# Detail Model design

Deep learning Model optimization is one of the toughest tasks in the implementation of machine learning algorithms. There is a whole branch of deep learning and machine learning theory dedicated to hyperparameter optimization and tuning. Normally, we think about model optimization as a process of regularly making changes in the code of the model in order to minimize the testing loss. However, machine learning model or deep learning model optimization often refers to fine tuning elements that live outside the model but that can heavily influence its behavior of the model.

Hyperparameters are nobs of settings that can be turned up and down to control the behavior of a machine learning algorithm. Theoretically, hyperparameters can be considered orthogonal to the deep or machine learning model itself in the sense that, although they live outside the model, there is a direct relationship between one another.

Generally ,the classification of what defines a hyperparameter is highly abstract and flexible. Sure, there are a lot of well created hyperparameters such as the number of hidden units or the learning rate of a model but there are also a multiple number of settings that can play the role of hyperparameters for specific models. In general, hyperparameters are very specific to the type of machine learning model under optimization. Usually, a setting is modeled as a hyperparameter because it is not correct to learn it from the training set. A classical example are settings that control the capacity of a model (the range of functions that the model can represent). If a deep learning model learns those changes directly from the training set, then it is likely to try to optimize for that dataset which will cause the model to overfit( poor generalization).

Validation dataset plays an important part in the hyperparameter tuning as it helps validating the model and reduces the overfitting of the model.

The number and classes of hyperparameters is specific for each model in machine learning and deep learning. There are some classical hyperparameters that can be used to do general optimization.

· **Learning Rate:** it is the process of moniraterning the learning progress of a model.

· **Number of Hidden Units:** A classical way of optimizing the model is selecting the number of hidden units in a model.

· **Convolution Kernel Width:** In convolutional Neural Networks(CNNs), the Kernel Width influences the number of parameters in a model which in turns, changes its capacity and helps fast processing of the image.

Model selection

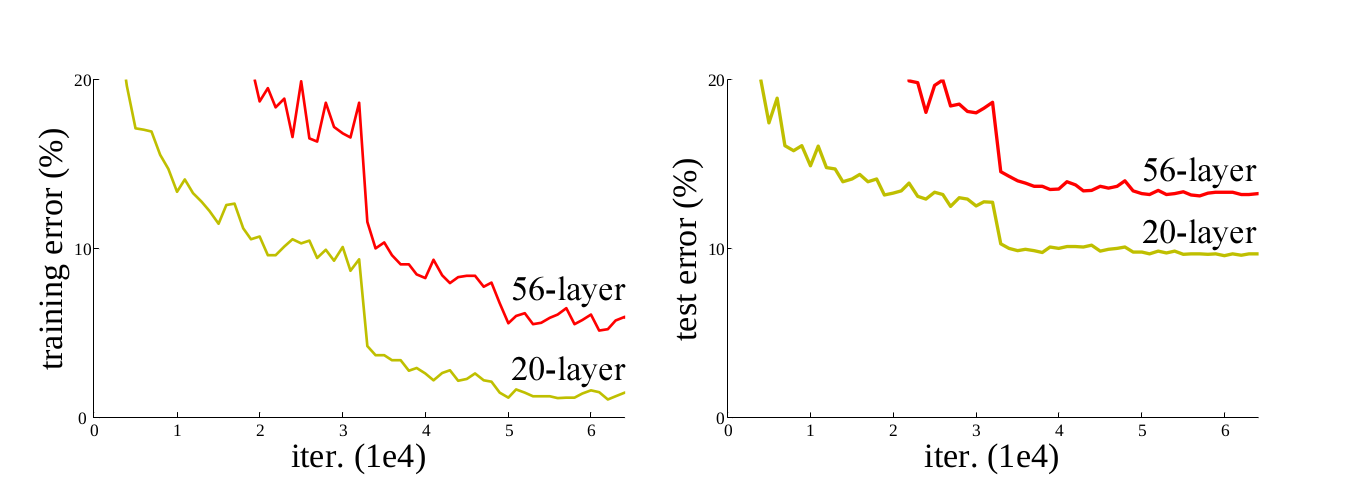
ResNet50

Deep Convolutional neural networks are really great at identifying varying range of features from the images and arranging more layers on top of each other generally gives us better results so a but getting better accuracy is not as easy as stacking up layers on top each other.

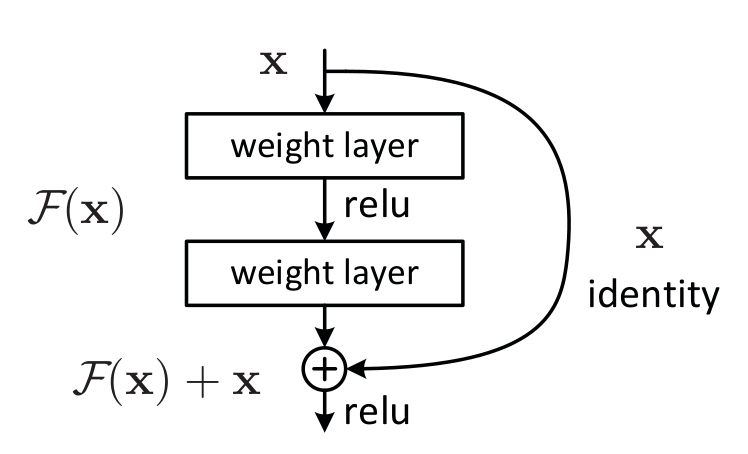
With stacking up layers result in the arises of a problem of vanishing/exploding gradients these problems were heavily handled by multiple ways and enabled networks with tens of layers to converge but when deep neural networks start converging it is evident to see another problem of the accuracy getting on a plateau and then lowering rapidly and this is not caused by overfitting as one may guess and adding more layers to a suitable deep learning model just increased the factor of training error.

This problem is rectified by taking a superficial model and a deep model that was constructed with the layers from the shallow model and adding identity layers to it and accordingly the deeper model shouldn't have produced any higher training error than its counterpart as the added layers were just the identity layers.

The problem was solved by making a superficial model and a deep model which was constructed using the layers of the superficial model and adding unitary layers to it and respectively the deeper model should not have produced any significant inflation in the training error then its counterpart as the added layers are normal identity layer of superficial model.



Authors of the Resnet models addressed this problem by creating a shortcut connection that normally performs identity mapping this framework is called residual learning framework.



Generally, the layers fit a residual mapping explicitly which is denoted by H(x) and another mapping F(x):= H(x)-x fit the nonlinear layers so the actual mapping becomes.

H(x) := f(x)+a which can be seen in fig2

And as a result of shortcut identity mapping is that no additional parameters added to the modeland the computation is also remains in check

Comparison

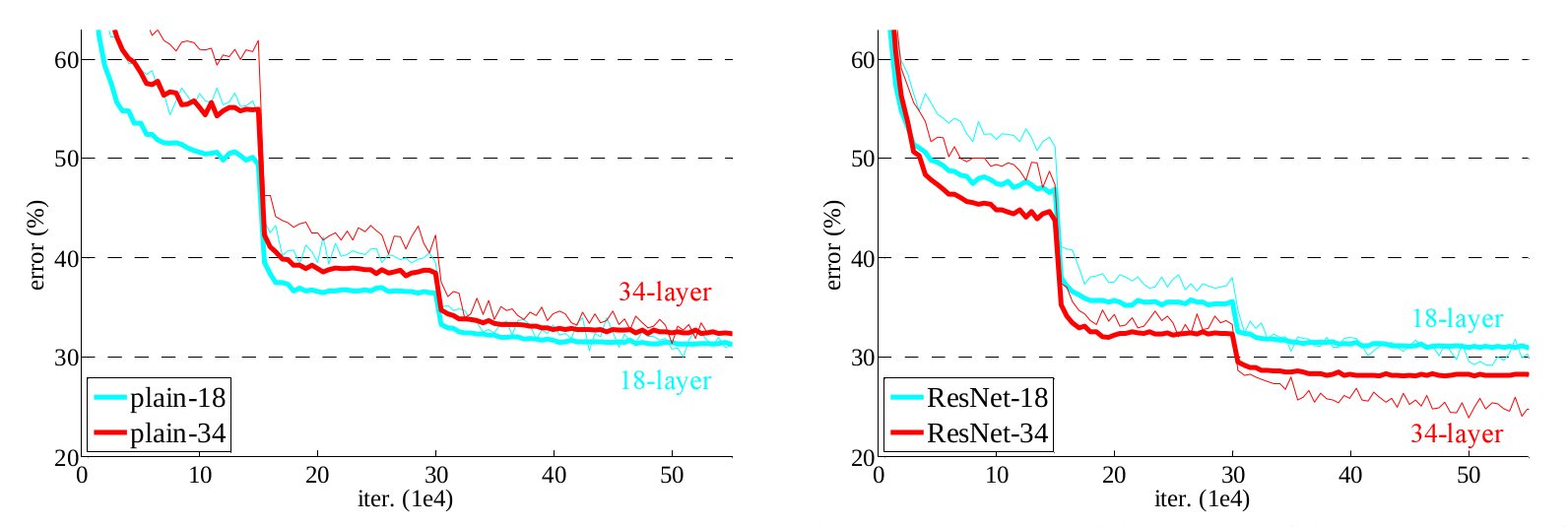
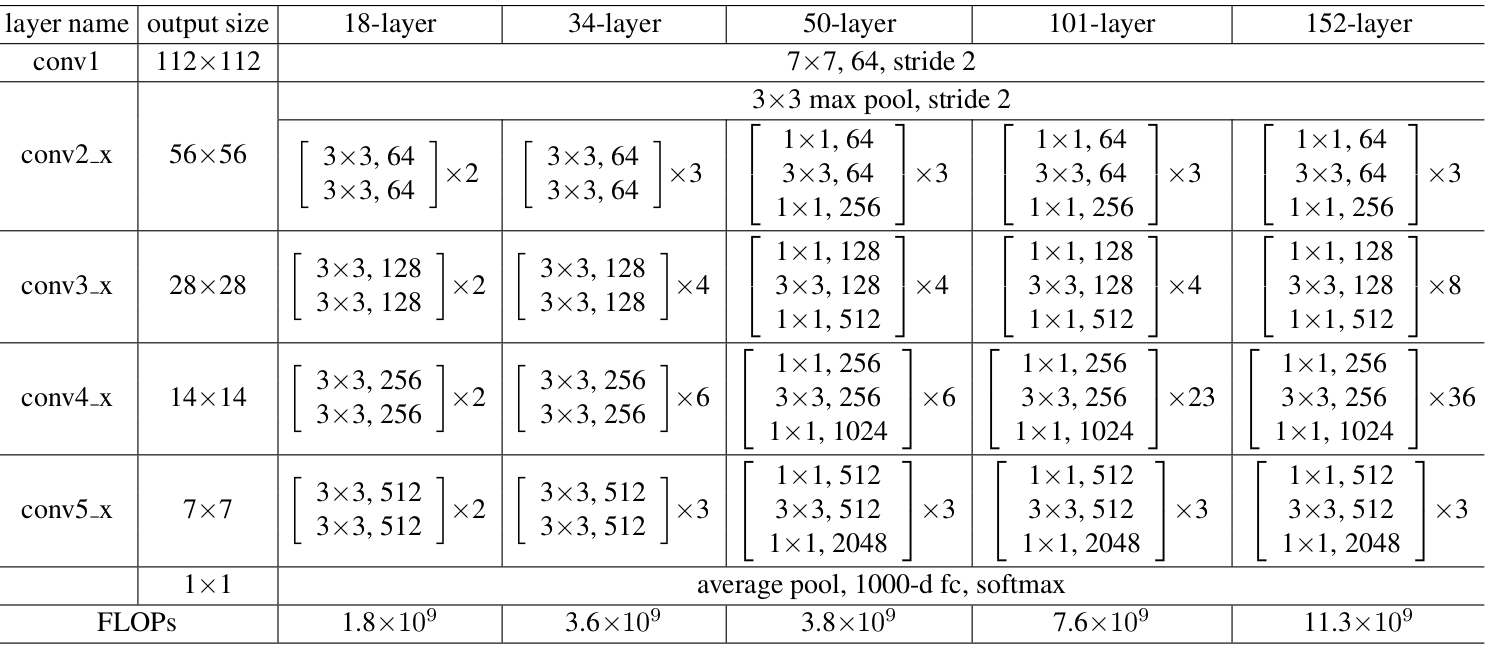


Figure 3 is the demonstration of how a 34 layer pain model is compared against a 18 layer residual network ResNet. And the results are astounding the 18 layer residual network outperformed the 18 layer plain network. Thus stating the fact that increasing the layers does not necessarily increase the accuracy.

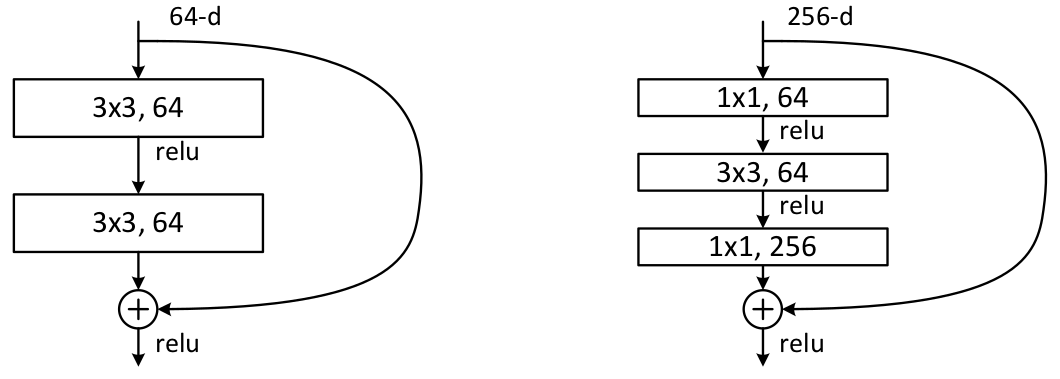
## 

## **Architecture**



Now we are going to discuss about Resnet 50 and also the architecture for the above talked 18 and 34 layer ResNet is also given residual mapping and not shown for simplicity.

There was a small change that was made for the ResNet 50 and above that before this the shortcut connections skipped two layers but now they skip three layers and also there was 1 \* 1 convolution layers added that we are going to see in detail with the ResNet 50 Architecture.



So as we can see in the table 1 the resnet 50 architecture contains the following element:

* A convoultion with a kernel size of 7 \* 7 and 64 different kernels all with a stride of size 2 giving us **1 layer**.
* Next we see max pooling with also a stride size of 2.
* In the next convolution there is a 1 \* 1,64 kernel following this a 3 \* 3,64 kernel and at last a 1 \* 1,256 kernel, These three layers are repeated in total 3 time so giving us **9 layers** in this step.
* Next we see kernel of 1 \* 1,128 after that a kernel of 3 \* 3,128 and at last a kernel of 1 \* 1,512 this step was repeated 4 time so giving us **12 layers** in this step.
* After that there is a kernal of 1 \* 1,256 and two more kernels with 3 \* 3,256 and 1 \* 1,1024 and this is repeated 6 time giving us a total of **18 layers**.
* And then again a 1 \* 1,512 kernel with two more of 3 \* 3,512 and 1 \* 1,2048 and this was repeated 3 times giving us a total of **9 layers**.
* After that we do a average pool and end it with a fully connected layer containing 1000 nodes and at the end a softmax function so this gives us **1 layer**.

We don't actually count the activation functions and the max/ average pooling layers.

so totaling this it gives us a 1 + 9 + 12 + 18 + 9 + 1 = 50 layers Deep Convolutional network.

The Result were pretty good on the ImageNet validation set, The ResNet 50 model achieved a top-1 error rate of 20.47 percent and and achieved a top-5 error rate of 5.25 percent, This is reported for single model that consists of 50 layers not a ensemble of it. below is the table given if you want to compare it with other ResNets or with other models.

## **Uses**

* This architecture can be used on computer vision tasks such as image classififcation, object localisation, object detection.
* and this framework can also be applied to non computer vision tasks to give them the benifit of depth and to reduce the computational expense also.

