# Residual Connections and Inception Modules

Tarushii Goel*

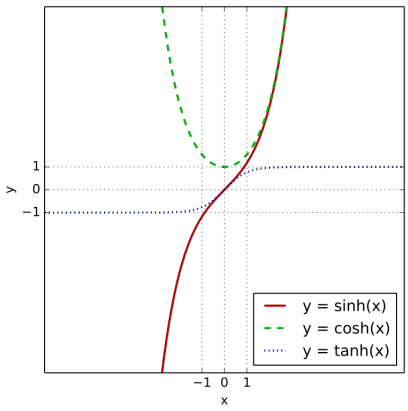
December 2017

# 1 Introduction

# 2 Residual Connections

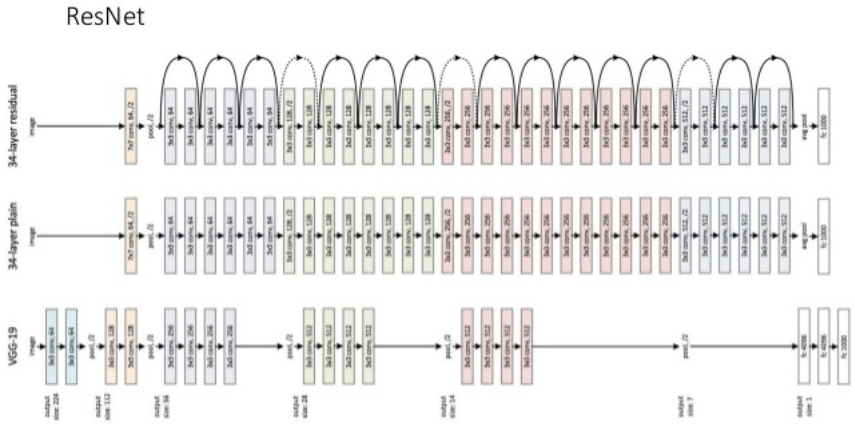
## 2.1 Motivation

For some years, the trend in machine learning was always adding more layers, as larger networks can capture more complexity (case in point, VGGs). However, one big issue that exists with this is the **vanishing gradient**. Since gradients are calculated by multplying partial derivatives, if the partial derivaties are small (as is often the case in practice), the gradients become exponentially smaller as you back propogate to the initial layers to your network to the point that initial layers recieve very little updates. For example, if use the hyperbolic tangent function, the derivatives are in the range (0,1), which means when you backpropogate through this function the gradient will always decrease. The initial layers train so slowly that the benefit of having them quickly saturates. Related to this is the **degradation problem**, which is the phenomenon that accuracy can start to decrease as networks get larger. To solve this, researchers produced residual connections, which are "skip" connections that fuction as information highways for passing gradient information and preventing a vanishing gradient.

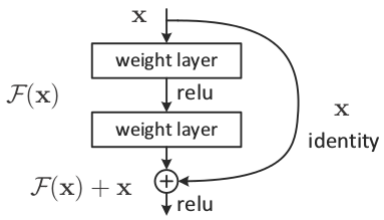



*This lecture is adapted from Justin Zhang's 2017 lecture on ResNets and Inception

## 2.2 Idea




As you can see, ResNet allows for much greater depth compared to VGG. 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 understanding 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. Let's define the function that we are trying to learn as $H(x)$. In a standard network, we would just train this block to learn $H(x)$. However, since this is a residual block, we are going to do something different. We are 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.

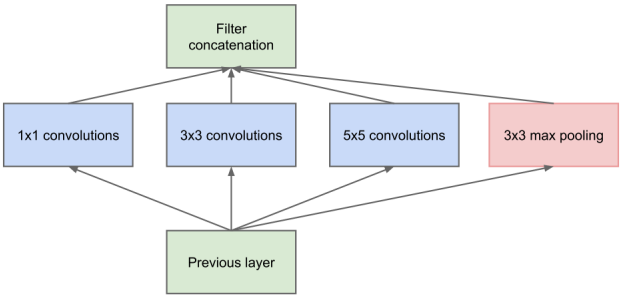
## 2.3 Residual Blocks, More Formally

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 $\sigma$ 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.

# 3 Inception




## 3.1 Overview

The next major improvement in recent years in image classification has been in the usage 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. This has culminated in Inception-ResNet structures capable of producing results more accurate than humans on image classification.

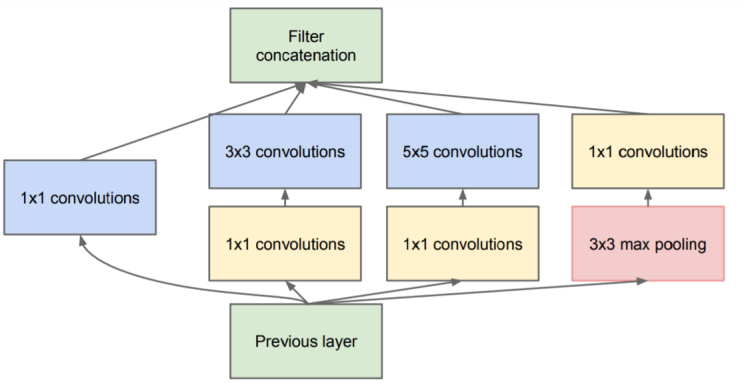
## 3.2 Specifics

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.

# 4 Sources

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