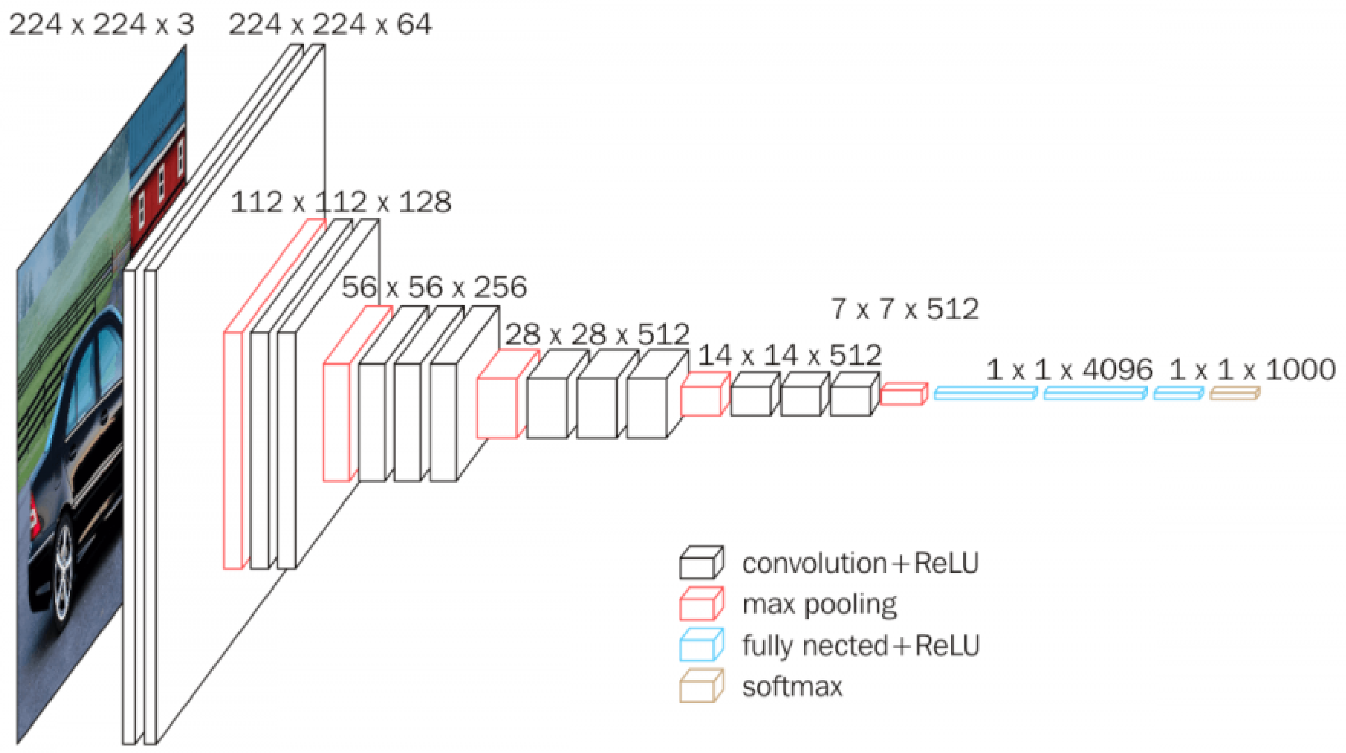
K_02 Understanding of VGG-16, VGG-19 - EN

VGG stands for Visual Geometry Group; it is a standard deep Convolutional Neural Network (CNN) architecture with multiple layers. The “deep” refers to the number of layers with VGG-16 or VGG-19 consisting of 16 and 19 convolutional layers.

The VGG architecture is the basis of ground-breaking object recognition models. Developed as a deep neural network, the VGGNet also surpasses baselines on many tasks and datasets beyond ImageNet. Moreover, it is now still one of the most popular image recognition architectures.



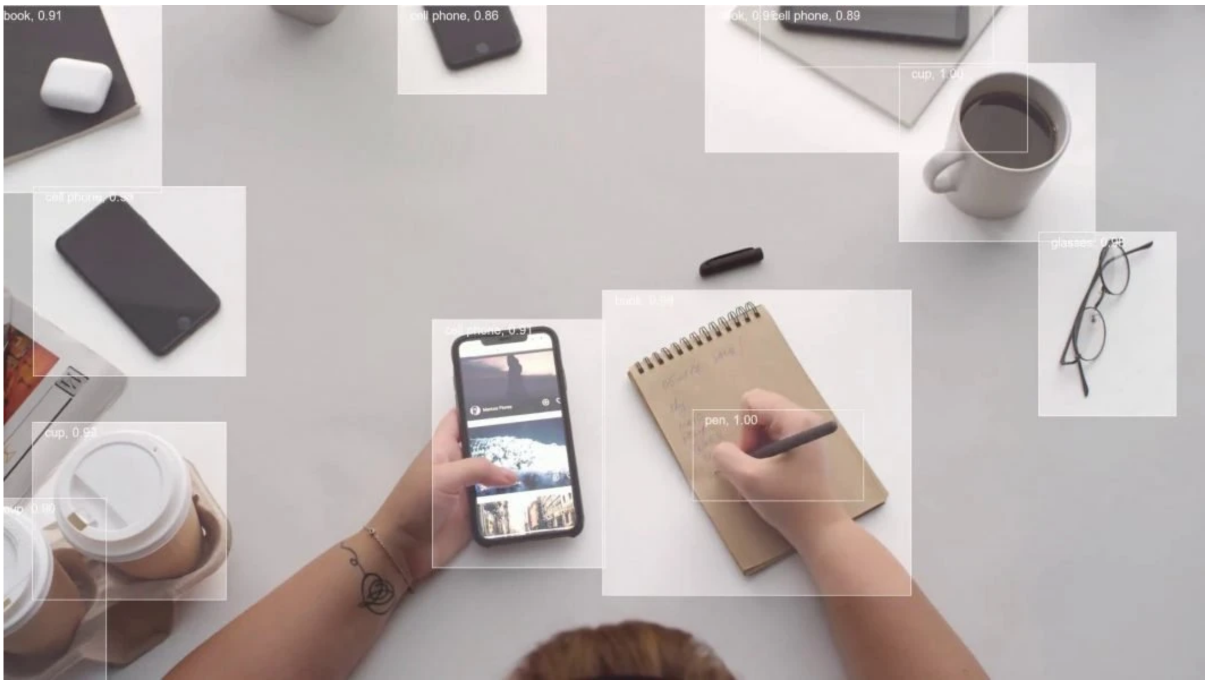
VGG Neural Network Architecture

1. What is VGG16?

The VGG model, or VGGNet, that supports 16 layers is also referred to as VGG16, which is a convolutional neural network model proposed by A. Zisserman and K. Simonyan from the University of Oxford. These researchers published their model in the research paper titled, “Very Deep Convolutional Networks for Large-Scale Image Recognition.”

The VGG16 model achieves almost 92.7% top-5 test accuracy in ImageNet. ImageNet is a dataset consisting of more than 14 million images belonging to nearly 1000 classes. Moreover, it was one of the most popular models submitted to ILSVRC-2014. It replaces the large kernel-sized filters with several 3×3 kernel-sized filters one after the other, thereby making significant improvements over AlexNet. The VGG16 model was trained using Nvidia Titan Black GPUs for multiple weeks.

As mentioned above, the VGGNet-16 supports 16 layers and can classify images into 1000 object categories, including keyboard, animals, pencil, mouse, etc. Additionally, the model has an image input size of 224×224.



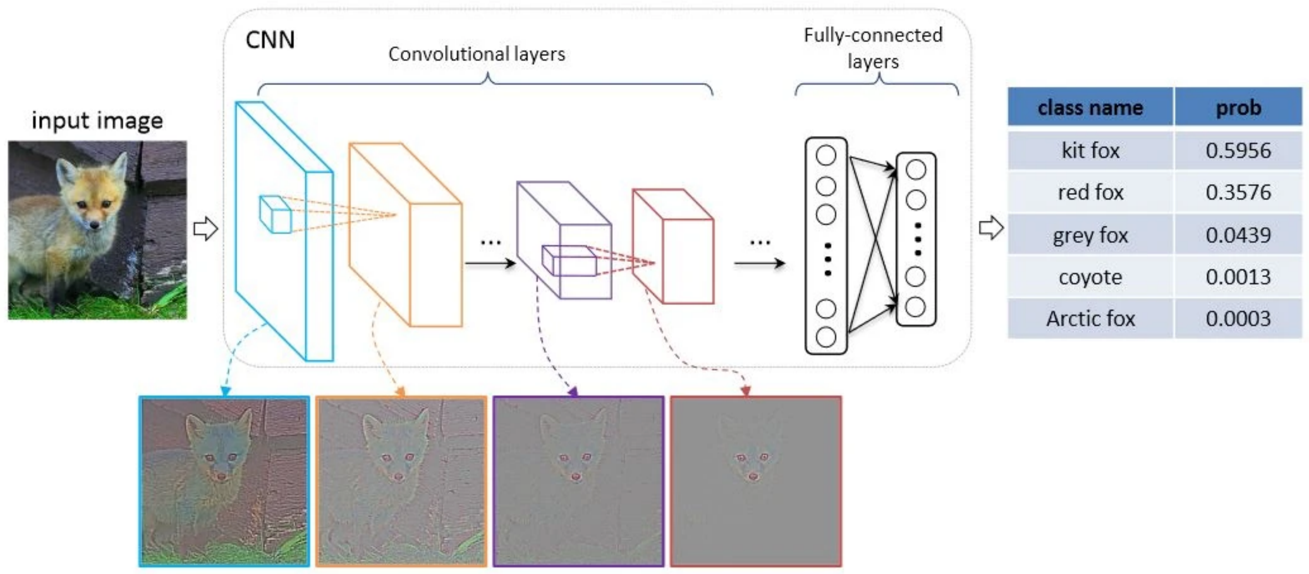
Real-time object detection application

2. What is VGG19?

The concept of the VGG19 model (also VGGNet-19) is the same as the VGG16 except that it supports 19 layers. The “16” and “19” stand for the number of weight layers in the model (convolutional layers). This means that VGG19 has three more convolutional layers than VGG16. We’ll discuss more on the characteristics of VGG16 and VGG19 networks in the latter part of this article.

3. VGG Architecture

VGGNets are based on the most essential features of convolutional neural networks (CNN). The following graphic shows the basic concept of how a CNN works:



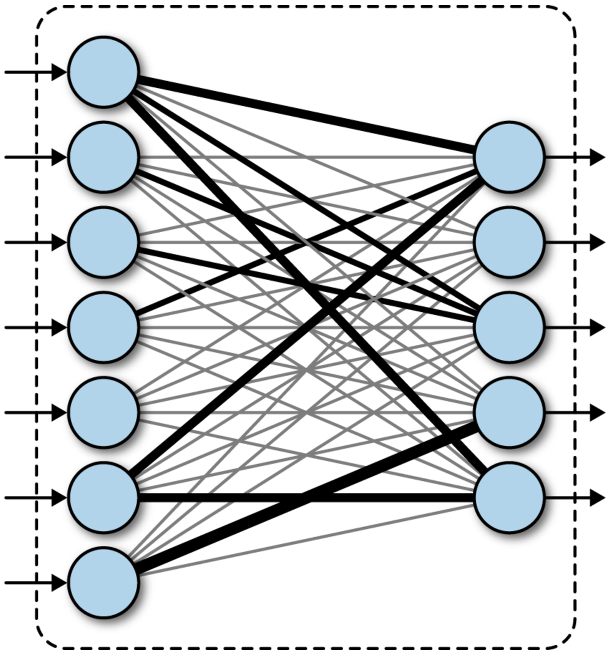
The architecture of a Convolutional Neural Network: Image data is the input of the CNN; the model output provides prediction categories for input images.

The VGG network is constructed with very small convolutional filters. The VGG-16 consists of 13 convolutional layers and three fully connected layers.

Let’s take a brief look at the architecture of VGG:

- **Input**: The VGGNet takes in an image input size of 224×224. For the ImageNet competition, the creators of the model cropped out the center 224×224 patch in each image to keep the input size of the image consistent.
- **Convolutional Layers**: VGG’s convolutional layers leverage a minimal receptive field, i.e., 3×3, the smallest possible size that still captures up/down and left/right. Moreover, there are also 1×1 convolution filters acting as a linear transformation of the input. This is followed by a ReLU unit, which is a huge innovation from AlexNet that reduces training time. ReLU stands for rectified linear unit activation function; it is a piecewise linear function that will output the input if positive; otherwise, the output is zero. The convolution stride is fixed at 1 pixel to keep the spatial resolution preserved after convolution (stride is the number of pixel shifts over the input matrix).
- **Hidden Layers**: All the hidden layers in the VGG network use ReLU. VGG does not usually leverage Local Response Normalization (LRN) as it increases memory consumption and training time. Moreover, it makes no improvements to overall accuracy.

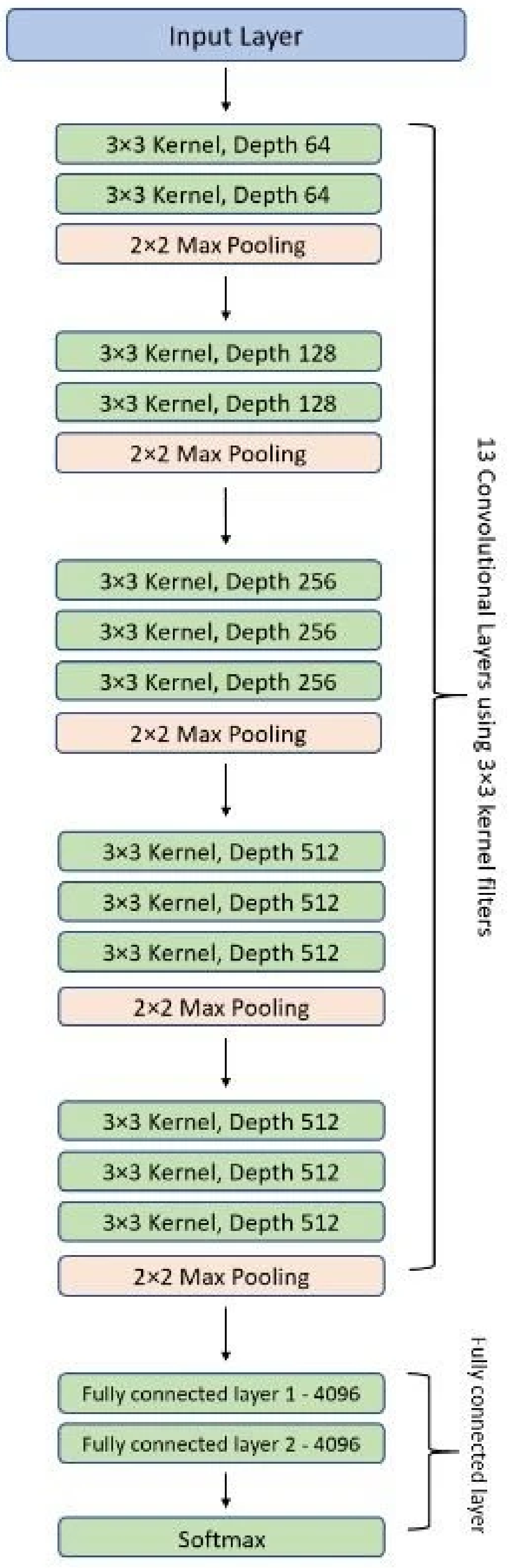
- **Fully-Connected Layers**: The VGGNet has three fully connected layers. Out of the three layers, the first two have 4096 channels each, and the third has 1000 channels, 1 for each class.


Fully Connected Layers

VGG16 Architecture

The number 16 in the name VGG refers to the fact that it is 16 layers deep neural network (VGGnet). This means that VGG16 is a pretty extensive network and has a total of around 138 million parameters. Even according to modern standards, it is a huge network. However, VGGNet16 architecture's simplicity is what makes the network more appealing. Just by looking at its architecture, it can be said that it is quite uniform.

There are a few convolution layers followed by a pooling layer that reduces the height and the width. If we look at the number of filters that we can use, around 64 filters are available that we can double to about 128 and then to 256 filters. In the last layers, we can use 512 filters.



VGG-16 Architecture of a VGG16 model

The input to the convolution neural network is a fixed-size 224 × 224 RGB image. The only preprocessing it does is subtracting the mean RGB values, which are computed on the training dataset, from each pixel.

Then the image is running through a stack of convolutional (Conv.) layers, where there are filters with a very small receptive field that is 3 × 3, which is the smallest size to capture the notion of left/right, up/down, and center part.

In one of the configurations, it also utilizes 1 × 1 convolution filters, which can be observed as a linear transformation of the input channels followed by non-linearity. The convolutional strides are fixed to 1 pixel; the spatial padding of convolutional layer input is such that the spatial resolution is maintained after convolution, that is the padding is 1 pixel for 3 × 3 Conv. layers.

Then the Spatial pooling is carried out by five max-pooling layers, 16 which follow some of the Conv. layers but not all the Conv. layers are followed by max-pooling. This Max-pooling is performed over a 2 × 2-pixel window, with stride 2.

The architecture contains a stack of convolutional layers which have a different depth in different architectures which are followed by three Fully-Connected (FC) layers: the first two FC have 4096 channels each and the third FC performs 1000-way classification and thus contains 1000 channels that is one for each class.

The final layer is the soft-max layer. The configuration of the fully connected layers is similar in all networks.

All of the hidden layers are equipped with rectification (ReLU) non-linearity. Also, here one of the networks contains Local Response Normalization (LRN), such normalization does not improve the performance on the trained dataset, but usage of that leads to increased memory consumption and computation time.

Architecture Summary:

• Input to the model is a fixed size 224×224 RGB image
• Pre-processing is subtracting the training set RGB value mean from each pixel
• Convolutional layers 17

```
        – Stride fixed to 1 pixel
        – padding is 1 pixel for 3×33×3
```

• Spatial pooling layers

```
        – This layer doesn’t count to the depth of the network by convention
        – Spatial pooling is done using max-pooling layers
        – window size is 2×22×2
        – Stride fixed to 2
        – Convnets used 5 max-pooling layers
```

• Fully-connected layers:

```
        • 1st: 4096 (ReLU).
        ▪ 2nd: 4096 (ReLU).
        ▪ 3rd: 1000 (Softmax).
```

Architecture Configuration

The below figure contains the Convolution Neural Network configuration of the VGG net with the following layers:

```
   • VGG-11
   • VGG-11 (LRN)
   • VGG-13
   • VGG-16 (Conv1)
   • VGG-16
   • VGG-19
```

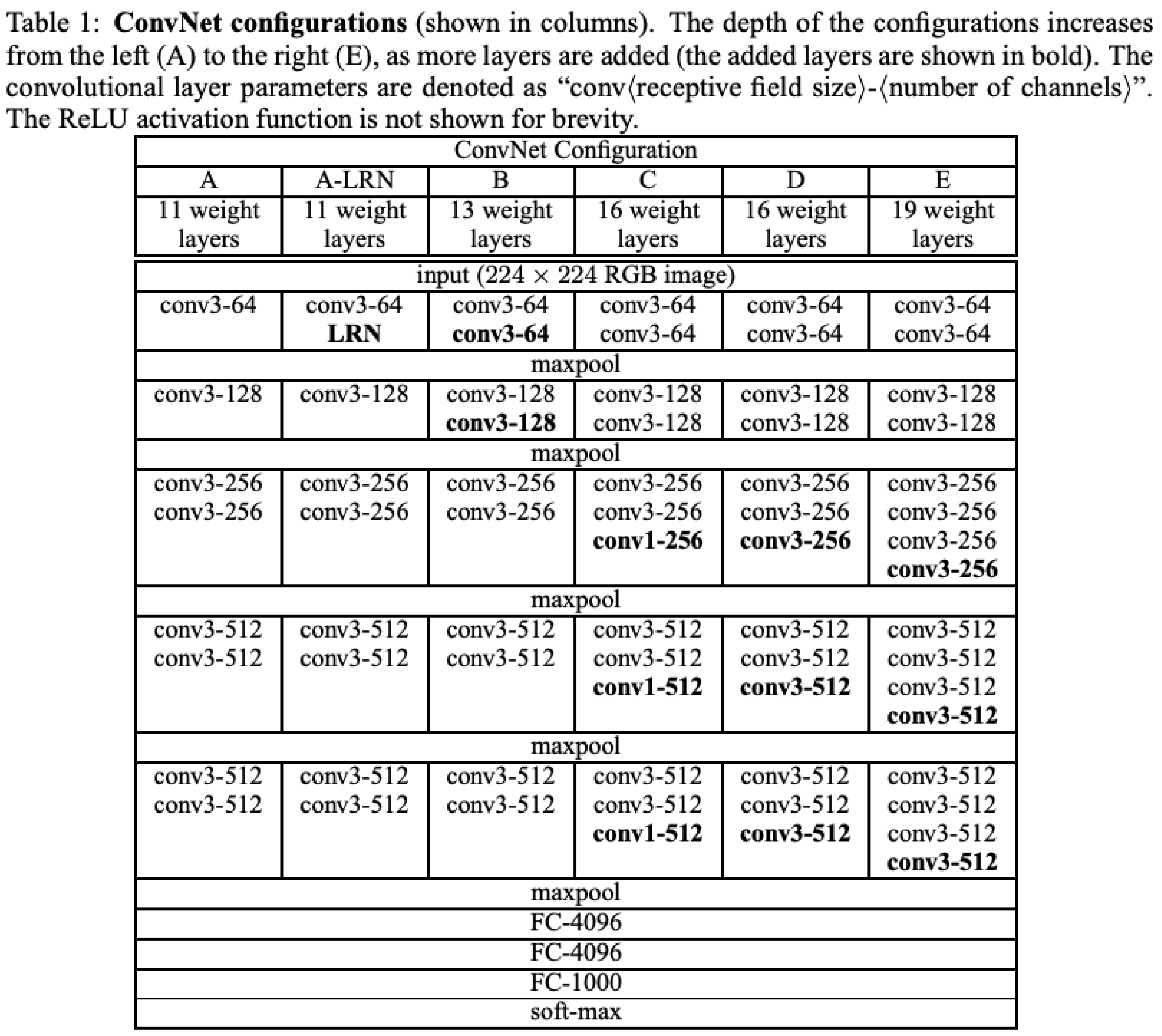


Table 1: **ConvNet configurations** (shown in columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold). The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩". The 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	conv3-64	conv3-64	conv3-64	conv3-64
	LRN	**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	conv3-256	conv3-256	conv3-256	conv3-256
			conv1-256	**conv3-256**	conv3-256
					conv3-256
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
conv3-512	conv3-512	conv3-512	conv3-512	conv3-512	conv3-512
			conv1-512	**conv3-512**	conv3-512
					conv3-512
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

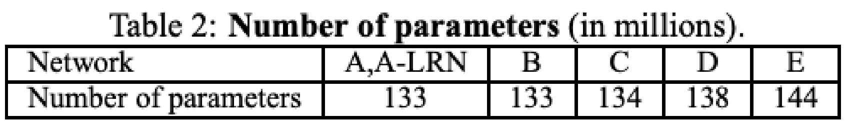


Table 2: **Number of parameters** (in millions).

Network	A,A-LRN	B	C	D	E
Number of parameters	133	133	134	138	144

The convolutional Neural Network configurations are mentioned above one per column.

In the following, the networks are referred by their names (A–E). All configurations follow the traditional design and differ only in the depth: from 11 weight layers in network A that is 8 Conv. and 3 FC layers to 19 weight layers in network E that is 16 Conv. and 3 FC layers. The width of each Conv. layer is the number of channels is rather small, which is starting from 64 in the first layer and then goes on increasing by a factor of 2 after each max-pooling layer until it reaches 512.

The number of parameters for each configuration is described below. Although it has a large depth, the number of weights in the networks is not greater than the number of weights in a shallower net with larger Conv. layer widths and receptive fields

Complexity and challenges

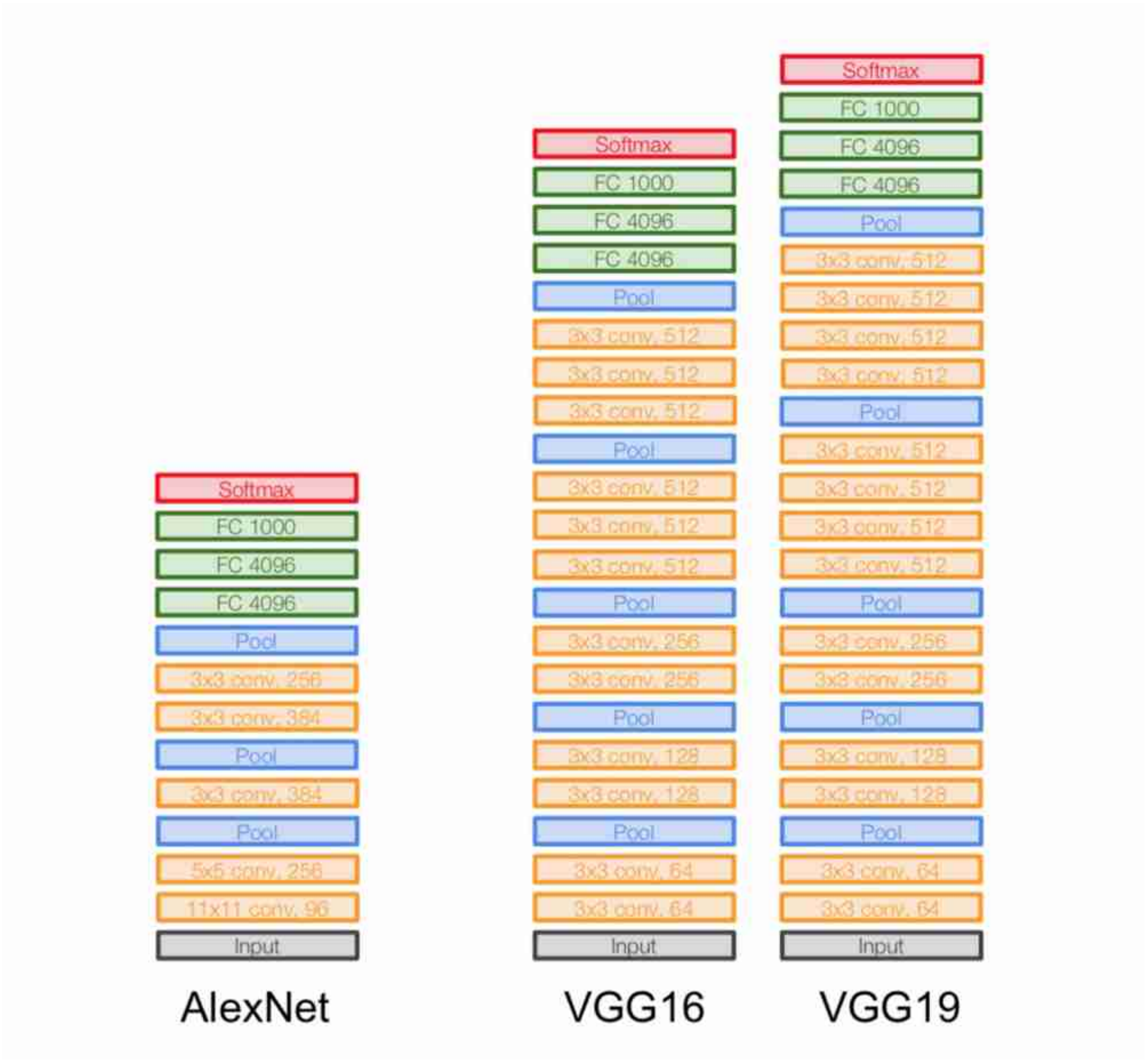
The number of filters that we can use doubles on every step or through every stack of the convolution layer. This is a major principle used to design the architecture of the VGG16 network. One of the crucial downsides of the VGG16 network is that it is a huge network, which means that it takes more time to train its parameters.

Because of its depth and number of fully connected layers, the VGG16 model is more than 533MB. This makes implementing a VGG network a time-consuming task.

The VGG16 model is used in several deep learning image classification problems, but smaller network architectures such as GoogLeNet and SqueezeNet are often preferable. In any case, the VGGNet is a great building block for learning purposes as it is straightforward to implement.

Performance of VGG Models

VGG16 highly surpasses the previous versions of models in the ILSVRC-2012 and ILSVRC-2013 competitions. Moreover, the VGG16 result is competing for the classification task winner (GoogLeNet with 6.7% error) and considerably outperforms the ILSVRC-2013 winning submission Clarifai. It obtained 11.2% with external training data and around 11.7% without it. In terms of the single-net performance, the VGGNet-16 model achieves the best result with about 7.0% test error, thereby surpassing a single GoogLeNet by around 0.9%.



4. VGGNet vs. ResNet

VGG stands for Visual Geometry Group and consists of blocks, where each block is composed of 2D Convolution and Max Pooling layers. It comes in two models — VGG16 and VGG19 — with 16 and 19 layers.

As the number of layers increases in CNN, the ability of the model to fit more complex functions also increases. Hence, more layers promise better performance. This should not be confused with an Artificial Neural Network (ANN), where an increase in the number of layers doesn't necessarily result in a better performance.

Now the question is, why shouldn't you use a VGGNet with more layers, such as VGG20, or VGG50, or VGG100? This is where the problem arises. The weights of a neural network are updated through the backpropagation algorithm, which makes a minor change to each weight so that the loss of the model decreases.

But how does it occur? It updates each weight so that it takes a step in the direction along which the loss decreases. This is nothing but the gradient of this weight which can be found using the chain rule.

However, as the gradient keeps flowing backward to the initial layers, the value keeps increasing by each local gradient. This results in the gradient becoming smaller and smaller, thereby making changes to the initial layers very small. This, in turn, increases the training time significantly.

The problem can be solved if the local gradient becomes 1. This is where ResNet comes into the picture since it achieves this through the identity function. So, as the gradient is back-propagated, it does not decrease in value because the local gradient is 1.

Deep residual networks (ResNets), such as the popular ResNet-50 model, are another type of convolutional neural network architecture (CNN) that is 50 layers deep. A residual neural network uses the insertion of shortcut connections in turning a plain network into its residual network counterpart. Compared to VGGNets, ResNets are less complex since they have fewer filters.

ResNet, also referred to as Residual Network, does not allow the vanishing gradient problem to occur. The skip connections act as gradient superhighways, which allow the gradient to flow undisturbed. This is also one of the most important reasons why ResNet comes in versions like ResNet50, ResNet101, and ResNet152.

Last edited by : May 18, 2023, 10 a.m.

Comment 0 Feedback

- **Prev** : K_01 Understanding of Alexnet - EN
- **Next** : K_03 Understanding of Inception - EN

TOP