Neural Nets have been with us for a long time but they are difficult to train. To make things less complex and easier to optimize , Residual neural nets were brought into the picture.

While designing the architecture of neural nets , depth is of crucial importance ,and questions like “Would the network learn better with increased depth ?” keeps on lingering.

Earlier obstacles to answer this question were vanishing and exploding gradients which would hamper convergence of deep nets. Though that obstacle was majorly resolved to allow convergence of nets with depths in multiples of 10 . Soon enough another major problem was exposed: “Degradation” , in layman terms it means “ increasing depth of your neural net beyond a certain point your accuracy starts degrading” , and surprisingly , it ain’t because of overfitting.

Now this problem shouldn’t arise and let's see why that's so ..

Let's say you have a trained neural net of depth 20 , you add “identity mapping layers “ in between which have the same input and output , your neural net remains the same (in terms of accuracy) but its depth increases.

In another scenario , you start with as many layers as those in final neural net above , your neural net can at least reach the same state as that above and achieve the same accuracy bcz that solution is a subset of solutions of this untrained model , But the universe hates us and it doesn't happen that way (at least not in a feasible time) .

So we come up with Residual neural nets (hell ya bitch) , basically we just change the architecture a little bit and add shortcut connections to convert two or more layers into residual blocks . In a formal sense -denoting the desired underlying mapping as H(x), we let the stacked nonlinear layers fit another mapping of F (x) := H(x) − x. The original mapping is recast into F(x)+x. We hypothesize that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. These shortcut connections perform identity mapping

Degradation problem suggests that the solvers might have difficulties in approximating identity mappings by multiple nonlinear layers. With Residual Neural net architecture , the solvers may simply drive the weights of multiple non linear layers to zero to approach identity mapping via shortcut connections. This method is way easier than complex computations to decide weights to make an identity layer .

U should note that these shortcut connections should be applied on two or more layers , applying it on one would defeat the purpose as the residual function would become similar to linear function : y = W1(x) + x

Network architecture :- The baselines of ResNets were inspired by existing VGC nets. The convolutional layers mostly have 3×3 filters and follow two simple design rules:

(i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the num- ber of filters is doubled so as to preserve the time complexity per layer.

Downsampling is performed by strides of 2 and the network ends with a global average pooling layer and 1000 way fully connected layer with softmax.

Shortcut connections are inserted on generally combinations of two layers, and it's obvious that if we are downsampling on convolutional layers , the shortcut layer would also perform the same to keep dimensions the same .Padding is also performed to prevent dimensions from changing.

Internal structure of a typical residual block is

COnv - Batch normalisation - activation - Conv - batch normalisation - Add - activation

Though version 2 uses a different form :- we will look into that in a while but for now let's look at how the performance of ResNet differs from Plain Net .

Plain-18 layer net has better performance than Plain-34 -(degradation)

Plain-18 and ResNet-18 have almost same performance but ResNet converges faster

ResNet-34 has better performance than ResNet-18 .(degradation resolved)

We conjecture that the deep plain nets may have exponentially low convergence rates, which impact the reduction of the training error.

Bottleneck architecture is another optimization implemented in version2 which reduces the number of parameters .For each residual function F, we use a stack of 3 layers instead of 2 . The three layers are 1×1, 3×3, and 1×1 convolutions, where the 1×1 layers are responsible for reducing and then increasing (restoring) dimensions, leaving the 3×3 layer a bottleneck with smaller input/output dimensions , apart from this another difference is that Batch normalisation and activation come before Convolution.

Since degradation problems are not observed 50/101/152 - layer ResNet are more accurate than 34 layers but as we still observe saturation as we increase layers to 1000 .

We argue that this is because of overfitting. The 1202-layer network may be unnecessarily large (19.4M) for this small dataset. Strong regularization such as max out or dropout [14] is applied to obtain the best results on this dataset.

ResNet has revolutionised the Neural Net architecture and the ensemble of ResNets- ResNet (ILSVRC’15) reported least error among the rest. 1st places in several tracks in ILSVRC & COCO 2015 competitions: ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation were won by Residual Neural Nets.