

Implementation of VGGNet

We will use the tensorflow.keras Functional API to build VGG
(<https://arxiv.org/pdf/1409.1556.pdf>)

In the paper we can read:

- [i] "All hidden layers are equipped with the rectification (ReLU (Krizhevsky et al., 2012)) non-linearity."

[ii] "Max-pooling is performed over a 2×2 pixel window, with stride 2."

We will also use the following Diagram [iii]:

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224×224 RGB image)					
conv3-64	conv3-64 LRN	conv3-64 conv3-64	conv3-64	conv3-64	conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 conv3-128	conv3-128	conv3-128	conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256	conv3-256 conv1-256	conv3-256 conv3-256	conv3-256 conv3-256 conv3-256
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 conv1-512	conv3-512 conv3-512	conv3-512 conv3-512 conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Network architecture

- The network consists of 5 *Convolutional* blocks and 3 *Fully Connected* Layers
- Each Convolutional block consists of 2 or more Convolutional layers and a Max Pool layer

Workflow

We will:

- import the necessary layers
- write the code for the *Convolution blocks*
- write the code for the *Dense layers*
- build the model

1. Imports

Code:

```
from tensorflow.keras.layers import Input, Conv2D, \
    MaxPool2D, Flatten, Dense
```

2. Convolution blocks

We start with the input layer:

Code:

```
input = Input(shape=(224, 224, 3))
```

1st block

from the paper:

- conv3-64
- conv3-64
- maxpool

Code:

```
x = Conv2D(filters=64, kernel_size=3, padding='same', activation='relu')(input)
x = Conv2D(filters=64, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)
```

2nd block

from the paper:

- conv3-128
- conv3-128
- maxpool

Code:

```
x = Conv2D(filters=128, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=128, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)
```

3rd block

from the paper:

- conv3-256
- conv3-256
- conv3-256
- maxpool

Code:

```
x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)
```

4th and 5th block

from the paper:

- conv3-512
- conv3-512
- conv3-512
- maxpool

Code: (x2)

```
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)

x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)
```

3. Dense layers

Before passing the output tensor of the last Convolutional layer to the first `Dense()` layer we flatten it by using the `Flatten()` layer.

from the paper:

- FC-4096
- FC-4096
- FC-1000
- soft-max

Code:

```
x = Flatten()(x)
x = Dense(units=4096, activation='relu')(x)
x = Dense(units=4096, activation='relu')(x)
output = Dense(units=1000, activation='softmax')(x)
```

4. Model

In order to build the *model* we will use the `tensorflow.keras.Model` object:

Code:

```
from tensorflow.keras import Model
```

To define the model we need the input tensor(s) and the output tensor(s).

Code:

```
model = Model(inputs=input, outputs=output)
```

Final code

Code:

```
from tensorflow.keras.layers import Input, Conv2D, \
    MaxPool2D, Flatten, Dense

input = Input(shape=(224, 224, 3))

x = Conv2D(filters=64, kernel_size=3, padding='same', activation='relu')(input)
x = Conv2D(filters=64, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)

x = Conv2D(filters=128, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=128, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)

x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=256, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)

x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)

x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = Conv2D(filters=512, kernel_size=3, padding='same', activation='relu')(x)
x = MaxPool2D(pool_size=2, strides=2, padding='same')(x)

x = Flatten()(x)
x = Dense(units=4096, activation='relu')(x)
x = Dense(units=4096, activation='relu')(x)
output = Dense(units=1000, activation='softmax')(x)

from tensorflow.keras import Model

model = Model(inputs=input, outputs=output)
```

Model diagram

