168
conv2d_21 (Conv2D)	(None, 56, 56, 256)	590,080
conv2d_22 (Conv2D)	(None, 56, 56, 256)	590,080
conv2d_23 (Conv2D)	(None, 56, 56, 256)	590,080
max_pooling2d_7 (MaxPooling2D)	(None, 28, 28, 256)	0
conv2d_24 (Conv2D)	(None, 28, 28, 512)	1,180,160
conv2d_25 (Conv2D)	(None, 28, 28, 512)	2,359,808
conv2d_26 (Conv2D)	(None, 28, 28, 512)	2,359,808
conv2d_27 (Conv2D)	(None, 28, 28, 512)	2,359,808
max_pooling2d_8 (MaxPooling2D)	(None, 14, 14, 512)	0
conv2d_28 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_29 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_30 (Conv2D)	(None, 14, 14, 512)	2,359,808
conv2d_31 (Conv2D)	(None, 14, 14, 512)	2,359,808
max_pooling2d_9 (MaxPooling2D)	(None, 7, 7, 512)	0
flatten_1 (Flatten)	(None, 25088)	0
dense_3 (Dense)	(None, 4096)	102,764,544
dense_4 (Dense)	(None, 4096)	16,781,312
dense_5 (Dense)	(None, 2)	8,194

Total params: 139,578,434 (532.45 MB)

Trainable params: 139,578,434 (532.45 MB)

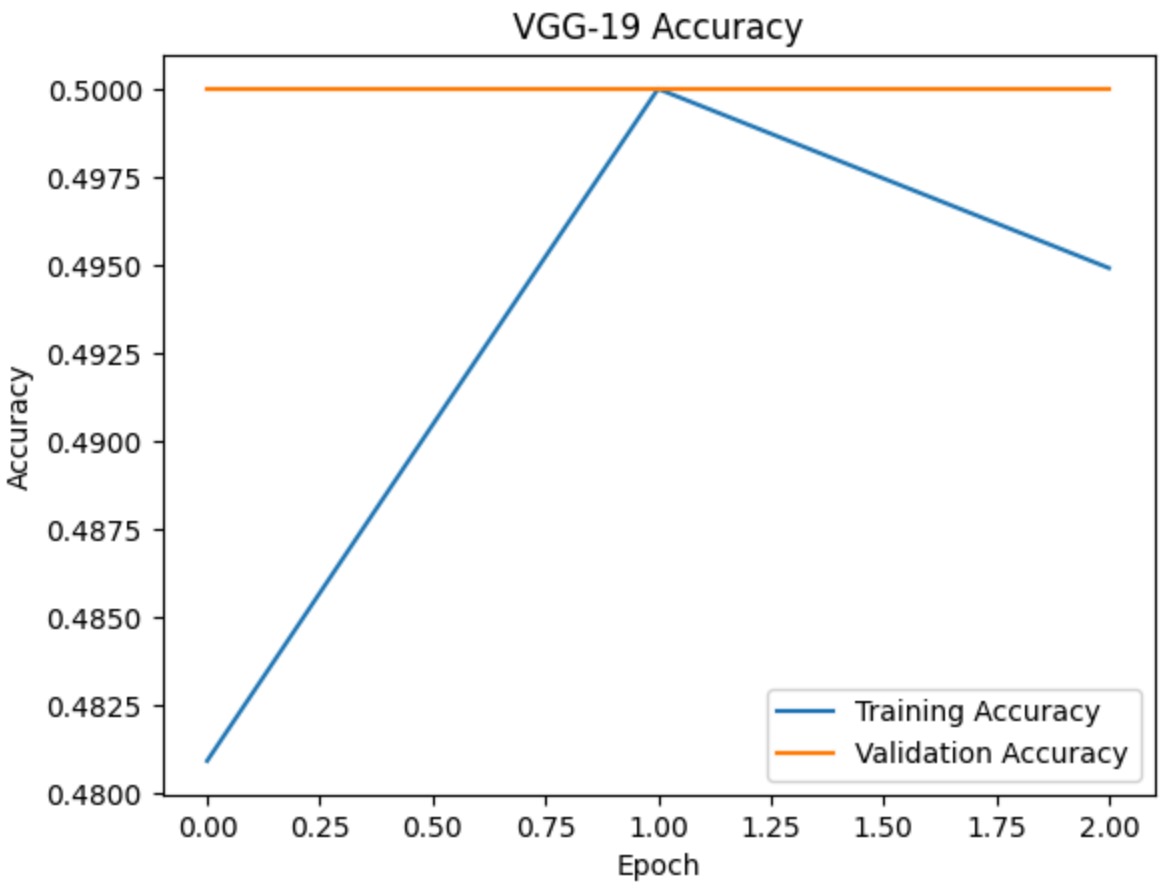
Non-trainable params: 0 (0.00 B)

In [ ]:
```
history = model.fit(
    train_data,
    validation_data=val_data,
    epochs=3
)
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/trainers/data_adapters/py_dataset_adapter.py:121: UserWarning: Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_multiprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
Epoch 1/3
99/99 ━━━━━━━━━━━━━━━━━━━━ 2130s 21s/step - accuracy: 0.4837 - loss: 1.5094 - val_accuracy: 0.5000 - val_loss: 0.6932
Epoch 2/3
99/99 ━━━━━━━━━━━━━━━━━━━━ 2118s 21s/step - accuracy: 0.5231 - loss: 0.6930 - val_accuracy: 0.5000 - val_loss: 0.6931
Epoch 3/3
99/99 ━━━━━━━━━━━━━━━━━━━━ 2123s 21s/step - accuracy: 0.4892 - loss: 0.6932 - val_accuracy: 0.5000 - val_loss: 0.6932
```

In [ ]:
```
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('VGG-19 Accuracy')
plt.legend()
plt.show()

model.save("vgg19_pizza_not_pizza.h5")
```




```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `k
eras.saving.save_model(model)`. This file format is considered legacy. We recom
mend using instead the native Keras format, e.g. `model.save('my_model.keras')`
or `keras.saving.save_model(model, 'my_model.keras')`.
```