Advanced CNN Architectures
ResNets, Inception Modules, & Xception

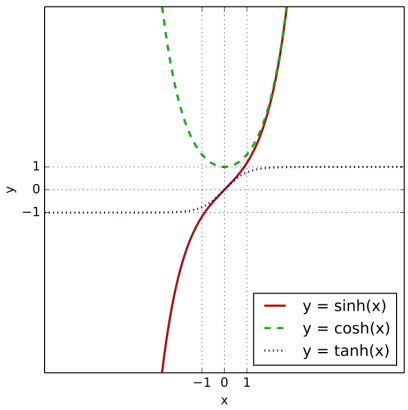
Tarushii Goel*

2020

1 ResNets

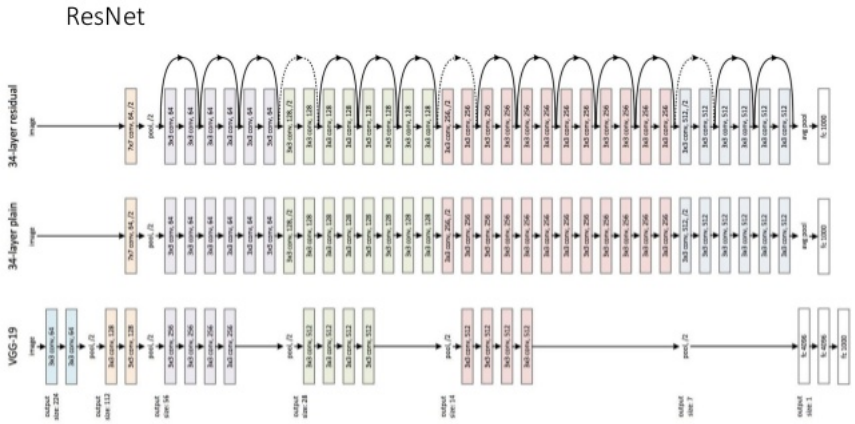
1.1 Motivation

For many years, the trend in machine learning was to add more layers, since larger networks can capture more complexity (case in point, VGGs). However, one big issue that arises with this approach is the **vanishing gradient**. Since gradients are calculated by multplying partial derivatives, if the partial derivatives are small, as is often the case in practice, the gradients become exponentially smaller, or vanish, as you back propogate to the initial layers to your network. This results in initial layers recieving very small updates. For example, the derivatives of the hyperbolic tangent function are in the range (0,1), so backpropogation through this function will always cause the gradient to decrease. The initial layers train so slowly that the benefit of having them quickly declines. In fact, the initial layers can start to hurt the accuracy of the model. To solve these issues, researchers invented residual connections, which are "skip" connections that fuction as information highways for passing gradient information and prevent a vanishing gradient.

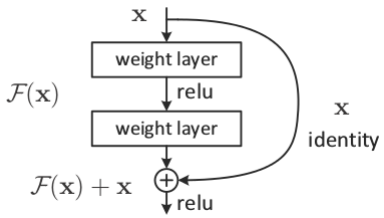


*This lecture is adapted from Justin Zhang's 2017 lecture on ResNets and Inception

1.2 Residual Blocks



As you can see, ResNet allows for much greater depth than VGG allows. The image above is the smallest conventional size of ResNet. Typical applications use ResNet-50 at least and high-end ones use ResNet-152. The sheer power of these structures has allowed for 92.9 percent accuracy on ImageNet and the usage of training on fancy GPUs with lots of data.



To understand the underlying idea behind resiudual connections, let's zoom into one block of a ResNet. The 2 layers shown in the diagram above receive an input, x, from the previous layer in the model. We will define the function that we are trying to learn as $H(x)$. In a standard network, we would train this block to learn $H(x)$. However, since this is a residual block, we are instead going to train the model to learn $F(x) := H(x) - x$. $F(x)$ is the residual. The main hypothesis of Residual Networks (one that has been shown to be true in practice) is that $F(x)$ is an easier function to learn than $H(x)$.

There are a number of observations you should note about shortcut connections:

1. They do not add extra parameters, and thus no extra computational complexity.
2. Since nonlinearities are universal approximators of functions, clearly the residual unit is also a universal approximator.
3. The degradation problem suggests that the solvers might have difficulties in approximating identity mappings with multiple nonlinear layers. However, with the residual learning formulation, in situations where identity mappings are optimal (or close to optimal), the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings. This means that we can reasonably expect that a deeper residual network will never be worse that its shallower counterpart.

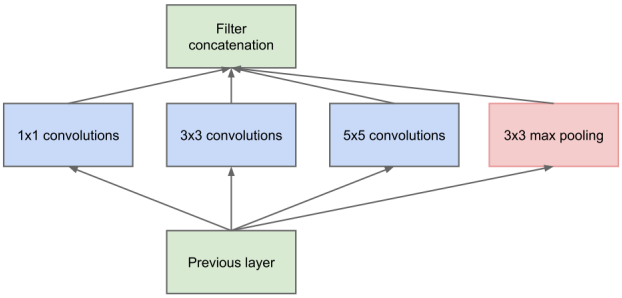
1.3 Residual Blocks, More Formally

The residual block protrayed above is defined as:

$$y = F(x; W_i) + x$$

where x is the input into the block and y is the output of the block. F is the function applied by the 2 stacked non-linear layers, $F = W_2\sigma(W_1x)$ (biases are ommited for notational convenience), and σ is the ReLU activation. A second non-linearity is applied after the addition, $\sigma(y)$. $F(x)$ may vary, depending on the number layers you have, and may produce an output of different dimensions than x. In this case, we can perform a linear projection W_s on x: $y = F(x; W_i) + W_sx$. This is sufficient to solve the degradation problem. Other methods (zero-padding, for instance) can also work, but the empirical difference between these methods is very small.

2 Inception



2.1 Overview

Another major improvement made in recent years in image classification is the use of multiple parallel layers in each residual node. Each of these sets of layers, as seen in the image above, contains different filter sizes, allowing for a varying sizes of features to be extracted, the removing the need to optimize filter size as another hyperparameter. These innovations have brought about Inception-ResNet structures capable of producing results more accurate than humans on image classification.

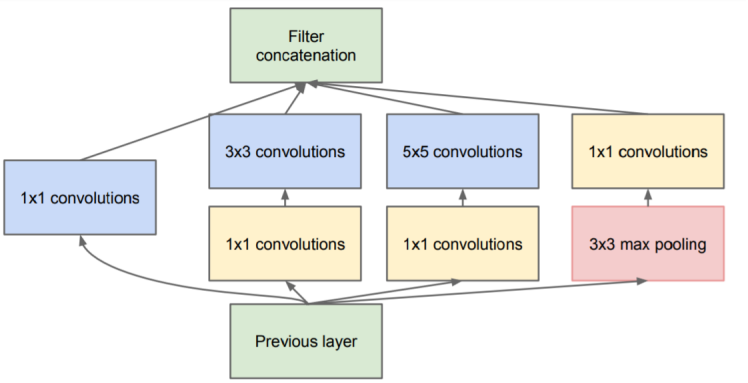
2.2 Specifics

The idea of Inception layers was based off the "we need to go deeper" internet meme. This automatically gives it more legitimacy over general neural networks, which were modeled after the brain, and reinforcement learning, which was modeled after operant conditioning.

"Deeper," in this context, refers to two things: a new level of network organization and literal deeper networks.

To understand the Inception architecture, consider a fundamental trade-off of convolutional networks: the most straightforward way to improve performance is to increase network size. This comes with two drawbacks: overfitting (due to more parameters) and a large drop in performance. Sparsely connected architectures could theoretically solve this as well as better approximate biological processes and single out discrimanatory features, but technical details prevent modern computers from efficiently doing numerical computations with sparse matrices.

The Inception architecture seeks to use varying filter sizes (allowing the model to choose the optimal one, or even combine them) while trying to mimic the result of a sparse structure.



This revised architecture uses dimensionality reduction to make the inception module more efficient and to keep representations of data sparse (i.e. dense convolutions are used with sparser data). Specifically, max pooling (for obvious reasons) and 1x1 convolutions are used.

These 1x1 convolutions do the following:

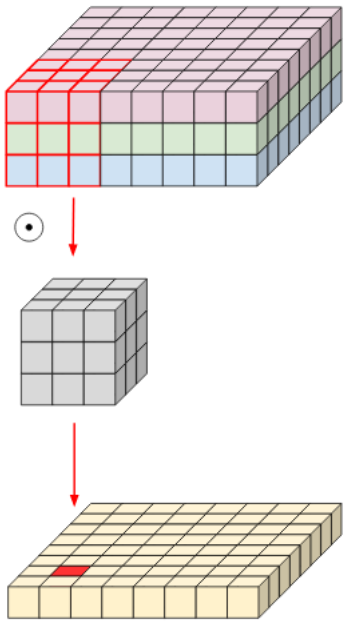
1. Make the network deeper
2. Reduce dimensions (the number of feature maps)
3. Add more non-linearities (ReLU after the 1x1 convolutions)

Generally, Inception modules are only used in the beginning of a convolutional network, for memory efficiency.

3 Xception

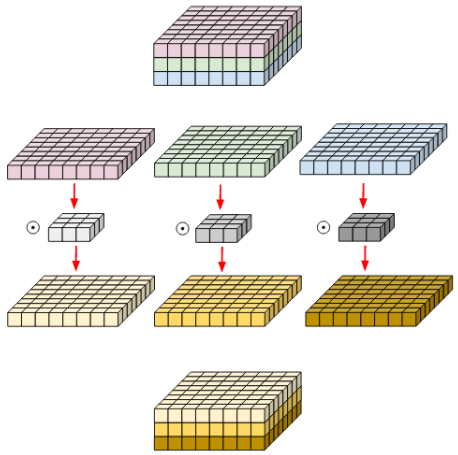
Xception is like Inception in most regards, but differs in one integral aspect: it uses depthwise seperable convolutions instead of standard convolutions.

3.1 Our Standard Convolution



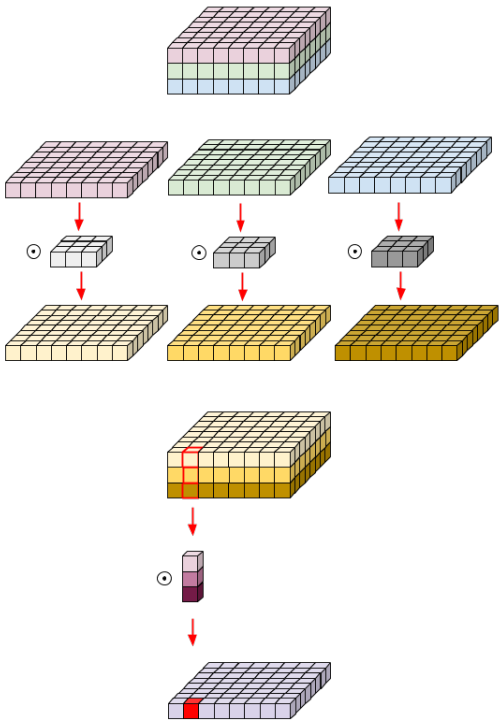
3.2 The Depthwise Convolution

In a depthwise convoution, rather than apply a 3d kernel on a 3d input, the 3d input is split, depthwise, into 2d inputs, and a 2d kernel is applied on each of the 2d inputs. The 2d outputs for each input are concatenated at the end.



3.3 The Depthwise Separable Convolution

The depthwise separable convolution extends the concept of the depthwise convolution by adding one extra step. After the outputs are concatenated, a $n*1*1$ standard convolution is applied to them (n is the number of input channels). This step combines the outputs depthwise.



The key advantage of the depthwise seperable convolution is that it requires fewer parameters than the standard convolution. To transform a $3*8*8$ input to a $n*8*8$ output, a depthwise seperable convolution world require $27+3*n$ parameters. 27 parameters are used in the depthwise convolution (three $1*3*3$

kernels), and n $3 * 1 * 1$ kernels are used to apply the last step, with each $3 * 1 * 1$ kernel generating one of the output channels. On the other hand, a standard convolution requires $27 * n$ parameters, a much larger number when $n > 1$. Note that we are using zero padding in these examples.

4 Sources

- Justin Zhang's Lecture
- Vanishing Gradient Wikipedia Article
- ResNet Paper
- Andrew Ng's Video on Inception
- ResNet Code