ML

LECTURE-24

BY
Dr. Ramesh Kumar Thakur
Assistant Professor (II)
School Of Computer Engineering

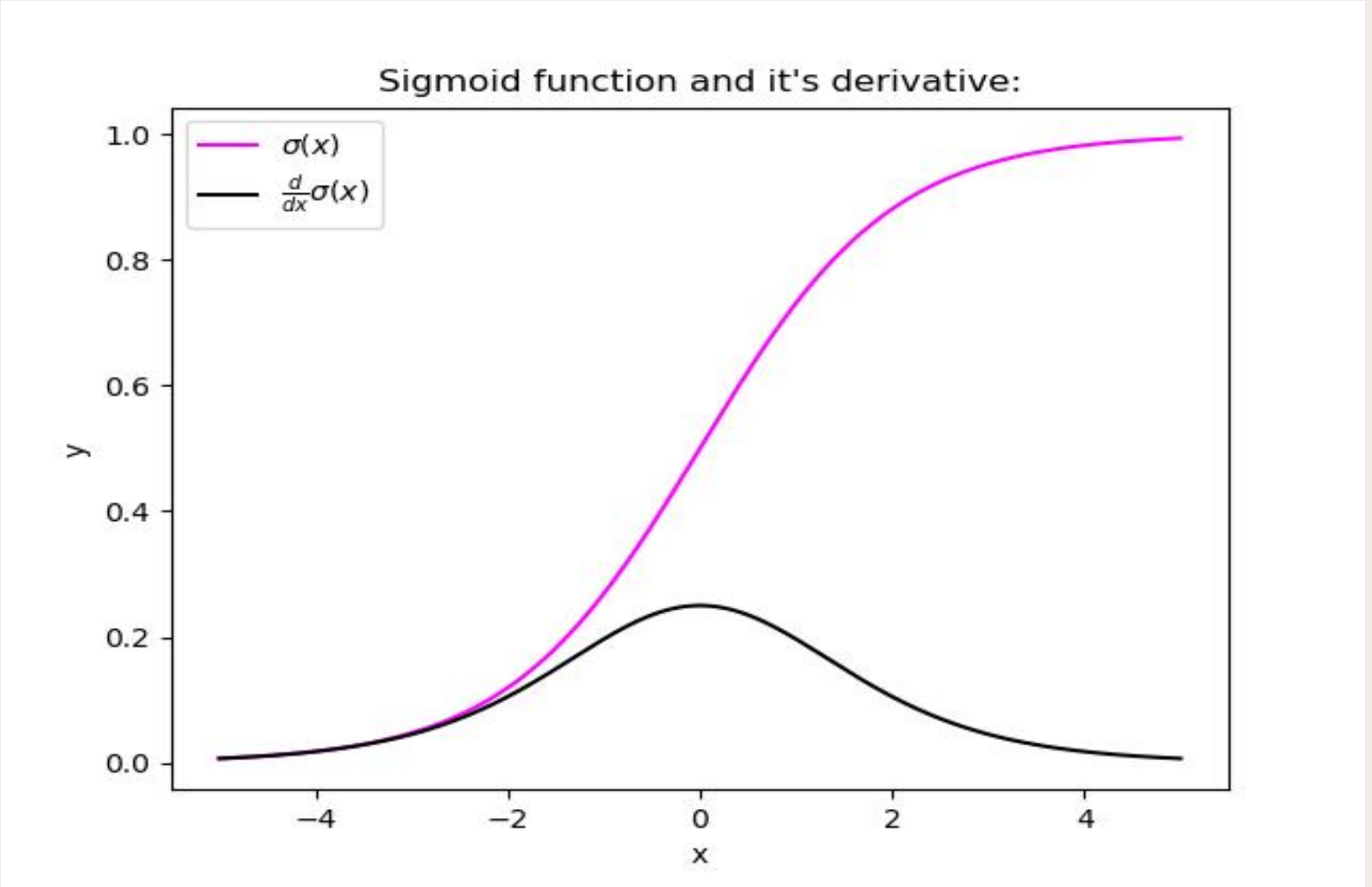
Understanding the Vanishing and Exploding gradient Problems

- **Vanishing gradients problem**
- As the backpropagation algorithm advances downwards(or backward) from the output layer towards the input layer, **the gradients often get smaller and smaller and approach zero which eventually leaves the weights of the initial or lower layers nearly unchanged.**
- As a result, the gradient descent never converges to the optimum.
- This is known as the **vanishing gradients problem.**

- **Exploding gradients problem**
- In some cases, the **gradients keep on getting larger and larger as the backpropagation algorithm progresses.**
- This, in turn, **causes very large weight updates and causes the gradient descent to diverge.**
- This is known as the **exploding gradients problem.**

Why Do the Gradients Even Vanish/Explode?

- Certain activation functions, like the **logistic function (sigmoid), have a very huge difference between the variance of their inputs and the outputs.**
- In simpler words, **they shrink and transform a larger input space into a smaller output space that lies between the range of [0,1].**



Why Do the Gradients Even Vanish/Explode?

- Observing the above graph of the Sigmoid function, we can see that **for larger inputs (negative or positive), it saturates at 0 or 1 with a derivative very close to zero.**
- Thus, when the backpropagation algorithm chips in, it virtually has **no gradients to propagate backward in the network, and whatever little residual gradients exist keeps on diluting as the algorithm progresses down through the top layers.**
- So, this **leaves nothing for the lower layers.**
- Similarly, in **some cases suppose the initial weights assigned to the network generate some large loss.**
- Now **the gradients can accumulate during an update and result in very large gradients which eventually results in large updates to the network weights and leads to an unstable network.**
- The **parameters can sometimes become so large that they overflow and result in NaN values.**

Signs of Exploding/Vanishing Gradient Problem

- Following are some signs that can indicate that our gradients are exploding/vanishing :

Exploding	Vanishing
There is an exponential growth in the model parameters.	The parameters of the higher layers change significantly whereas the parameters of lower layers would not change much (or not at all).
The model weights may become NaN during training.	The model weights may become 0 during training.
The model experiences avalanche learning.	The model learns very slowly and perhaps the training stagnates at a very early stage just after a few iterations.

How to avoid Exploding/Vanishing Gradient Problem

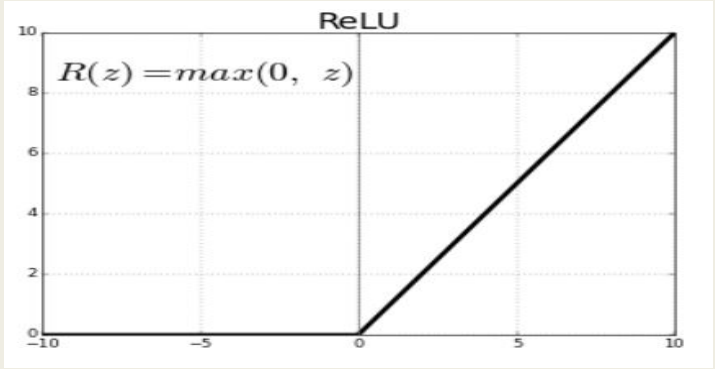
- **Proper Weight Initialization**
- For the proper flow of the signal:
- The **variance of outputs of each layer should be equal to the variance of its inputs.**
- The **gradients should have equal variance before and after flowing through a layer in the reverse direction.**
- To satisfy both the conditions, Xavier Glorot et al. proposed an initialization method known as **Xavier initialization** (after the author's first name) or **Glorot initialization** (after his last name).
- **Uniform Xavier Initialization**
- We can initialize the weights by drawing them from a **random uniform distribution within a specific range, which is determined by the formula:**

$$x = \sqrt{\frac{6}{n_{inputs} + n_{outputs}}}$$

- x is calculated using above formula.
- n_{inputs} : Number of Input in the input layer
- n_{output} : Number of Output in the Output layer
- For each weight in network, we draw a random value w from a uniform distribution in the range [-x, x].

How to avoid Exploding/Vanishing Gradient Problem

- **Using Non-saturating Activation Functions**
- we observed that sigmoid activation function's **nature of saturating for larger inputs (negative or positive) came out to be a major reason behind the vanishing of gradients** thus making it non-recommendable to use in the hidden layers of the network.
- So to **tackle the issue regarding the saturation of activation functions like sigmoid and tanh, we must use some other non-saturating functions like ReLu and its alternatives.**



- Unfortunately, the **ReLu function is also not a perfect pick for the intermediate layers of the network "in some cases"**.
- It suffers from a problem known as **dying ReLus wherein some neurons just die out, meaning they keep on throwing 0 as outputs with the advancement in training.**
- Some popular alternative functions of the ReLU that mitigates the problem of vanishing gradients when used as activation for the intermediate layers of the network are LReLU, PReLU, ELU, SELU.

How to avoid Exploding/Vanishing Gradient Problem

- Using **He initialization along with any variant of the ReLU activation function can significantly reduce the chances of vanishing/exploding problems** at the beginning. However, it does not guarantee that the problem won't reappear during training.
- In **2015, Sergey Ioffe and Christian Szegedy proposed a paper in which they introduced a technique known as Batch Normalization to address the problem of vanishing/exploding gradients.**
- The **Following key points** explain the intuition behind BN and how it works:
- It consists of **adding an operation in the model just before or after the activation function of each hidden layer.**
- **This operation simply zero-centers and normalizes each input, then scales and shifts the result using two new parameter vectors per layer: one for scaling, the other for shifting.**
- In other words, the **operation lets the model learn the optimal scale and mean of each of the layer's inputs.**
- **To zero-center and normalize the inputs, the algorithm needs to estimate each input's mean and standard deviation.**
- **It does so by evaluating the mean and standard deviation of the input over the current mini-batch (hence the name "Batch Normalization").**

How to avoid Exploding/Vanishing Gradient Problem

- **Gradients clipping**
- To **prevent gradients from exploding, one of the most effective ways is gradient clipping.**
- In a nutshell, gradient clipping **caps the derivatives to a threshold and uses the capped gradients to update the weights throughout.**

- **L2 norm regularization**
- In addition to weight initialization, another excellent approach is employing L2 regularization, which **penalizes large weight values by imposing a squared term of model weights to the loss function:**

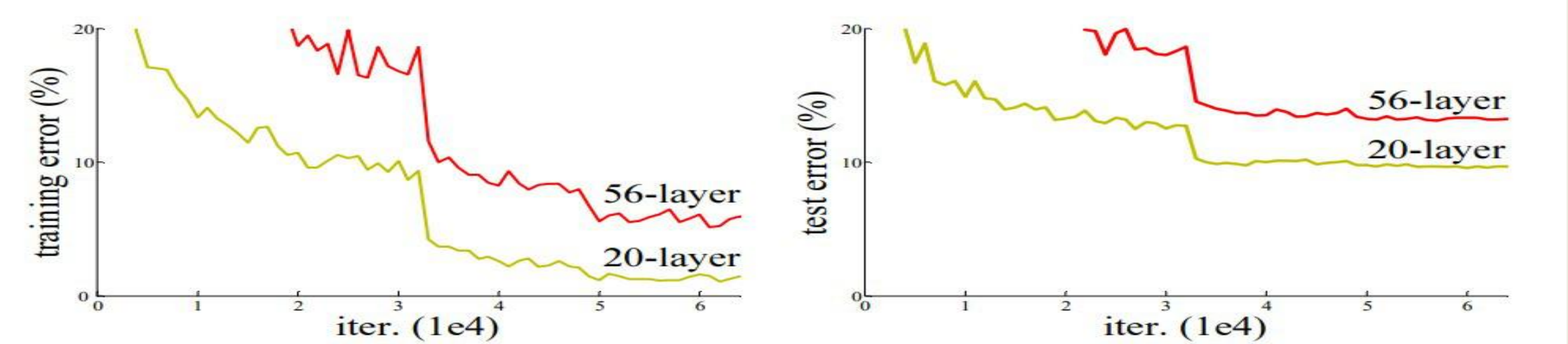
$$Loss = Error(Y - \hat{Y}) + \lambda \sum_{1}^{n} w_i^2$$

- **Adding the L2 norm oftentimes will result in smaller weight updates throughout the network.**

- **Using less number of layers**
- Using fewer layer will ensure that the **gradient is not multiplied too many times.**
- This **may stop the gradient from vanishing or exploding, but it affect the ability of network to understand complex features.**

Problem with deep neural networks

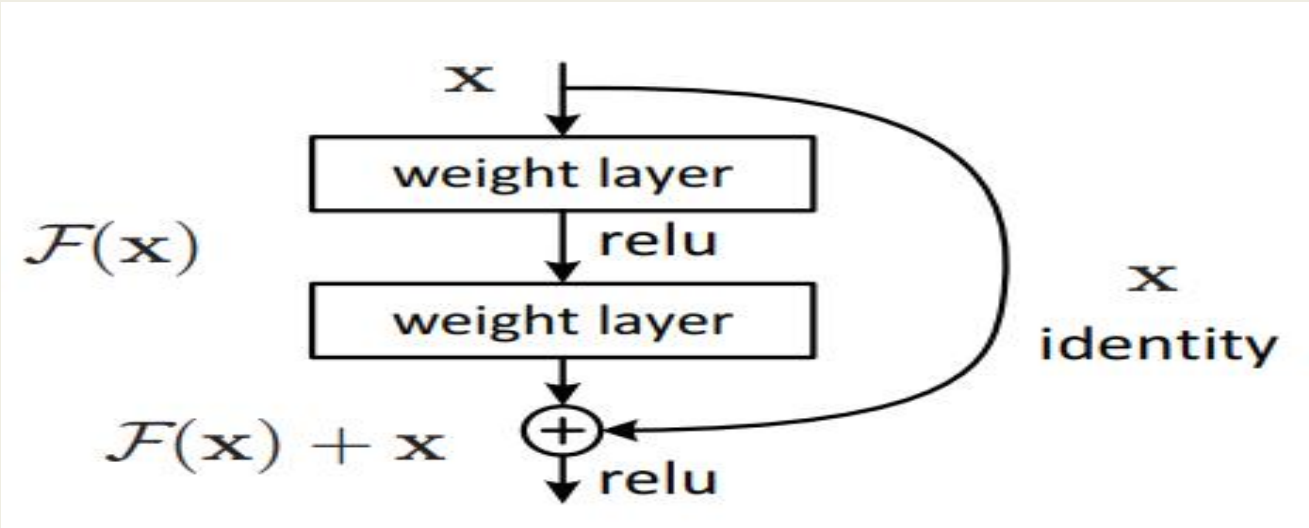
- After the **first CNN-based architecture (AlexNet) that win the ImageNet 2012 competition, Every subsequent winning architecture uses more layers in a deep neural network to reduce the error rate.**
- This works for less number of layers, but **when we increase the number of layers, there is a common problem in deep learning associated with that called the Vanishing/Exploding gradient.**
- This causes the gradient to become 0 or too large.
- Thus when we increases number of layers, the training and test error rate also increases



- In the above plot, we can observe that a **56-layer CNN gives more error rate on both training and testing dataset than a 20-layer CNN architecture.**
- After analyzing more on error rate the authors were able to reach conclusion that it is **caused by vanishing/exploding gradient.**

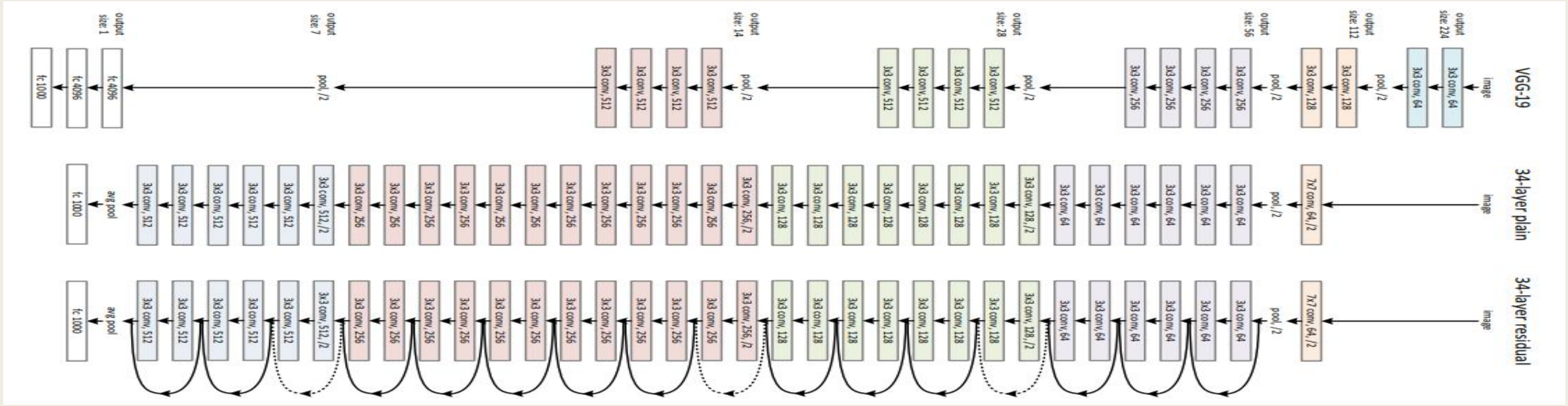
Residual Network (ResNet)

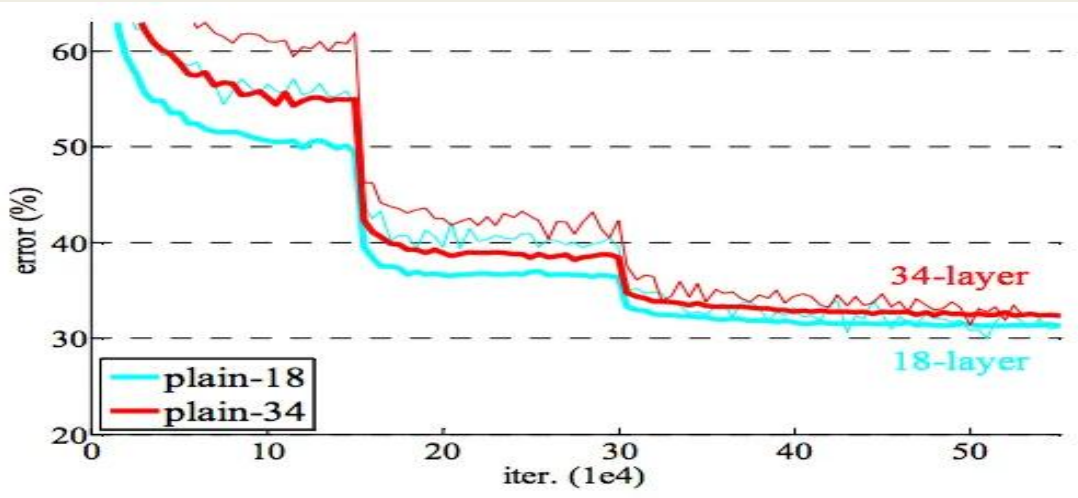
- In order to solve the problem of the vanishing/exploding gradient, **Microsoft Research introduced a new architecture called Residual Network (ResNet) in 2015.**
- In this network, we use a technique called **skip connections.**
- The **skip connection connects activations of a layer to further layers by skipping some layers in between.** This forms a **residual block.**
- Resnets are made by **stacking these residual blocks together.**
- The approach behind this network is **instead of layers learning the underlying mapping, we allow the network to fit the residual mapping.**
- So, instead of say H(x), initial mapping, let the network fit,
- F(x) := H(x) - x which gives H(x) := F(x) + x.

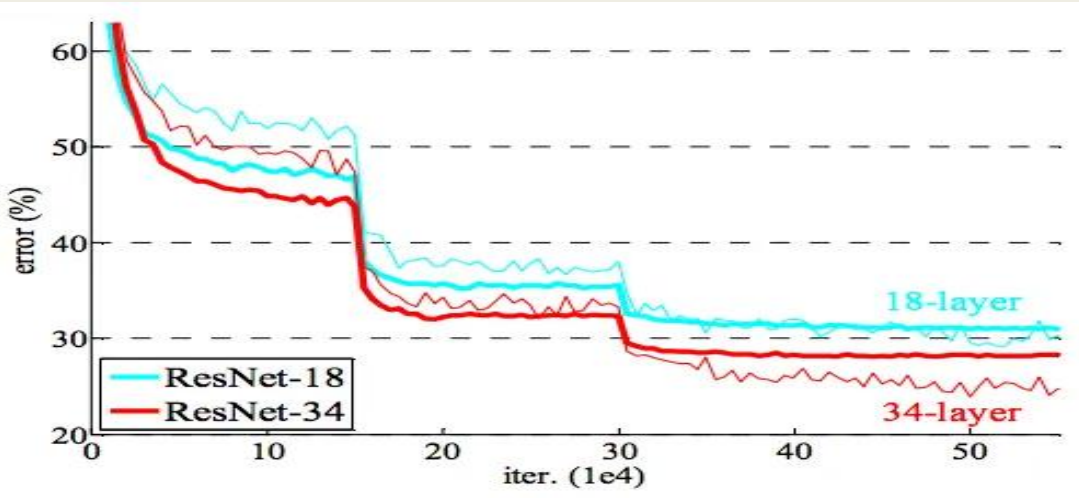


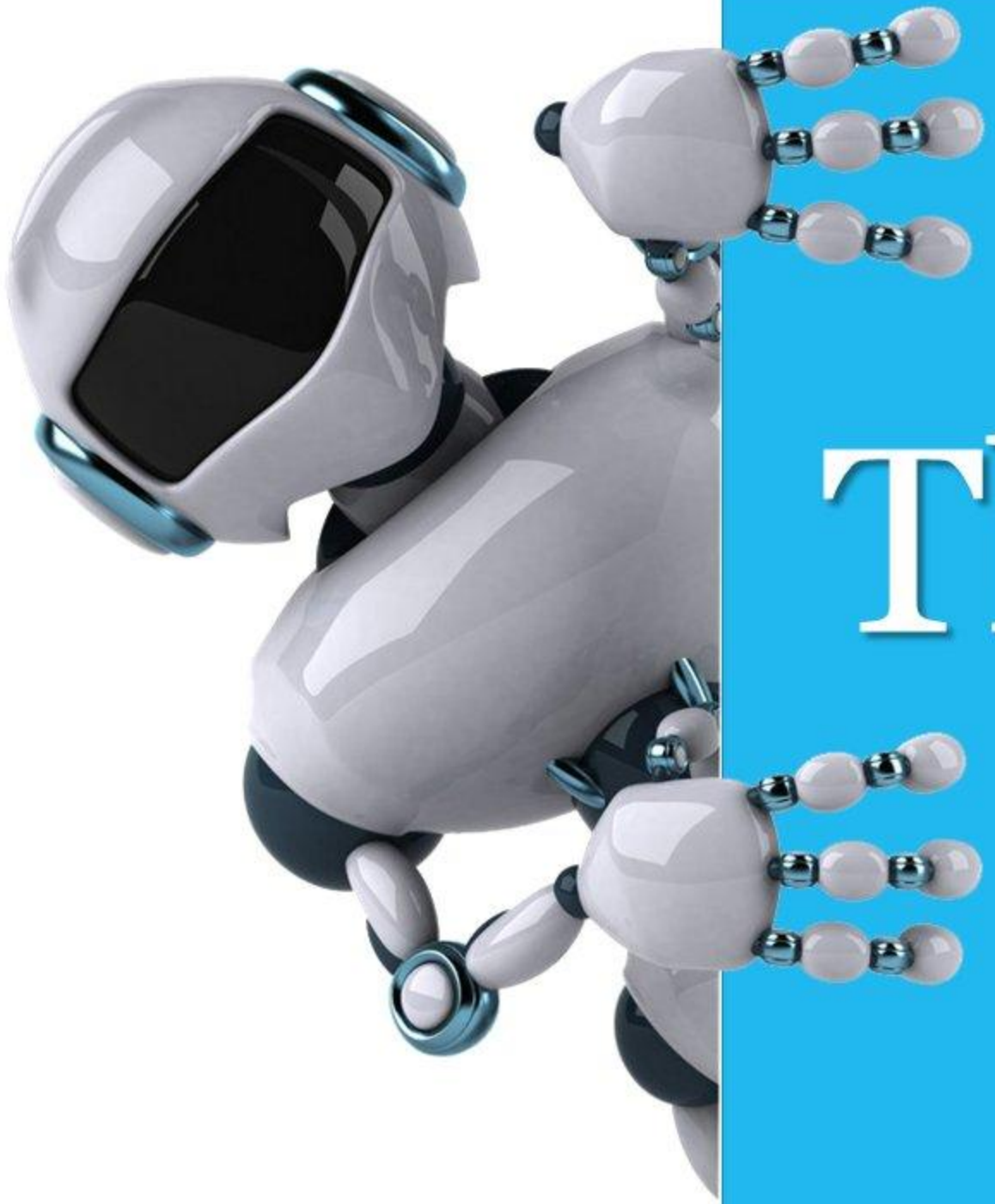
Residual Network (ResNet)

- The advantage of adding this type of skip connection is that **if any layer hurt the performance of architecture then it will be skipped by regularization.**
- This results in training a very deep neural network without the problems by vanishing/exploding gradient.









Thank you