**Deep Residual Learning for Image Recognition**

**1. Introduction -** Deeper neural networks are more difficult to train. While normalisation techniques solved the issue of vanishing/exploding gradients , with the network depth increasing, accuracy gets saturated and then degrades rapidly .

This research paper presents a residual learning framework to ease the training of networks that are substantially deeper than those used previously.

**2. Proposed methodology -**  Instead of hoping each few stacked layers directly fit a desired underlying mapping, we explicitly let these layers fit a residual mapping. Formally, 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 .This mapping is easier to optimize (hypothesis) and even in extreme cases , if the identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers by simply driving the weights of the multiple nonlinear layers toward zero to approach identity mappings.

**3. Network Architecture -** 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 number of filters is doubled so as to preserve the time complexity per layer. We perform downsampling directly by convolutional layers that have a stride of 2. The network ends with a global average pooling layer and a 1000-way fully-connected layer with softmax. The total number of weighted layers is 34 , hence called **resnet-34 .**

**shortcut connections -** The formulation of F(x) +x can be realized by feedforward neural networks with “shortcut connections” :

1. The identity shortcuts *(y = F(x, {Wi}) + x)* can be directly used when the input and output are of the same dimensions .Identity shortcut connections add neither extra parameter nor computational complexity.
2. When the dimensions increase, we consider two options:
3. The shortcut still performs identity mapping, with extra zero entries padded for increasing dimensions. This option introduces no extra parameter .
4. The projection shortcut [ y = F(x, {Wi}) + Wx ] is used to match dimensions (done by 1×1 convolutions). For both options, when the shortcuts go across feature maps of two sizes, they are performed with a stride of 2.

**Deeper Bottleneck Architectures -** Because of concerns on the training time that we can afford, we modify the building block as a bottleneck design4 . 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. **50-layer ResNet** : We replace each 2-layer block in the **resnet-34** with this 3-layer bottleneck block . Similarly , We construct 101- layer and 152-layer ResNets by using more 3-layer blocks.

**4. Implementation -**

**ImageNet Classification -**  for **Plain Networks** we have observed the degradation problem - the 34-layer plain net has higher training error than the 18-layer plain network throughout the whole training procedure . For **Residual Networks** the situation is reversed with residual learning – the 34-layer ResNet is better than the 18-layer ResNet (by 2.8%) and lower training error so is generalizable to the validation data. This indicates that the degradation problem is well addressed in this setting and we manage to obtain accuracy gains from increased depth .

**CIFAR-10 and Analysis** - The network inputs are 32×32 images, with the per-pixel mean subtracted. The first layer is 3×3 convolutions. Then we use a stack of 6n layers with 3×3 convolutions on the feature maps of sizes {32, 16, 8} respectively, with 2n layers for each feature map size. The numbers of filters are {16, 32, 64} respectively. The subsampling is performed by convolutions with a stride of 2. The network ends with a global average pooling, a 10-way fully-connected layer, and softmax. There are totally 6n+2 stacked weighted layers.

We compare n = {3, 5, 7, 9}, leading to 20, 32, 44, and 56-layer networks. The deep plain nets suffer from increased depth, and exhibit higher training error when going deeper .ResNets manage to overcome the optimization difficulty and demonstrate accuracy gains when the depth increases. Classification tasks On CIFAR10 dataset ResNet with 20 layers give 8.75% error, with 32 layers give 7.51% error , with 44 layers give 7.17% error, with 56 layers give 6.97% error, with 110 layers give 6.43% error, with 1202 layers give 7.93% error.

**Exploring Over 1000 layers :** We explore an aggressively deep model of over 1000 layers. We set n = 200 that leads to a 1202-layer network. Our method shows no optimization difficulty, and this 1000 -layer network is able to achieve training error <0.1% . Its test error is still fairly good (7.93%). But there are still open problems on such aggressively deep models. The testing result of this 1202-layer network is worse than that of our 110-layer network. But combining with stronger regularization may improve results, which we will study in the future.