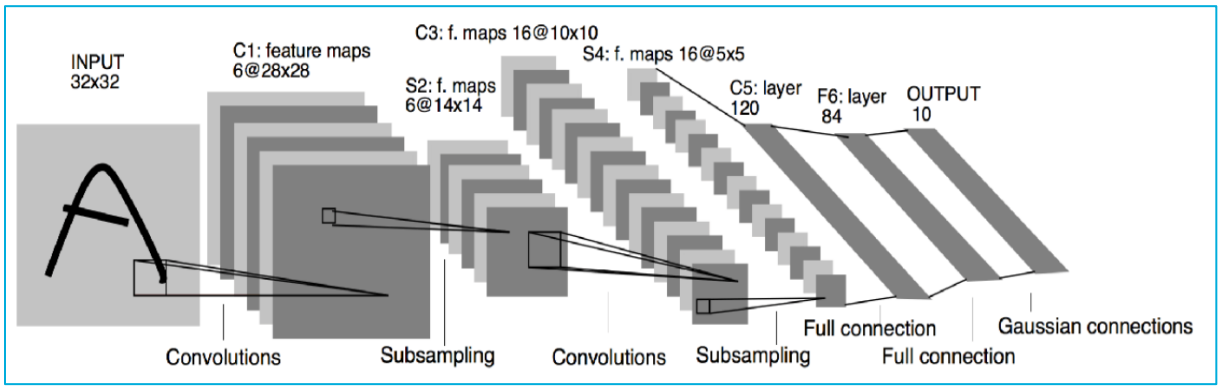
The Evolution of Image Classification Architectures

Nouran Salah

LeNet:

In early image classification research, directly applying a fully connected multi-layer perceptron to raw pixel values proved impractical. Such an approach quickly becomes computationally intractable due to the extremely high dimensionality of image data. Moreover, it fails to exploit an important property of images: **spatial correlation**, where neighboring pixels are highly related. Effective visual recognition therefore requires transforming raw images into **meaningful, low-dimensional feature representations** that preserve essential structural information. Convolutional neural networks (CNNs) address this challenge by introducing localized receptive fields and shared weights, enabling efficient feature extraction. Building on this principle, Yann LeCun's pioneering LeNet architecture proposed a multi-stage process in which convolution and pooling operations are sequentially applied to progressively learn hierarchical visual features.



First, an input image is fed to the network. Filters of a given size scan the image and perform convolutions. The obtained features then go through an activation function.
Then, the output goes through a succession of pooling and other convolution operations.

As you can see, **features** are **reduced in dimension** as the network goes on.

At the end, high-level features are flattened and fed to fully connected layers, which will eventually yield class probabilities through a softmax layer.

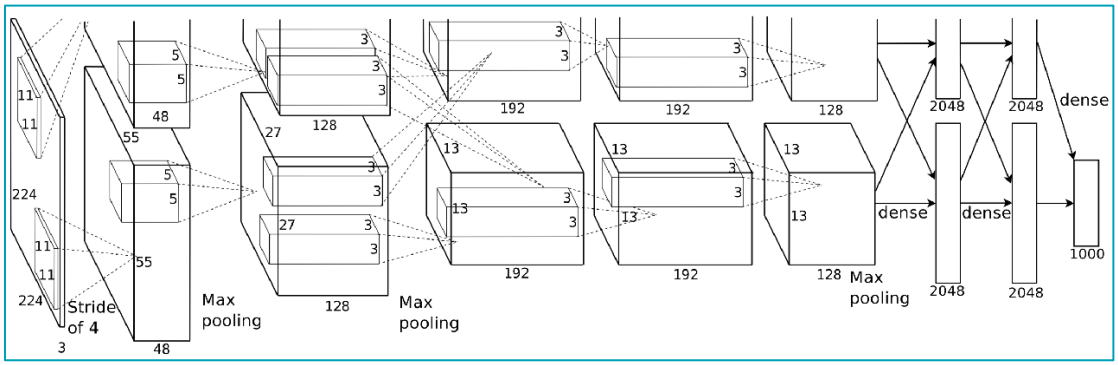
During training time, the **network learns** how to recognize the features that make a sample belong to a given class **through backpropagation**.

To give an example of what such a network can 'see': let's say we have an image of a horse. The first filters may focus on the animal's overall shape. And then as we go deeper, we can reach a higher level of abstraction where details like eyes and ears can be captured.

AlexNet

Convolution's rise to fame

Then you could wonder, why have ConvNets not been trendy since 1998? The short answer is: we had not leveraged their full potential back then.



Here, AlexNet takes the **same top-down approach**, where successive filters are designed to capture more and more subtle features. But here, his work explored several crucial details.

1. First, Krizhevsky introduced **better non-linearity** in the network with the ReLU activation, whose derivative is 0 if the feature is below 0 and 1 for positive values. This proved to be efficient for gradient propagation.
2. Second, his paper introduced the concept of **dropout as regularization**. From a representation point of view, you force the network to forget things at random, so that it can see your next input data from a better perspective.

Just to give an example, after you finish reading this post, you will have most probably forgotten parts of it. And yet this is OK, because you will have only kept in mind what was essential.

Well, hopefully, the same happens for neural networks, and leads the **model** to be **more robust**.

1. Also, it introduced **data augmentation**. When fed to the network, images are shown with random translation, rotation, crop. That way, it forces the network to be more aware of the attributes of the images, rather than the images themselves.

Finally, another trick used by AlexNet is to be **deeper**. You can see here that they stacked more convolutional layers before pooling operations. The representation captures consequently **finer features** that reveal to be **useful for classification**.

This network largely outperformed what was state-of-the-art back in 2012, with a 15.4% top 5 error on the ImageNet dataset.

VGGNet

Deeper is better

The next big milestone of image classification further explored the last point that I mentioned: **going deeper**.

And it works. This suggests that such networks can achieve a better hierarchical representation of visual data with more 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					

As you can see, something else is very special on this network. It contains almost exclusively **3 by 3 convolutions**. This is curious, isn't?

In fact, the authors were driven by three main reasons to do so:

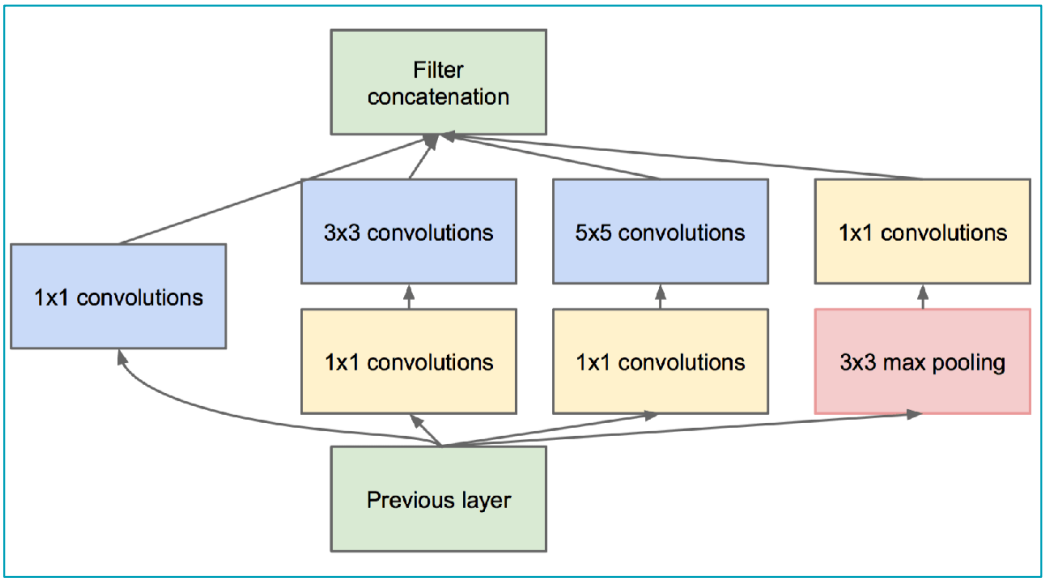
1. First, using small filters induces more non-linearity, which means **more degrees of freedom** for the network.
2. Second, the fact of **stacking** these **layers together** enables the network to **see more things than it looks like**.
   For example, with two of these, the network in fact *sees* a 5x5 receptive field. And when you stack 3 of these filters, you have in fact a 7x7 receptive field! Therefore, the same feature extraction capabilities as in the previous examples can be achieved on this architecture as well.
3. Third, using only small filters also **limits the number of parameters**, which is good when you want to go that deep.

Quantitatively speaking, this architecture achieved a 7.3% top-5 error on ImageNet.

GoogLeNet

Time for inception

Next, GoogLeNet came in the game. It bases its success on its *inception modules*.



As you can see, convolutions with different filter sizes are processed on the same input, and then concatenated together.

From a representation point of view, this allows the model to take advantage of **multi-level feature extraction at each step**.
For example, general features can be extracted by the 5x5 filters at the same time that more local features are captured by the 3x3 convolutions.

But then, you could tell me. Well, that's great. But isn't that insanely expensive to compute?

And I would say: very good remark! Actually, the Google team had a brilliant solution for this: **1x1 convolutions**.

1. On the one hand, it **reduces** the **dimensionality** of your features.
2. On the other, it combines feature maps in a way that can be **beneficial from a representation perspective**.

Then you could ask, why is it called *inception*? Well, you can see all of those modules as being networks stacked one over another inside a bigger network.

And for the record, the best GoogLeNet ensemble achieved a 6.7% error on ImageNet.

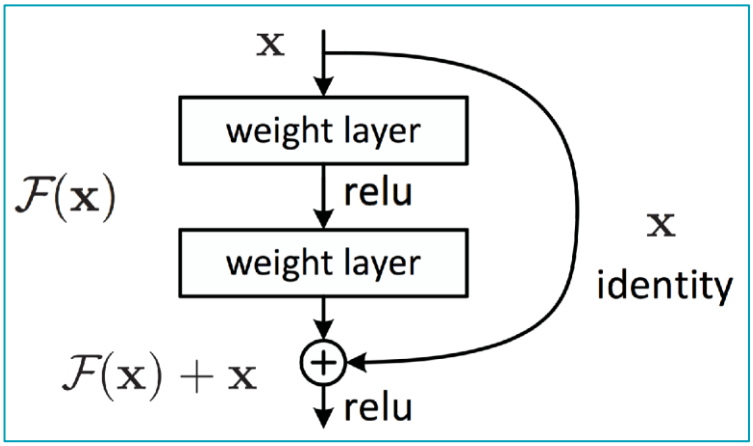
ResNet

Connect the layers

So all these networks we talked about earlier followed the same trend: going deeper.
But **at some point**, we realize that stacking **more layers** does **not lead to better performance**. In fact, the exact **opposite** occurs. But why is that?

In one word: the **gradient**, ladies and gentlemen.

But don't worry, researchers found a trick to counter this effect. Here, the key concept developed by ResNet is *residual learning*.



As you can see, every two layers, there is an **identity mapping** via an element-wise addition. This proved to be very **helpful for gradient propagation**, as the error can be backpropagated through **multiple paths**.

Also, from a representation point of view, this helps to **combine different levels of features** at each step of the network, just like we saw it with the *inception modules*.

It is to this date one of the best performing network on ImageNet, with a 3.6% top-5 error rate.
