Week 4 Paper Summaries

Joseph Camacho

July 18, 2022

Layer Normalization

Layer normalization consists of normalizing each layer in the neural network so that the expected value of the neurons are 0 and their variance is 1. Note that this differs from batch normalization because batch normalization takes the expected value and variance for a *single* neuron over all the training data (or an estimate done by batches) whereas layer normalization takes the expected value and variance over all the neruons in a single layer in a single instance of the training data.

Batch normalization, weight normalization, and layer normalization are all invariant to different modifications of the dataset. Layer normalization is invariant to weight matrix re-scaling, weight matrix re-centering, dataset re-scaling, and single training case re-scaling. It is not invariant to weight vector re-scaling and dataset re-centering.

The paper then goes into some Riemannian geometry that doesn't serve much of a purpose except showing off how smart the author is. (Maybe I'm missing something, but the conclusion they come to, "The curvature along the w_i direction will change by a factor of $\frac{1}{2}$ because the σ_i will also be twice as large," could probably have been explained solely using first-year differential equations.)

Empirically, layer normalization is competitive with or better than other methods (e.g. batch normalization & no normalization) on a large number of tasks, including handwriting generation, filling in blanks, and skip-thought generation. Layer normalization is worse at training convolutional networks, however.

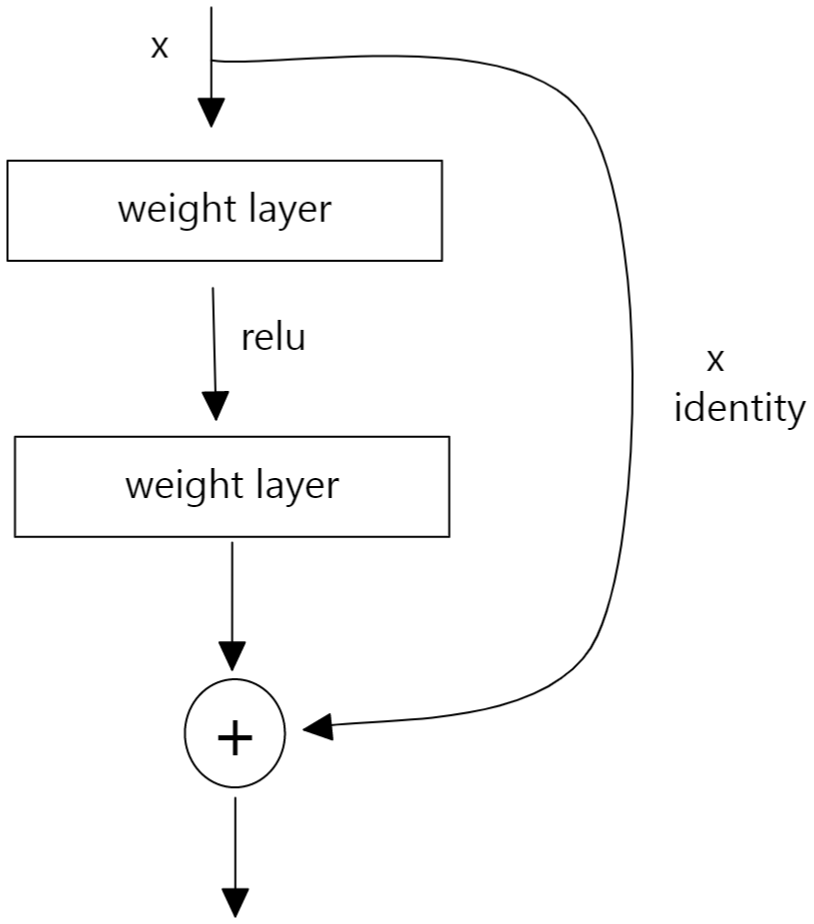
Weaknesses

This does very little to stop overfitting. It also fails to solve the vanishing gradient problem (and may in fact make it worse, since small weights can have an even smaller when the layer is normalized). The only thing layer normalization seems to help with is training time, and even then it doesn't do much better than batch normalization for anything except recurrent neural networks with small batch sizes.

Deep Residual Learning for Image Recognition

Neural networks appear to degrade in accuracy (even on the training set) as they get deeper. However, having deep neural networks is valuable--they are able to implicitly understand higher level features with more layers. Residual learning is a technique to stop degradation from happening.

The basic building block of residual learning is the following:



An identity function is added to the ReLU. The identity function creates a shortcut path for gradients to update earlier layers. Networks with these shortcuts can be much deeper and still efficiently trained. At the time of its publication, using an ensemble of neural networks with varying depths and residual learning led to the best classification in the world on ImageNet, with a top-5 error of 3.57%. It was similarly impressive on the CIFAR-10 dataset.

The standard deviation of residual responses indicates that ResNets really do learn residual functions (as opposed to just copying the previous layer through the shortcut), and each residual layer has a smaller impact as more layers are introduced.

ResNets stop improving for extremely large models, but are still trainable.

Strength

ResNets are fairly easy to implement--just add shortcuts between layers of a normal neural network.

Weaknesses

ResNets don't handle transitioning between different dimensions very well.

ResNets require batch normalization (or at least the ones in the paper did), which isn't good for recurrent neural networks.