ResNet

By 2015, researchers knew that deeper models used to be better but as they went on to train deeper neural networks (>20 layers) something strange started happening. Training error started to increase and networks began to "degrade", not because of overfitting, but because optimization itself became harder.

ResNet solved the **Degradation Problem**

Let's take a simple example:

⇒ Suppose we add more layers to a well-performing shallow CNN. ⇒ The deeper model should perform at least as well (since it can learn identity mappings), right?

⇒ But in practice – it was performing worse!

Even with ReLU, Batch Norm, and good initialization – the deeper network cannot optimize properly. Training and validation accuracy both degrade.

This wasn't caused by vanishing gradients (ReLU already helped with that), but rather by gradient attenuation and difficulty learning identity mappings through multiple non-linear transformations.

* The innovation ResNet (Residual Networks) / Residual Block ("skip connections / shortcut connections")

The ResNet paper published in 2015 introduced the idea of Residual Block – a simple but revolutionary idea that enabled training of extremely deep neural networks (upto 152 layers). It won the famous ImageNet challenge with an top-5 error of 3.57%, outperforming VGGNet, GoogleNet, etc.

⇒ In a standard CNN layer stack, we try to learn a direct mapping $H(x)$ from input x to output $H(x)$.

⇒ ResNet let the layers learn the difference, or **residual**, instead. This was achieved with a "skip connection" (or "shortcut connection")!

Here's how it works:

1. Traditional Block ⇒ A plain network block takes an input x and tries to learn a target function $H(x)$.

$$\text{output} = H(x)$$

2. Residual Block ⇒ A ResNet block takes an input x and passes it through its layers like (Conv-BN-ReLU) to learn a function $F(x)$. It then adds the original input x to the output of this block before the final activation.

$$\text{output} = H(x) = F(x) + x$$

So, the input x is added (via shortcut connection) to the output of the stacked layers $F(x)$.

* Why is this so revolutionary?

Think of the "degradation" problem, where adding a new layer was hurting performance. The ideal, safest thing for a new layer to do is nothing (i.e., just pass the input through, $H(x) = x$).

⇒ In a plain network, to learn $H(x) = x$, the layers must learn to make their weights act as an identity matrix. This is non-trivial and very difficult for the optimizer.

⇒ In a residual network, to learn $H(x) = x$, the network just need to learn $F(x) = 0$. The final output becomes $0 + x = x$.

It is exponentially easier for a network to learn to set its weights zero than to learn an identity mapping.

How this helped (The Impact)

This one simple idea ($F(x) + x$) had two massive effects:

(1) Solved the Degradation problem: If a new layer was not helpful, the optimizer could simply set its weights to zero ($F(x) = 0$). The "skip connection" would ensure the original information from 'x' passed through unharmed. This meant we could add layers without the risk of hurting performance.

2. Solved the Vanishing Gradient problem: The skip connection acts as an "information highway." During backpropagation the gradient can flow back directly through the +x paths, bypassing the weights layers. This ensures a strong, non-vanishing gradient signal reaches even the earliest layers of the network, allowing them to learn effectively.

With these problems solved, the ResNet team was able to train models like ResNet-50, ResNet-101, ResNet-152 and show that deeper models consistently and significantly outperformed shallower ones.