https://arxiv.org/pdf/1512.03385.pdf

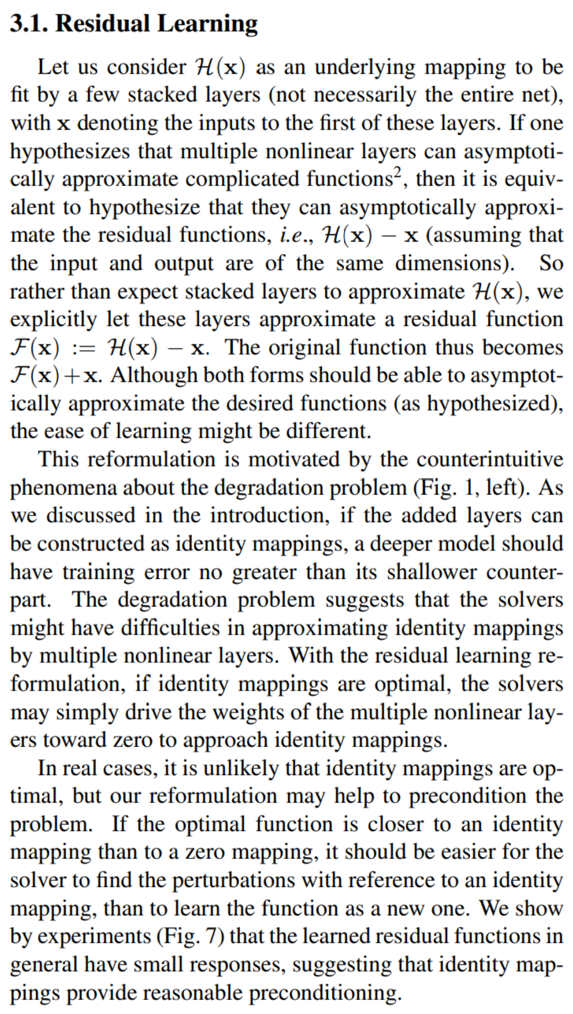
Problem: beyond a point, adding complexity to a model (in the form of additional non-linear layers) actually degrades model performance

- this is not because of overfitting. Adding complexity to a model can sometimes increase *training* error, not validation!
  - the deeper network also displays worse test error
- you can trivially compare performance between a 'simple' model 1, of depth n_1, with a more complex model 2, of depth n_2, with $n_2 > n_1$, by stacking on depth to the simple model in the form of identity layers
  - then, you would expect 2 to display no worse training error than model 1
  - but it does!
- this instead suggests something about the capability of our solvers/optimizers. There is some kind of degradation problem here that we are not able to handle
  - argument: this is unlikely to be because of vanishing gradients.
    * models trained with BN (batch norm?) and authors 'ensure' that back prop gradients have healthy norms
      · How??
    * I dont understand why batchnorm means we don't get vanishing gradients
- The degradation problem seems to be that: stacked non-linear mappings that are meant to approximate an (optimal) identity function are not learnt easily or well

One approach to handling this degradation problem: residual learning. Make some layers fit a residual mapping instead of the desired mapping. That is, assuming the desired mapping is $H(x)$, have some layers fit $F(x) = H(x) - x$ instead of $H(x)$. The original mapping $H(x)$ is instead cast into $F(x) + x$

The hypothesis is that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. In the extreme, if the optimal mapping was the identity mapping $H(x) = x$, it would be easier to push the residual to zero than to fit an identity mapping through a stack of non linear layers

- Q: Why? Intuitively it feels like all this is saying is 'this is an easier opt because all the layer weights just need to be set to zero'. But isnt the process of optimization, via gradient descent, going to be exactly as compex independent of what the true desired mapping is?
- It feels like in the other extreme, where $H(x) = constant$, this proposal would be worse for the same reason?
- From the paper section 3.1.
  –



3.1. Residual Learning

Let us consider $\mathcal{H}(\mathbf{x})$ as an underlying mapping to be fit by a few stacked layers (not necessarily the entire net), with $\mathbf{x}$ denoting the inputs to the first of these layers. If one hypothesizes that multiple nonlinear layers can asymptotically approximate complicated functions[2], then it is equivalent to hypothesize that they can asymptotically approximate the residual functions, *i.e.*, $\mathcal{H}(\mathbf{x}) - \mathbf{x}$ (assuming that the input and output are of the same dimensions). So rather than expect stacked layers to approximate $\mathcal{H}(\mathbf{x})$, we explicitly let these layers approximate a residual function $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The original function thus becomes $\mathcal{F}(\mathbf{x}) + \mathbf{x}$. Although both forms should be able to asymptotically approximate the desired functions (as hypothesized), the ease of learning might be different.

This reformulation is motivated by the counterintuitive phenomena about the degradation problem (Fig. 1, left). As we discussed in the introduction, if the added layers can be constructed as identity mappings, a deeper model should have training error no greater than its shallower counterpart. The degradation problem suggests that the solvers might have difficulties in approximating identity mappings by multiple nonlinear layers. With the residual learning reformulation, if identity mappings are optimal, the solvers may simply drive the weights of the multiple nonlinear layers toward zero to approach identity mappings.

In real cases, it is unlikely that identity mappings are optimal, but our reformulation may help to precondition the problem. If the optimal function is closer to an identity mapping than to a zero mapping, it should be easier for the solver to find the perturbations with reference to an identity mapping, than to learn the function as a new one. We show by experiments (Fig. 7) that the learned residual functions in general have small responses, suggesting that identity mappings provide reasonable preconditioning.

-

$F(x) + x$ can be realized via a shortcut conection in a feed forward network
- a shortcut connection simply skips layers
- in this case, we are implicitly using a shortcut connection to perform an identity mapping, the output of which is added to the outputs of the stacked layers

Findings:
1. extremely deep residual nets are easy to optimize, while the counterpart 'plain' nets that stack layers exhibit higher training error as depth increases
2. deep residual nets easily enjoy accuracy gains from greatly increased depth, u=proudinc better results than previous networks

But why?? What is the intuiition here behind the solvers performing better on residuals? even in expectation?