Paper Summaries

By James Camacho

Deep Residual Learning for Image Recognition

Neural nets are just function approximaters. However, the activation functions make them not so great at approximating the identity function compared to zero. If you have a decently good function approximater, and then add a hundred layers to it, the neural net might become worse at approximating. Residual networks solve this by allowing the last layers to approximate zero instead of an identity function.

The authors' add "gap" connections between layers, adding one layers' values to another several down the line. If a network with ten layers is decently good after the first eight, and only needs a few small corrections, then a gap connection from the eighth to the tenth layer allows the last two to focus solely on those few small corrections, which are approximately zero.

The results are quite impressive, allowing hundreds of layers in a neural network while the previous state-of-the-art results began having higher errors after just a few dozen. This paper allowed deep neural nets to become really deep. One question they do not resolve though is why plain neural nets cannot handle so many layers. The gradients aren't vanishing, so you would expect them to still learn faster than a plain network with half as many layers, yet the opposite seems to occur.

Layer Normalization

The idea with layer normalization is to keep neuron values spread out. If you initialize the weights/biases in a neural net poorly, each layer would increase in value until everything is near one. Even worse, if the outputs in an RNN are slightly off from zero, it's inevitable that the values will increase every time step. Layer normalization fixes this by shifting/rescaling the outputs of each layer to have mean zero and standard deviation one (or whatever scale you want).

I added this paragraph after I realized more info was wanted in these summaries. I could rewrite layer normalization with just the above paragraph, but I guess that isn't technical enough?

They do a little analysis on how layer normalization works. Well really, that's like 75% of the paper which is kind of dumb... because who cares? One graph of it working better should be enough, but I guess they got to impress the peer reviewers. Anyway, it worked. It trained way faster, because the gradients didn't disappear, whereas in batch normalization or no normalization large values in each layer lead to almost no gradients. Lots of pretty graphs, and some not-so-pretty ones too.

One thing kind of interesting was the Riemannian manifold geometry analysis. The conclusion was that layer normalization leads to a natural regularization, as increasing weights has diminishing returns. My guess is they used these manifolds to explore what happens, but I think once they had the conclusion they should have gone back and found a simpler proof for the conclusion.

Okay back to my original summary.

I think this is a terrible idea. Sure, it prevents the values from running off, which is really important for getting strong gradients, but sometimes you actually need a large mean or small variance. For example, say you're trainning a Deep RL model to count cards in Black Jack. The mean of the layers should be pretty correlated with how many face cards it's seen, but layer normalization will wipe away that information. Or maybe you're training a neural net to mask edges of an image (i.e. outlines of all the objects in an image). If there are a lot of edges, the neural net should have a higher variance in each of its layers then if there isn't, but again, layer normalization would wipe this out.