VGG Net

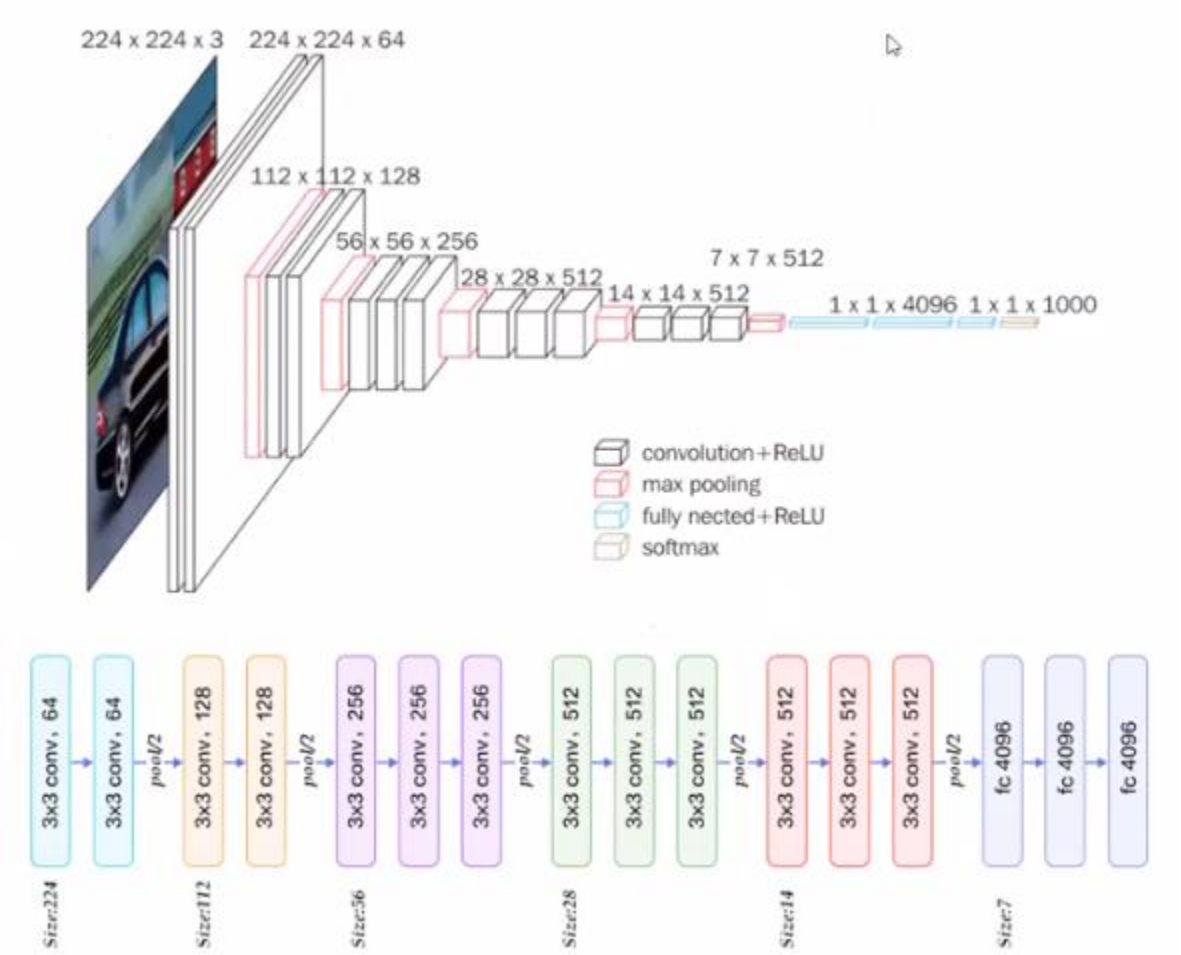
In AlexNet there was a different kernel size and total number of kernels also.

There is group called as visual geometric group (VGG) it belongs from Oxford University and this group try to do a lots of research with respect to different kinds of image nets challenges. This university have proposed one network like what if using instead of using kernel with different size and using this LRN what if we are going to use the similar kinds of kernel size.

Here they have tested with many variations like network with 11 layers or 12 layers or 13 layers. In short starting from 11 till 19th layer.

Out of which two VGG's are very popular one is VGG 16 and another one is VGG19.

Most of them are using VGG16 compare to VGG19.

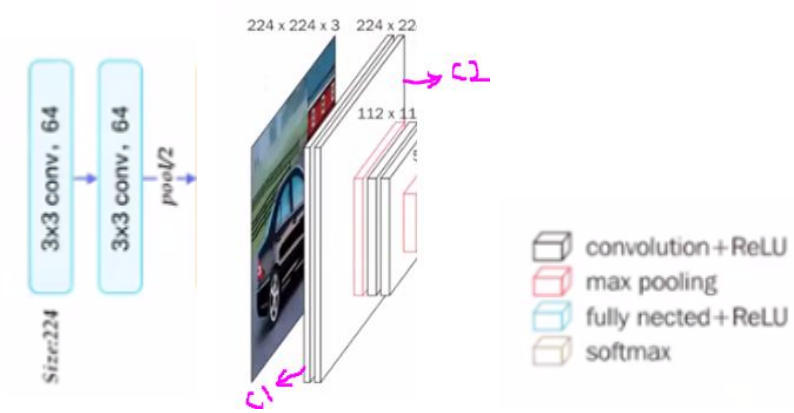


Here they have used convolution + pooling, max pooling, fully connected + relu and softmax.

In this VGG they have taken the image of size 224X224x3

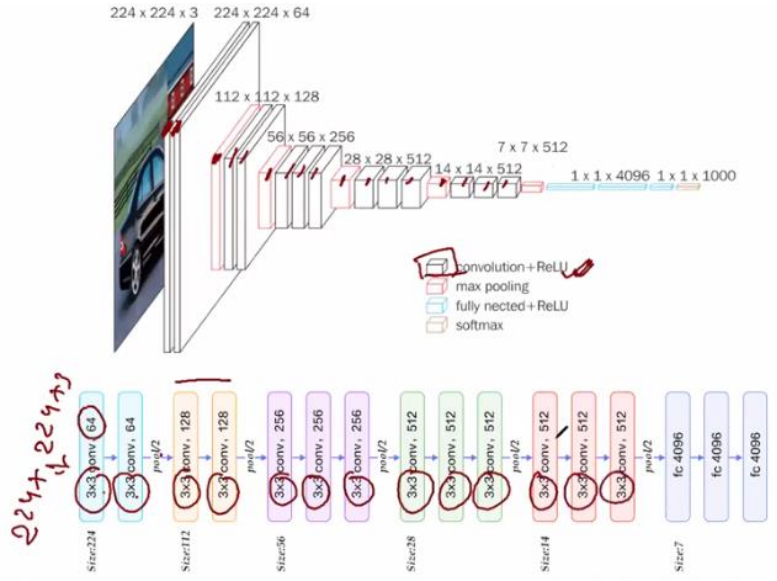
Here the input image of any size.

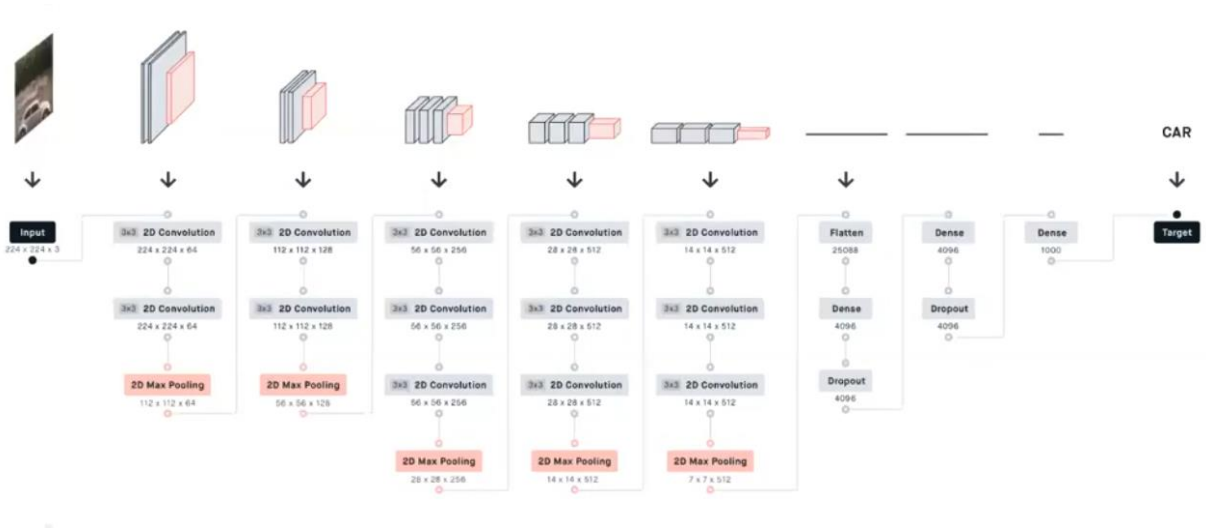
On top of that image they have applied 3X3 convolution with 64 filters. On top of that again they have applied 3X3 convolution with 64 filters and then they have applied pooling.



After that we are trying to apply a pooling and 2 convolution and 1 pooling (Max pooling).

Again we are trying to apply two convolution.





Here you will be able to observe that, every time we are trying to take a kernel of size 3X3 across the entire network and finally we are sending our data's to the fully connected layer for training.

If you try to count the number of layers then the layers will be 16. Basically it is 16 layer architecture that people have created that's the reason it is called as **VGG16.**

In AlexNet if you remember the total number of parameters like 60 million parameter over here.

Whatever layer we are trying to use in VGG16 still we have a lesser number of weights and the number of parameters are more.

The main idea or objective behind the VGG16 is to optimize the entire network. So that I will be able to get network with large depth.

If the depth of the network is increasing in that case it will be able to understand more amount of feature or high resolution data into a better way.

In VGG16 it is having a lesser amount of trainable parameter as compare to a AlexNet.

Apart from that the dense of the network is more as compare to a AlexNet.

	Layer	Feature Map	Size	Kernel Size	Stride	Activation
Input	Image	1	224 x 224 x 3	-	-	-
1	2 X Convolution	64	224 x 224 x 64	3x3	1	relu
	Max Pooling	64	112 x 112 x 64	3x3	2	relu
3	2 X Convolution	128	112 x 112 x 128	3x3	1	relu
	Max Pooling	128	56 x 56 x 128	3x3	2	relu
5	2 X Convolution	256	56 x 56 x 256	3x3	1	relu
	Max Pooling	256	28 x 28 x 256	3x3	2	relu
7	3 X Convolution	512	28 x 28 x 512	3x3	1	relu
	Max Pooling	512	14 x 14 x 512	3x3	2	relu
10	3 X Convolution	512	14 x 14 x 512	3x3	1	relu
	Max Pooling	512	7 x 7 x 512	3x3	2	relu
13	FC	-	25088	-	-	relu
14	FC	-	4096	-	-	relu
15	FC	-	4096	-	-	relu
Output	FC	-	1000	-	-	Softmax

Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The ReLU activation function is not shown for brevity.

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 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **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	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **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 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 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Summary:

Summary of VGGNet improvement points

1. A smaller 3 * 3 convolution kernel and a deeper network are used . The stack of two 3 * 3 convolution kernels is relative to the field of view of a 5 * 5 convolution kernel, and the stack of three 3 * 3 convolution kernels is equivalent to the field of view of a 7 * 7 convolution kernel. In this way, there can be fewer parameters (3 stacked 3 * 3 structures have only 7 * 7 structural parameters (3 * 3 * 3) / (7 * 7) = 55%); on the other hand, they have more The non-linear transformation increases the ability of CNN to learn features.
2. In the convolutional structure of VGGNet, a 1 * 1 convolution kernel is introduced. Without affecting the input and output dimensions, non-linear transformation is introduced to increase the expressive power of the network and reduce the amount of calculation.
3. During training, first train a simple (low-level) VGGNet A-level network, and then use the weights of the A network to initialize the complex models that follow to speed up the convergence of training .

Last time we have used 11X11 and 5X5 in our AlexNet, here we are using 3X3 and 3X3, which is equivalent to 5X5

3X3, 3X3, 3X3 which is equivalent to 7X7 – So instead of doing on time (7X7) I'm going depth of the future, example feature of square (3X3) --> out of feature I extract the feature (inside square(3X3)) -- > going inside of depth again (3X3)

Eg Human face (relu) – Eye (relu) – retina (relu) – Here I'm trying to increase the non-linearity, then my network able to understand in an efficient way.

Instead of doing this just for one time it always better to do a convolution multiple times.

Code Implementation

```
[ ]: 1 from keras.layers import Input, Lambda, Dense, Flatten
     2 from keras.models import Model
     3 from keras.applications.vgg16 import VGG16
     4 from keras.applications.vgg16 import preprocess_input
     5 from keras.preprocessing import image
     6 from keras.preprocessing.image import ImageDataGenerator
     7 from keras.models import Sequential
     8 import numpy as np
     9 from glob import glob
    10 import matplotlib.pyplot as plt
    11
    12 import warnings
    13 warnings.filterwarnings("ignore", category=FutureWarning)
```

```
[ ]: 1 IMAGE_SIZE = [224, 224]
```

```
[ ]: 1 #Give dataset path
     2 train_path = 'data/train'
     3 test_path = 'data/test'
```

```
[ ]: 1 vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
```

```
[ ]: 1 # don't train existing weights
     2 for layer in vgg.layers:
     3   layer.trainable = False
```

```
[ ]: 1 # useful for getting number of classes
     2 folders = glob('data/train/*')
     3 print(len(folders))
```

```
6
```

```
[ ]: 1 x = Flatten()(vgg.output)
     2 prediction = Dense(len(folders), activation='softmax')(x)
     3 model = Model(inputs=vgg.input, outputs=prediction)
     4 model.summary()
```

```
Model: "model_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (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
```

```
from keras import optimizers


sgd = optimizers.SGD(lr=0.01, decay=1e-6, momentum=0.9)
model.compile(loss='binary_crossentropy',
              optimizer=sgd,
              metrics=['accuracy'])
```

```

```

```
datagen = ImageDataGenerator(rescale=1.0/255.0)
train_set = datagen.flow_from_directory('data/train/', batch_size=64, target_size=(224, 224))
test_set = datagen.flow_from_directory('data/test/', batch_size=64, target_size=(224, 224))
```

```
Found 4246 images belonging to 6 classes.
Found 1854 images belonging to 6 classes.
```

```
# Data Augmentation
test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    rotation_range=40,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
```

```
from datetime import datetime
from keras.callbacks import ModelCheckpoint, LearningRateScheduler
from keras.callbacks import ReduceLROnPlateau


#num_epochs = 1000
#num_batch_size = 32

checkpoint = ModelCheckpoint(filepath='mymodel.h5',
                             verbose=1, save_best_only=True)

callbacks = [checkpoint]

start = datetime.now()

model.fit_generator(
  train_set,
  validation_data=test_set,
  epochs=10,
  steps_per_epoch=5,
  validation_steps=32,
    callbacks=callbacks ,verbose=1)


duration = datetime.now() - start
print("Training completed in time: ", duration)
```

```
Epoch 1/10
```

```
C:\Users\jiwit\AppData\Roaming\Python\Python37\site-packages\PIL\Image.py:989: UserWarning: Palette images with Transparency ex
pressed in bytes should be converted to RGBA images
  "Palette images with Transparency expressed in bytes should be "
```

```
Epoch 1/10
C:\Users\jiwit\AppData\Roaming\Python\Python37\site-packages\PIL\Image.py:989: UserWarning: Palette images with Transparency ex
pressed in bytes should be converted to RGBA images
  "Palette images with Transparency expressed in bytes should be "
5/5 [==============================] - 544s 109s/step - loss: 0.6504 - accuracy: 0.7870 - val_loss: 2.0601 - val_accuracy: 0.70
66

Epoch 00001: val_loss improved from inf to 2.06007, saving model to mymodel.h5
Epoch 2/10
4/5 [=========================>.....] - ETA: 14s - loss: 2.8518 - accuracy: 0.7214
```

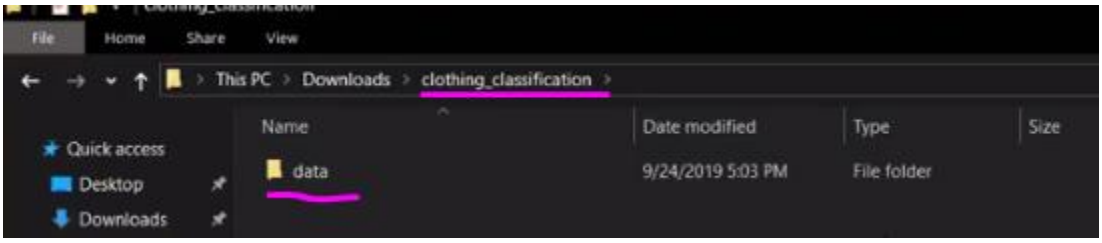
Files Running Clusters

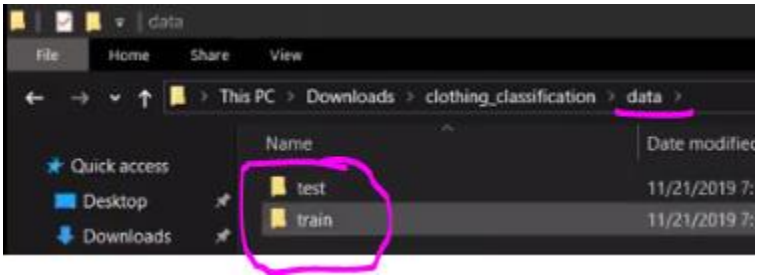
Select items to perform actions on them.

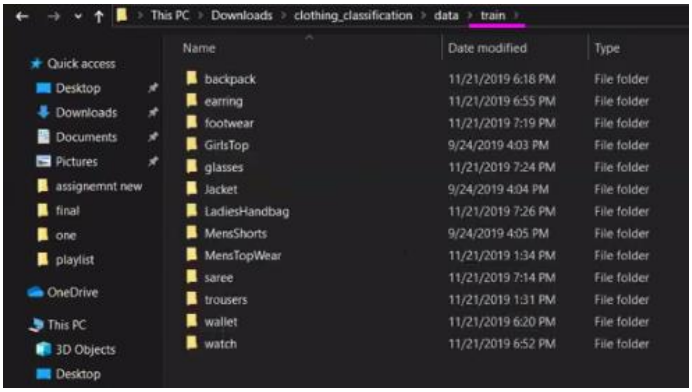
Upload New

0 / Downloads / Chapter5- Going depper with Convolutions-20200417T193801Z-001 / Chapter5- Going depper with Convolutions

Name	Last Modified	File size
..	seconds ago	
data	2 hours ago	
img	8 days ago	
Alexnet.ipynb	Running 6 days ago	31.4 kB
Cnn_Architectures.ipynb	7 days ago	149 kB
Inception.ipynb	Running 2 hours ago	35 kB
LeNet.ipynb	7 days ago	20.3 kB
Resnet.ipynb	Running 2 hours ago	30.9 kB
VGG.ipynb	Running seconds ago	20.5 kB
Assignments5.md	a month ago	1.17 kB
mymodel.h5	2 minutes ago	60.1 MB







No of classes should be same in both train and test.

Here this is the classification model I will try to create. For sure classification model will not work accurate as compare to the object detection.

First classification model - > once done

```
IMAGE_SIZE = [224, 224]
```

Image resize

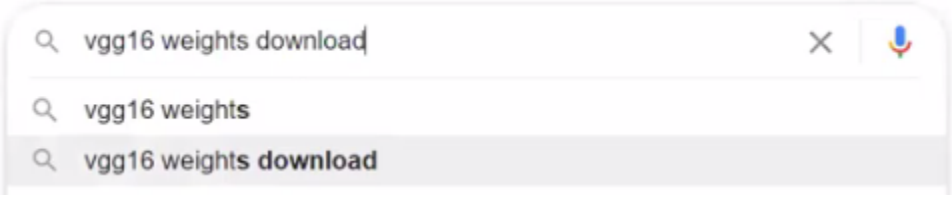
```
1 vgg = VGG16(input_shape=IMAGE_SIZE + [3], weights='imagenet', include_top=False)
2 vgg
```

```
<keras.engine.training.Model at 0x1c1e7d13e08>
```

Here we are trying to load a model, input shape 3 represents RGB

Weights and include_top:

Here we are trying to download a weights of a VGG16 if some has already trained a model, why should I trained the model again from the scratch.




tensorspace.org/html/playground/inceptionv3.html
TensorSpace.js
Documentation
Playground
Download