# CMP_SC 8770: Neural Networks

# “Comparative Study between Convolution Neural Networks and Deep Residual Networks”

**Farhan Quadir**

***May 13, 2019***

**Department of Electrical Engineering and Computer Science (EECS), University of Missouri- Columbia**

## Overview

- Deep Neural Networks are difficult to train
- Stacking more layers tend to introduce vanishing/exploding gradients due to squashing which can lead to slower or no convergence. (Can be solved by Normalization)
- Deeper Networks lead to higher training error which leads to higher testing error, so introduces degradation in the accuracy.
- The degradation problem is addressed by introducing a deep residual architecture.
- Adding the input information to the output of a hidden layer neuron (shortcut connections) via identical mapping prevents loss of input information through layers. This does not introduce extra parameters not does it require more computation.
- I have run separate experiments using convolutional neural networks (CNN) and deep residual networks (ResNet) on MNIST and CIFAR10 datasets to compare and contrast the power of ResNets over CNNs.
- It was found that Deep ResNets with same number of layers as that in CNNs, outperforms CNNs by far with respect to accuracy, and convergence speed.

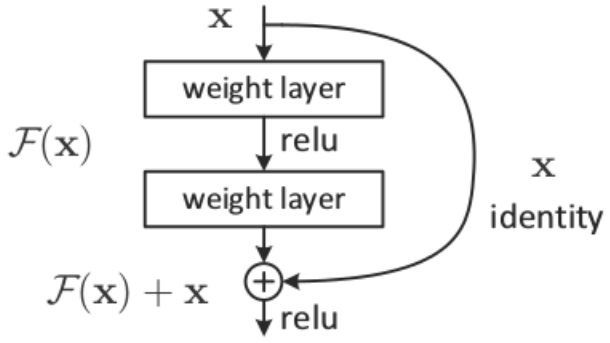


Figure 1. Adapted from original ResNet paper. Shows how the input information is propagated in the forward pass.

## Methodology: Part 1

The shortcut output for the neuron $y_l$ is given by the expression

$$y_l = x_l + F(x_l, W_l) \qquad \textbf{(1)}$$

where $x_l$ is the input and $F(x_l, W_l)$ is the convolution output where $W_l$ is the weight vector.

$$\text{So } F(x_l, W_l) = (W_l)^T x_l + b_l \qquad \textbf{(2)}$$

Input for the next neuron is $x_{l+1} = f(y_l)$ where the function *f* is the activation function (Relu in our case). Since *f* is Relu, $f(y_l) = y_l$

Therefore, we can rewrite equation (1) as

$$x_{l+1} = x_l + F(x_l, W_l) \qquad \textbf{(3)}$$

So for any network of depth *L*

$$x_L = x_{L-1} + F(x_{L-1}, W_{L-1}) = x_{L-2} + F(x_{L-2}, W_{L-2}) + F(x_{L-1}, W_{L-1}) = x_1 + F(x_1, W_1) + F(x_2, W_2) + \ldots + F(x_{L-1}, W_{L-1}) \qquad \textbf{(4)}$$

$$\text{or, } x_L = x_1 + \sum_{i=1}^{L-1} F(x_i, W_i) \qquad \textbf{(5)}$$

For any neuron $x_l$ the derivative for equation (5) becomes $\frac{\partial x_L}{\partial x_l} = 1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i, W_i)$ While for CNNs, the output is given by products of the nodes in the layers. $y_l = \prod_{i=0}^{L-1} W_i x_i$

As a result, the backpropagation

$$\frac{\partial \mathcal{E}}{\partial W_i} = \frac{\partial \mathcal{E}}{\partial x_L} \frac{\partial x_L}{\partial x_l} \frac{\partial x_l}{\partial W_i} \qquad \textbf{(6)}$$

Becomes,

$$\frac{\partial \mathcal{E}}{\partial W_i} = \frac{\partial \mathcal{E}}{\partial x_L} [1 + \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i, W_i)] \frac{\partial x_l}{\partial W_i} \qquad \textbf{(7)}$$

From equation (7) we can see that the term $\frac{\partial \mathcal{E}}{\partial x_L}$ propagates information directly using the “shortcuts” so full gradient is passed back to the first layer without gradient vanishing, while the term $\frac{\partial \mathcal{E}}{\partial x_L} \frac{\partial}{\partial x_l} \sum_{i=1}^{L-1} F(x_i, W_i) \frac{\partial x_l}{\partial W_i}$ propagates through the weight layers.

## Methodology (Continued): The Deep CNN Architecture

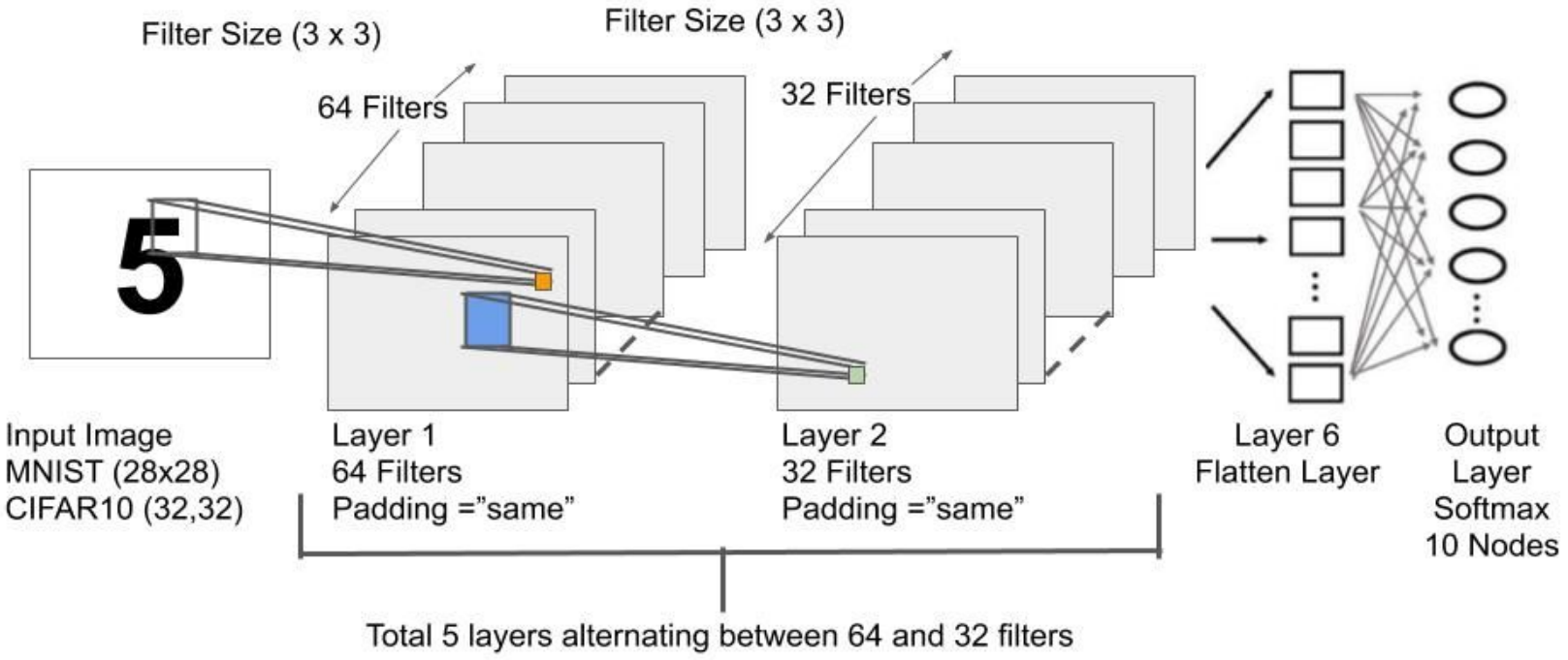


Figure 2. Network Architecture of Deep Convoluted Neural Network (CNN). The input image is a 28 x 28 (mnist) or 32 x 32 (cifar10) matrix. There are five convolution layers that alternate between 64 and 32 filters produced by kernel size 3 x 3. After the final convolution, the network is flattened and a dense layer of ten output nodes are added for the ten classes. The final layer performs a softmax to obtain the final output class.

Table1: Table showing training strategy used- the different parameter settings during training as as follows.

| **Parameter** | **Usage/Value** | **Parameter** | **Usage/Value** |
|---|---|---|---|
| Loss function | Categorical cross entropy | MNIST training dataset | 60000 |
| Optimizer | Adam ($\beta_1$=0.9, $\beta_1$=0.999) | MNIST Testing dataset | 10000 |
| Learning rate | 0.001 (fixed) | CIFAR10 Training dataset | 50000 |
| Normalization | None | CIFAR10 Testing dataset | 10000 |

## Methodology (Continued): The Deep ResNet Architecture

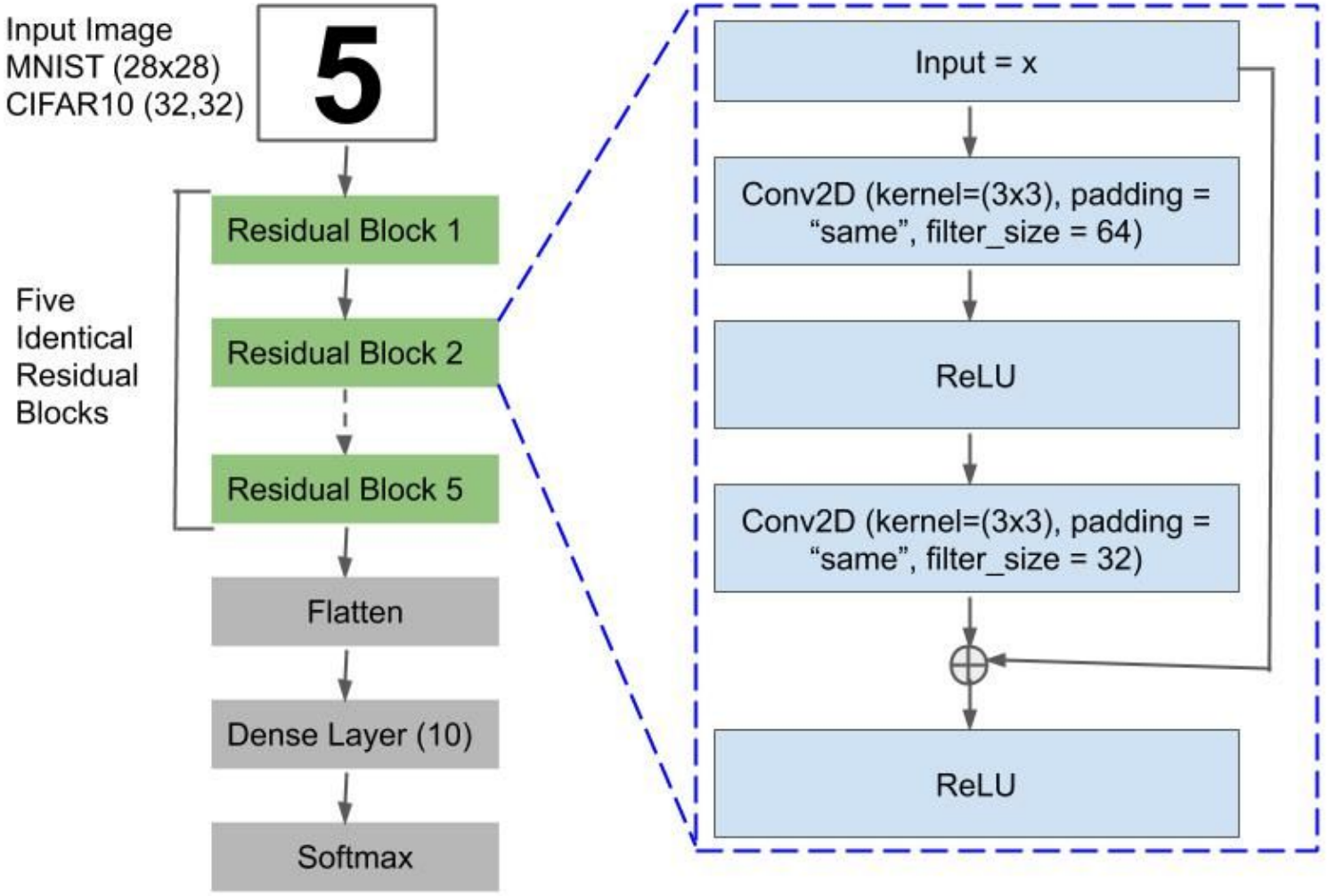


Figure 3. Deep Residual Architecture containing five identical Residual Blocks. The input image is a 28 x 28 (mnist) or 32 x 32 (cifar10) matrix. Within each residual block there are two convolution layers that alternate between 64 and 32 filters produced by kernel size 3 x 3. There is ReLU activation in between the convolution layers. After the final convolution, the input (x) is added to the output followed by another ReLU activation. After the final residual block, the network is flattened and a dense layer of ten output nodes are added for the ten classes. The final layer performs a softmax to obtain the final output class.

# Results

**(a) CNN** **(b) ResNet**

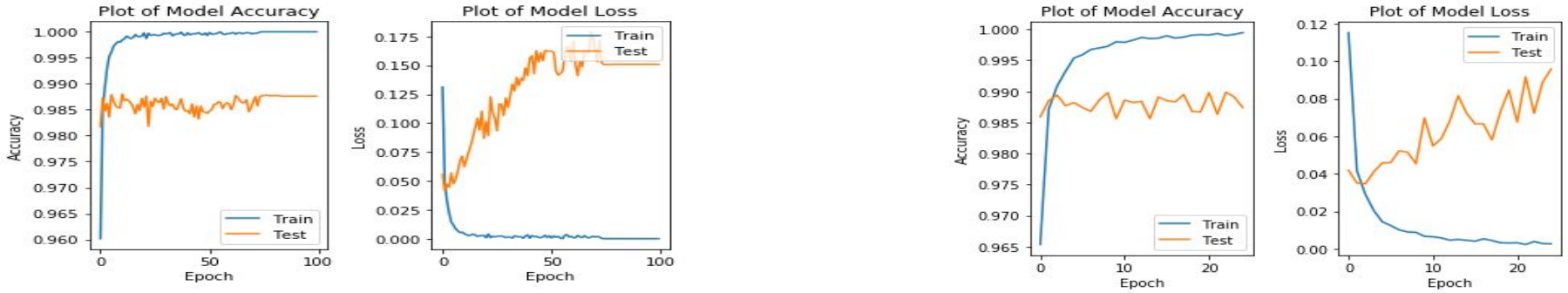


Figure 4. Figure showing comparison of accuracy (left) and loss (right) between the two networks for MNIST data. (a) shows the performance of CNN while (b) shows the performance of ResNet. It can be seen that ResNet outperforms CNN overwhelmingly by converging within ten epochs with extremely high validation accuracy close to 0.99 while CNN converges at epoch 25 also with pretty high validation accuracy of 0.985.

**(a) CNN** **(b) ResNet**


Figure 4. Figure showing comparison of accuracy (left) and loss (right) between the networks for CIFAR10. (a) shows information on CNN while (b) shows the performance of ResNet. It can be seen that ResNet slightly outperforms CNN by converging within 50 epochs with extremely high test accuracy close to 1.00 while CNN converges at epoch 80 with pretty high test accuracy but with a lower validation accuracy of around 0.55 less than that of ResNet which is slightly above 0.60.

## Summary and Future Work

Table: Summarizes the Final Accuracies between the two Networks

| | Training Accuracy | | Test Accuracy | |
|---|---|---|---|---|
| **Dataset** | **CNN** | **ResNet** | **CNN** | **ResNet** |
| **MNIST** | 0.9977 | 0.9994 | 0.9848 | 0.9874 |
| **CIFAR10** | 0.9882 | 0.98926 | 0.5569 | 0.6067 |

- Residual Network outperformed CNN.
- Due to GPU limitations unable to test deeper networks.
- ResNets are an overkill for MNIST but works only slightly better with CIFAR10
- Due to not using normalization, probably we experienced vanishing gradient problem with CIFAR10 data.
- Future Work can be to try deeper networks like 152 or 1000 layers on more memory GPUs
- We can try out different normalization techniques and optimizers.
- Also we can train with other datasets like CIFAR100 or ImageNet.
- We can actually try to monitor gradients within the layers as we backpropagate to monitor actually how ResNet effects gradient values.