Artificial Neural Networks
Residual Networks

Mayukh Sattiraju
msattir@clemson.edu

ECE 872
Takehome1 Report

March 6, 2018

Residual MLFF Networks
ECE 872 Takehome 1 Proposal

Mayukh Sattiraju

February 9, 2018

1 Proposal

For this assignment, I intend to implement Residual Feed Forward Networks [2] and explore the benefits of this architecture over the conventional architecture. I would also like to show that training an ensemble of networks is more fruitful than training a single network.

2 Residual Networks

Residual Networks are Feed forward Networks that have one additional change. They include "skip" connections from one layer to another. This is depicted in figure 1.

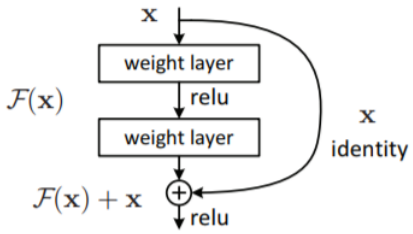


Figure 1: Residual Network Building block

This idea was first proposed by this paper https://arxiv.org/pdf/1512.03385.pdf. The purpose of these skip connections, according to the paper, was to fix vanishing gradients during training. The paper also claims the architecture would optimize better than the conventional MLFF and produce lower training errors.

I intend to explore situations where the gradients in a conventional MLFF vanish and compare how the Residual Network with skip connections alleviates this problem. Also as the new network has a slightly different architecture than conventional MLFFs; I would modify the Back Propagation algorithm accordingly to correctly propagate the error residuals. A quantitative measure of the working of these Residual Networks would be a difference in the training error for a network with these "skip" connections as compared with a conventional network with regular connections. The Residual network should ideally have a lesser training error.

I would also like to show that to train networks, an ensemble of many networks, combined intelligently would produce a better and/or faster training method than compared to training a single network.

1 Introduction

This assignment deals with the application of Deeper Architectures of Feed Forward Networks to create a better mapping for a given set of inputs to the corresponding outputs. The network is also able to create mappings for sets of inputs that it is not 'trained' on.

This paper tries to create a mapping for a 4 bit Analog to Digital converter. So for a normalized analog input the network is able to produce a corresponding 4-bit mapping.

The Network is trained on a set of 15 values and tested with a different set of 15 input values. So in effect the training:testing ratio is a 50:50 ratio.

Finally errors from the neural network are reported as the deviation of the generated output to the expected value. The error/deviations are reported on a per sample basis.

A 2 layer model is used as the reference benchmark. It has been shown that deeper networks have more 'degrees of freedom' and hence create better mappings [Reference material]. First, 'deeper' models such as 4, 5 and 6 layered Feed Forward networks are trained to achieve the ideal mapping. It is almost immediately found that 'deeper' Feed Forward models generate worse mappings as the deltas propagated diminish and provide no useful 'delta' henceforth, resulting in inactive neurons.

To alleviate the problems of inactive neurons (caused by diminishing gradients) Residual blocks with jump ahead connections are explored [reference ResNet paper]. These networks, in general, produce better mappings than those produced with conventional Feed Forward networks.

The architecture of these Residual Blocks are inherited from the 2015 Paper. But, to this implementation a simple modification is explored. Then this is taken a step further by parameter-izing these jump ahead connections.

The comparision of performances of these models on the A-D mapping and (other mapping) are shown as a part of the Results of this paper.

2 Methods

This section describes the various Network architectures employed to create the mapping with the least error/deviation from the expected output.

2.1 Conventional Feed Forward Networks

Feed Forward Networks use the weights and biases as the 'degrees of freedom' that get tuned to find the best mapping with the least error. So we build on a 2 layer Network that has about 300 parameters (287 weights and 26 biases).

A 3 layer Network with an connections similar to the Conventional MLFF would have about parameters, and similarly a 4 and a 5 layered network would have (somanny parameters) respectively. The addition of these parameters gives the network additional degrees of freedom to obtain a better mapping.

The structure and the update equations for a 5 layered Feed Forward Network with sigmoid activation functions and Gradient Descent as the optimization function is shown in figure 1.

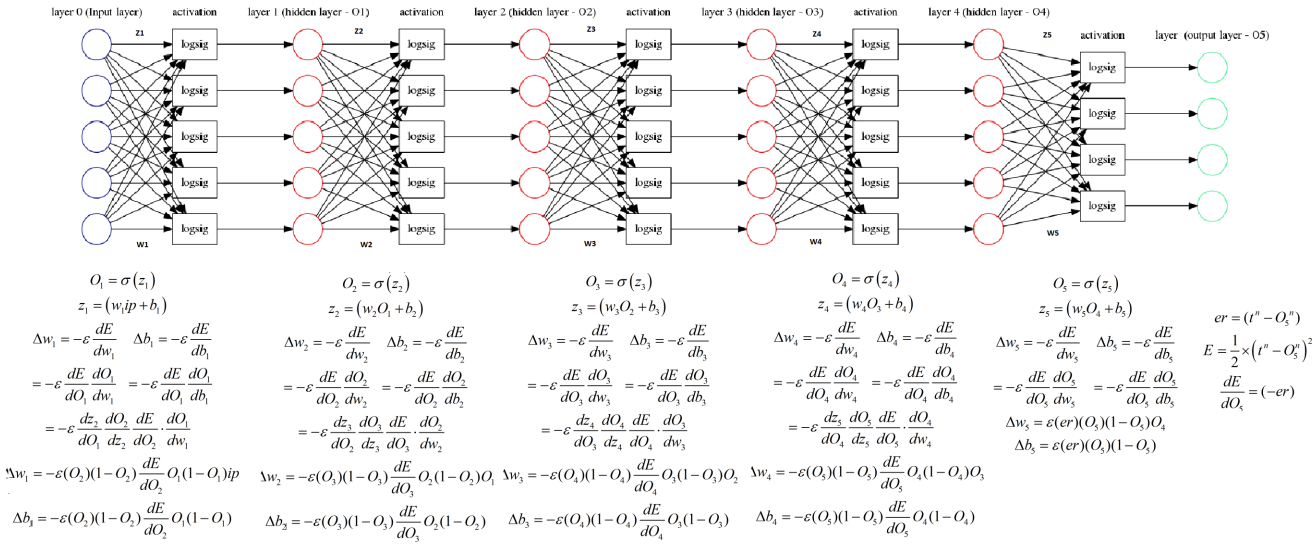


Figure 1: Conventional Feed Forward 5 layered Network

As a comparison to the original 2 layer model. Models with depths of 3, 4 and 5 were trained and explored.

Here similar architecture Networks are compared. All hidden layers have 16 nodes, thus they have weight matrices that have 16x16 weights and 16 biases. All have a sigmoid squasher as the activation function. Hence as a new layer is added, a 16x16 weights matrix is added along with a 16 biases. All models were trained with similar learning rate (0.05) and number of iterations (100k).

Network Depth	Number of Parameters	Training Error	Validation Error
2 Layers	340	7.42	0.31
3 Layers	612	7.62	0.6
4 Layers	884	9.56	0.46
5 Layers	1156	16.01	3.73

Table 1: MLFFs of varying depths for 4-bit A-D mapping

The Validation Error reported here is the deviation from the expected analogue value of the input (analogue value) per samples. So, on an average the 2 Layer Network deviates 0.31 from the expected analogue mapping, while the conventional 5 layer MLFF has a 3.73 average error. This degradation in performance can be attributed to diminishing gradients.

As the residuals or deltas are computed from the outer-most layer to the inner layers, in deep networks the amount of this residuals or deltas that propagate to the initial layers becomes negligent. Effectively rendering those neurons futile in the learning process. Thus, though it might be theoretically better to use deeper networks, the problems associated with vanishing gradients curb the effectiveness of deeper models.

2.2 Residual Networks

ResNets or Residual Networks attempt to solve the vanishing gradients problem by adding jump ahead connections. An example of a Residual block is shown in Figure 2

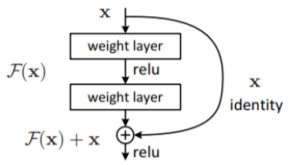
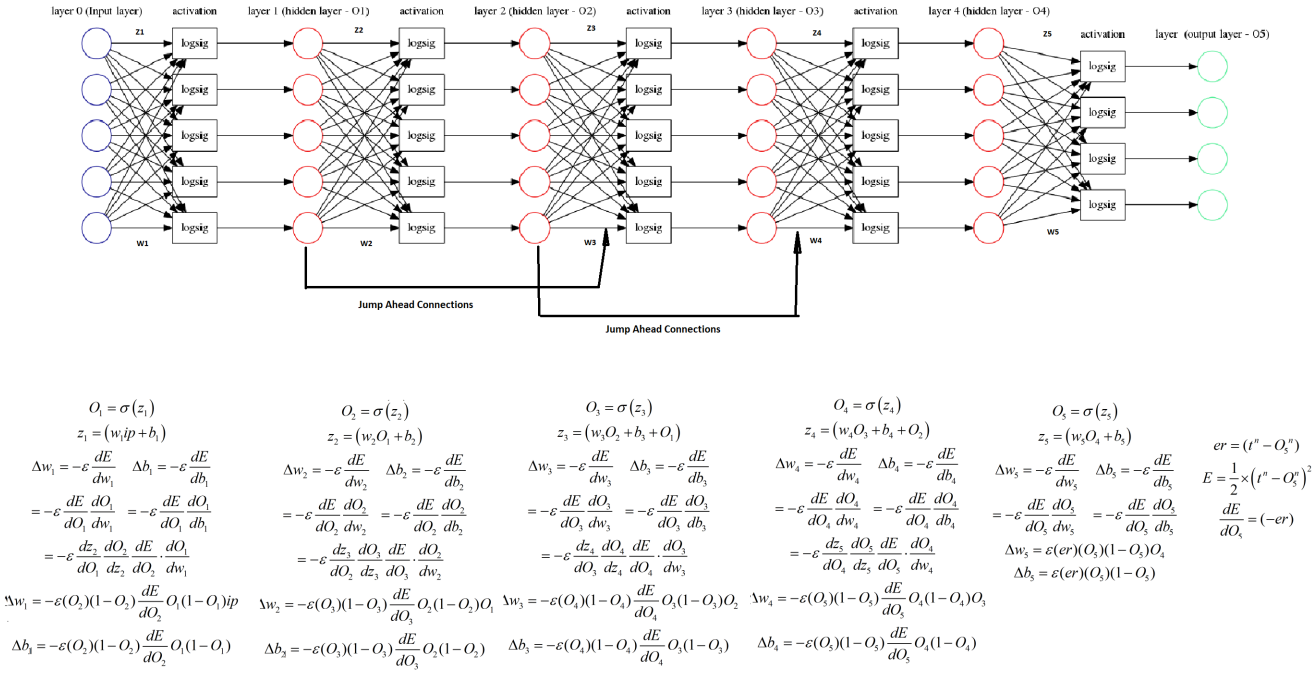


Figure 2: Residual unit block diagram

The jump ahead connections connect the $O(l_i)$ to the $O(l_{i-2})$. As a result of the the neuron activation for the i^{th} row would be

$$O(l_i) = \log sig((W_i O_{i-1} + b_i) + O_{i-2}) \tag{1}$$

As the jump ahead connection adds a identity edge of the $i-2^{th}$ layer, this has no difference from the MLFF for weight update during backprop. The entire structure and the update equations for a 5-Layered ResNet with two jump ahead connections are shown in Figure 3



$$O_1 = \sigma(z_1) \quad z_1 = (w_1 ip + b_1)$$
$$\Delta w_1 = -\varepsilon \frac{dE}{dw_1} \quad \Delta b_1 = -\varepsilon \frac{dE}{db_1}$$
$$= -\varepsilon \frac{dE}{dO_1}\frac{dO_1}{dw_1} \quad = -\varepsilon \frac{dE}{dO_1}\frac{dO_1}{db_1}$$
$$= -\varepsilon \frac{dz_2}{dO_1}\frac{dO_2}{dz_2}\frac{dE}{dO_2}\cdot\frac{dO_1}{dw_1}$$
$$\Delta w_1 = -\varepsilon(O_2)(1-O_2)\frac{dE}{dO_2}O_1(1-O_1)ip$$
$$\Delta b_1 = -\varepsilon(O_2)(1-O_2)\frac{dE}{dO_2}O_1(1-O_1)$$

$$O_2 = \sigma(z_2) \quad z_2 = (w_2 O_1 + b_2)$$
$$\Delta w_2 = -\varepsilon \frac{dE}{dw_2} \quad \Delta b_2 = -\varepsilon \frac{dE}{db_2}$$
$$= -\varepsilon \frac{dE}{dO_2}\frac{dO_2}{dw_2} \quad = -\varepsilon \frac{dE}{dO_2}\frac{dO_2}{db_2}$$
$$= -\varepsilon \frac{dz_3}{dO_2}\frac{dO_3}{dz_3}\frac{dE}{dO_3}\cdot\frac{dO_2}{dw_2}$$
$$\Delta w_2 = -\varepsilon(O_3)(1-O_3)\frac{dE}{dO_3}O_2(1-O_2)O_1$$
$$\Delta b_2 = -\varepsilon(O_3)(1-O_3)\frac{dE}{dO_3}O_2(1-O_2)$$

$$O_3 = \sigma(z_3) \quad z_3 = (w_3 O_2 + b_3 + O_1)$$
$$\Delta w_3 = -\varepsilon \frac{dE}{dw_3} \quad \Delta b_3 = -\varepsilon \frac{dE}{db_3}$$
$$= -\varepsilon \frac{dE}{dO_3}\frac{dO_3}{dw_3} \quad = -\varepsilon \frac{dE}{dO_3}\frac{dO_3}{db_3}$$
$$= -\varepsilon \frac{dz_4}{dO_3}\frac{dO_4}{dz_4}\frac{dE}{dO_4}\cdot\frac{dO_3}{dw_3}$$
$$\Delta w_3 = -\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)O_2$$
$$\Delta b_3 = -\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)$$

$$O_4 = \sigma(z_4) \quad z_4 = (w_4 O_3 + b_4 + O_2)$$
$$\Delta w_4 = -\varepsilon \frac{dE}{dw_4} \quad \Delta b_4 = -\varepsilon \frac{dE}{db_4}$$
$$= -\varepsilon \frac{dE}{dO_4}\frac{dO_4}{dw_4} \quad = -\varepsilon \frac{dE}{dO_4}\frac{dO_4}{db_4}$$
$$= -\varepsilon \frac{dz_5}{dO_4}\frac{dO_5}{dz_5}\frac{dE}{dO_5}\cdot\frac{dO_4}{dw_4}$$
$$\Delta w_4 = -\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)O_3$$
$$\Delta b_4 = -\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)$$

$$O_5 = \sigma(z_5) \quad z_5 = (w_5 O_4 + b_5)$$
$$\Delta w_5 = -\varepsilon \frac{dE}{dw_5} \quad \Delta b_5 = -\varepsilon \frac{dE}{db_5}$$
$$= -\varepsilon \frac{dE}{dO_5}\frac{dO_5}{dw_5} \quad = -\varepsilon \frac{dE}{dO_5}\frac{dO_5}{db_5}$$
$$\Delta w_5 = \varepsilon(er)(O_5)(1-O_5)O_4$$
$$\Delta b_5 = \varepsilon(er)(O_5)(1-O_5)$$

$$er = (t^n - O_5^n) \quad E = \frac{1}{2}\times(t^n - O_5^n)^2 \quad \frac{dE}{dO_5} = (-er)$$

Figure 3: Residual 5 layered Network ResNet_5

Some benefits of Resnets would be

- They provide alternate paths for derivatives to flow

- As the jump ahead connections provide a identity mapping, the same Backpropogation framework used for the MLFFs can be employed
- There are no increase in number of parameters so the time and space complexity is similar to Feed Forward Networks

Figure 4 shows the training errors over iterations. We see that the conventional MLFF doesn't find a usable gradient initially, thus has an unchanging training error, while the ResNet finds a gradient and starts to decrease its training error.

The accuracy for the two Networks are shown in Table 2

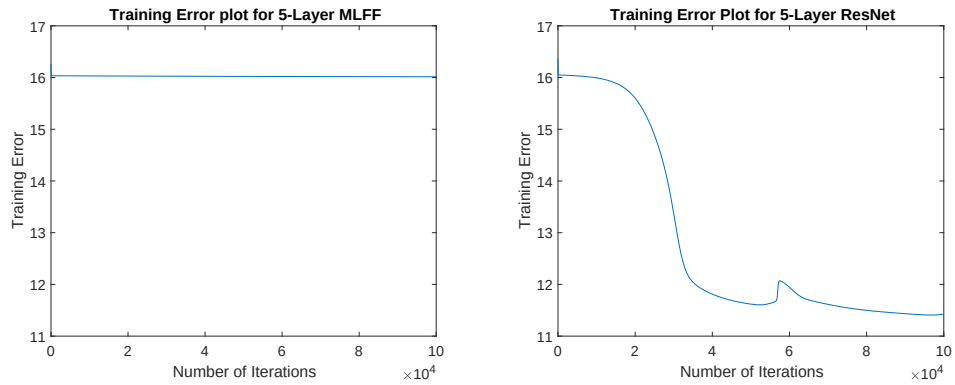


Figure 4: Training and Validation comparision for 5-Layered MLFF and ResNet

Model	5-Layer MLFF	5-Layer Resnet
Training Error	16.01	11.42
Validation Error	3.73	0.97

Table 2: Accuracy: MLFF and ResNet

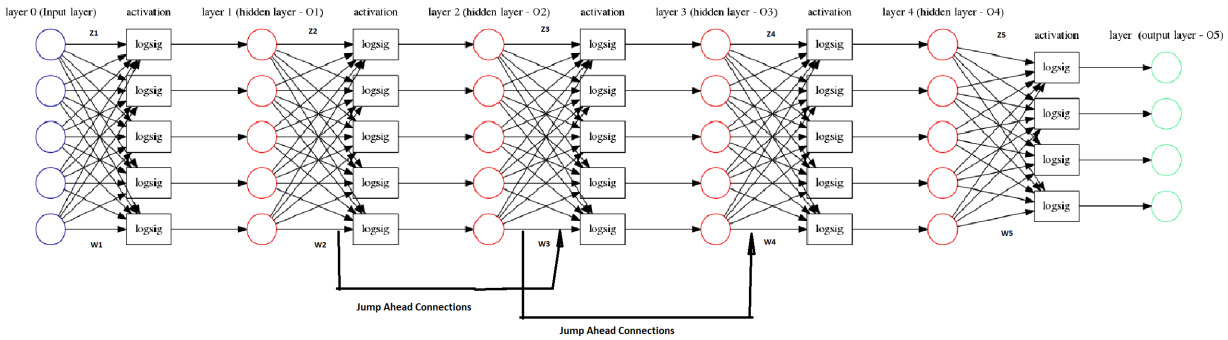
2.3 Modified Resnet

From the above definition of ResNets Eq (1) the output of the $i-2^{th}$ activation layer is fed to the i^{th} layer. It can be noticed here that the quantity O_{i-2} undergoes one 'thresholding' by the $i-2^{th}$ activation layer and another 'thresholding' by the i^{th} activation layer. This double-thresholding can be avoided by connecting z_{i-2}^{th} to O_i.

This modification is depicted by the below equation

$$O\left(l_i\right) = \log sig\left(\left(W_i O_{i-1} + b_i\right) + Z_{i-2}\right) \Rightarrow \log sig\left(\left(W_i O_{i-1} + b_i\right) + \left(W_{i-2} O_{i-3} + b_{i-2}\right)\right) \quad (2)$$

The accuracy for the two Networks are shown in Table 3



$$O_1=\sigma(z_1)$$
$$z_1=(w_1 ip+b_1)$$
$$\Delta w_1=-\varepsilon\frac{dE}{dw_1}\quad \Delta b_1=-\varepsilon\frac{dE}{db_1}$$
$$=-\varepsilon\frac{dE}{dO_1}\frac{dO_1}{dw_1}\quad =-\varepsilon\frac{dE}{dO_1}\frac{dO_1}{db_1}$$
$$=-\varepsilon\frac{dz_2}{dO_1}\frac{dO_2}{dz_2}\frac{dE}{dO_2}\cdot\frac{dO_1}{dw_1}$$
$$\Delta w_1=-\varepsilon(O_2)(1-O_2)\frac{dE}{dO_2}O_1(1-O_1)ip$$
$$\Delta b_1=-\varepsilon(O_2)(1-O_2)\frac{dE}{dO_2}O_1(1-O_1)$$

$$O_2=\sigma(z_2)$$
$$z_2=(w_2O_1+b_2)$$
$$\Delta w_2=-\varepsilon\frac{dE}{dw_2}\quad \Delta b_2=-\varepsilon\frac{dE}{db_2}$$
$$=-\varepsilon\frac{dE}{dO_2}\frac{dO_2}{dw_2}\quad =-\varepsilon\frac{dE}{dO_2}\frac{dO_2}{db_2}$$
$$=-\varepsilon\frac{dz_3}{dO_2}\frac{dO_3}{dz_3}\frac{dE}{dO_3}\cdot\frac{dO_2}{dw_2}$$
$$\Delta w_2=-\varepsilon(O_3)(1-O_3)\frac{dE}{dO_3}O_2(1-O_2)O_1$$
$$\Delta b_2=-\varepsilon(O_3)(1-O_3)\frac{dE}{dO_3}O_2(1-O_2)$$

$$O_3=\sigma(z_3)$$
$$z_3=(w_3O_2+b_3+z_1)$$
$$\Delta w_3=-\varepsilon\frac{dE}{dw_3}\quad \Delta b_3=-\varepsilon\frac{dE}{db_3}$$
$$=-\varepsilon\frac{dE}{dO_3}\frac{dO_3}{dw_3}\quad =-\varepsilon\frac{dE}{dO_3}\frac{dO_3}{db_3}$$
$$=-\varepsilon\frac{dz_4}{dO_3}\frac{dO_4}{dz_4}\frac{dE}{dO_4}\cdot\frac{dO_3}{dw_3}$$
$$\Delta w_3=-\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)O_2$$
$$\Delta b_3=-\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)$$

$$O_4=\sigma(z_4)$$
$$z_4=(w_4O_3+b_4+z_2)$$
$$\Delta w_4=-\varepsilon\frac{dE}{dw_4}\quad \Delta b_4=-\varepsilon\frac{dE}{db_4}$$
$$=-\varepsilon\frac{dE}{dO_4}\frac{dO_4}{dw_4}\quad =-\varepsilon\frac{dE}{dO_4}\frac{dO_4}{db_4}$$
$$=-\varepsilon\frac{dz_5}{dO_4}\frac{dO_5}{dz_5}\frac{dE}{dO_5}\cdot\frac{dO_4}{dw_4}$$
$$\Delta w_4=-\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)O_3$$
$$\Delta b_4=-\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)$$

$$O_5=\sigma(z_5)$$
$$z_5=(w_5O_4+b_5)$$
$$\Delta w_5=-\varepsilon\frac{dE}{dw_5}\quad \Delta b_5=-\varepsilon\frac{dE}{db_5}$$
$$=-\varepsilon\frac{dE}{dO_5}\frac{dO_5}{dw_5}\quad =-\varepsilon\frac{dE}{dO_5}\frac{dO_5}{db_5}$$
$$\Delta w_5=\varepsilon(er)(O_5)(1-O_5)O_4$$
$$\Delta b_5=\varepsilon(er)(O_5)(1-O_5)$$

$$er=(t^n-O_5^n)$$
$$E=\frac{1}{2}\times(t^n-O_5^n)^2$$
$$\frac{dE}{dO_5}=(-er)$$

Figure 5: Modified ResNet_5 Network

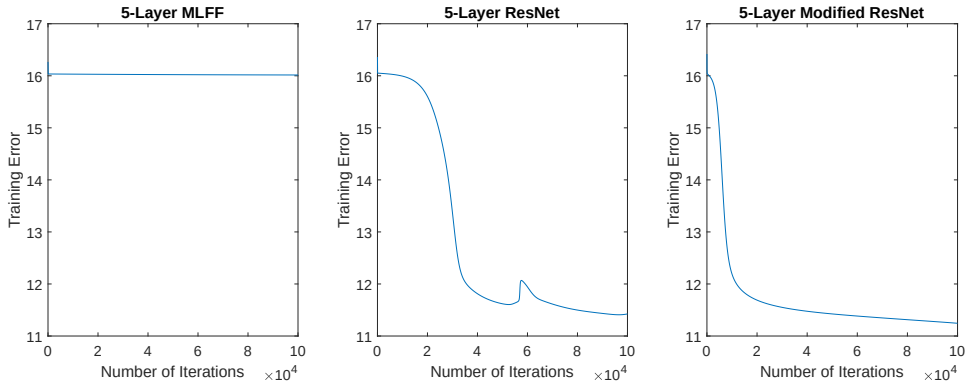


Figure 6: Training and Validation comparision for 5-Layered MLFF, ResNet and Modified ResNet

Model	5-Layer MLFF	5-Layer Resnet	5-Layer Resnet Mod
Training Error	16.01	11.42	11.24
Validation Error	3.73	0.97	1.02

Table 3: Accuracy: MLFF ResNet and Modified ResNet

2.4 Paramterized Resnets

The next modification would intuitively be to parameterize this Jump ahead connection. Parameterizing this gives some control to the amount of gradient/delta/residual that is shared between these layers.

This also adds a some more 'degrees of freedom' to achieve a better mapping.

The neuron activation for this implementation becomes

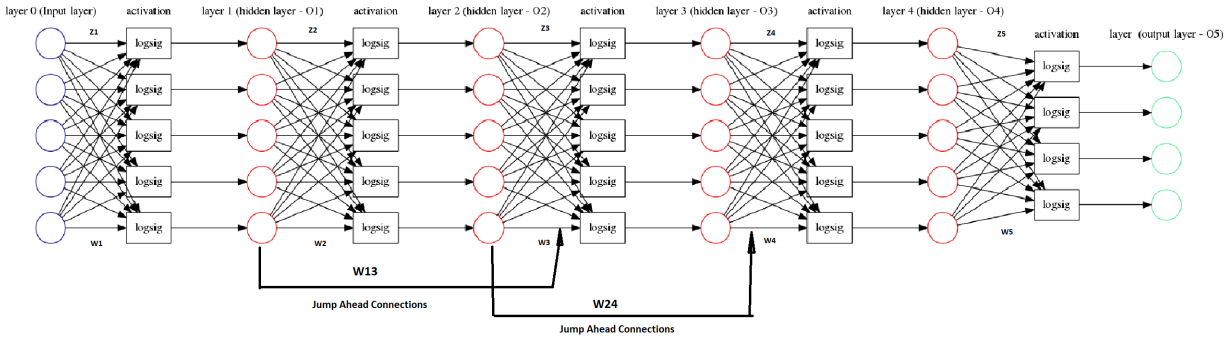
$$O\left(l_i\right) = \log sig\left(\left(W_i O_{i-1} + b_i\right) + W_{i,i-2} O_{i-2}\right) \tag{3}$$

Where $W_{i,i-2}$ is the newly introduced parameter.

So now the weight update equations for this new parameter can be written as:

$$\Delta w_{13} = -\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)O_1$$

$$\Delta w_{24} = -\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)O_1$$



$$O_1 = \sigma(z_1)$$
$$z_1 = (w_1 ip + b_1)$$
$$\Delta w_1 = -\varepsilon\frac{dE}{dw_1} \quad \Delta b_1 = -\varepsilon\frac{dE}{db_1}$$
$$= -\varepsilon\frac{dE}{dO_1}\frac{dO_1}{dw_1} \quad = -\varepsilon\frac{dE}{dO_1}\frac{dO_1}{db_1}$$
$$= -\varepsilon\frac{dz_2}{dO_1}\frac{dO_2}{dz_2}\frac{dE}{dO_2}\cdot\frac{dO_1}{dw_1}$$
$$\Delta w_1 = -\varepsilon(O_2)(1-O_2)\frac{dE}{dO_2}O_1(1-O_1)ip$$
$$\Delta b_1 = -\varepsilon(O_2)(1-O_2)\frac{dE}{dO_2}O_1(1-O_1)$$

$$O_2 = \sigma(z_2)$$
$$z_2 = (w_2 O_1 + b_2)$$
$$\Delta w_2 = -\varepsilon\frac{dE}{dw_2} \quad \Delta b_2 = -\varepsilon\frac{dE}{db_2}$$
$$= -\varepsilon\frac{dE}{dO_2}\frac{dO_2}{dw_2} \quad = -\varepsilon\frac{dE}{dO_2}\frac{dO_2}{db_2}$$
$$= -\varepsilon\frac{dz_3}{dO_2}\frac{dO_3}{dz_3}\frac{dE}{dO_3}\cdot\frac{dO_2}{dw_2}$$
$$\Delta w_2 = -\varepsilon(O_3)(1-O_3)\frac{dE}{dO_3}O_2(1-O_2)O_1$$
$$\Delta b_2 = -\varepsilon(O_3)(1-O_3)\frac{dE}{dO_3}O_2(1-O_2)$$

$$O_3 = \sigma(z_3)$$
$$z_3 = (w_3 O_2 + b_3 + (w_{13} O_1))$$
$$\Delta w_3 = -\varepsilon\frac{dE}{dw_3} \quad \Delta w_{13} = -\varepsilon\frac{dE}{dw_{13}}$$
$$= -\varepsilon\frac{dE}{dO_3}\frac{dO_3}{dw_3} \quad = -\varepsilon\frac{dE}{dO_3}\frac{dO_3}{dw_{13}}$$
$$\Delta w_3 = -\varepsilon\frac{dz_4}{dO_3}\frac{dO_4}{dz_4}\frac{dE}{dO_4}\cdot\frac{dO_3}{dw_3}$$
$$\Delta w_{13} = -\varepsilon\frac{dz_4}{dO_3}\frac{dO_4}{dz_4}\frac{dE}{dO_4}\cdot\frac{dO_3}{dw_{13}}$$
$$\Delta w_3 = -\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)O_2$$
$$\Delta b_3 = -\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)$$
$$\Delta w_{13} = -\varepsilon(O_4)(1-O_4)\frac{dE}{dO_4}O_3(1-O_3)O_1$$

$$O_4 = \sigma(z_4)$$
$$z_4 = (w_4 O_3 + b_4 + (w_{24} O_2))$$
$$\Delta w_4 = -\varepsilon\frac{dE}{dw_4} \quad \Delta O_{24} = -\varepsilon\frac{dE}{dO_{24}}$$
$$= -\varepsilon\frac{dE}{dO_4}\frac{dO_4}{dw_4} \quad = -\varepsilon\frac{dE}{dO_4}\frac{dO_4}{dO_{24}}$$
$$\Delta w_4 = -\varepsilon\frac{dz_5}{dO_4}\frac{dO_5}{dz_5}\frac{dE}{dO_5}\cdot\frac{dO_4}{dw_4}$$
$$\Delta w_{24} = -\varepsilon\frac{dz_5}{dO_4}\frac{dO_5}{dz_5}\frac{dE}{dO_5}\cdot\frac{dO_4}{dw_{24}}$$
$$\Delta w_4 = -\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)O_3$$
$$\Delta b_4 = -\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)$$
$$\Delta w_{24} = -\varepsilon(O_5)(1-O_5)\frac{dE}{dO_5}O_4(1-O_4)O_1$$

$$O_5 = \sigma(z_5)$$
$$z_5 = (w_5 O_4 + b_5)$$
$$\Delta w_5 = -\varepsilon\frac{dE}{dw_5} \quad \Delta b_5 = -\varepsilon\frac{dE}{db_5}$$
$$= -\varepsilon\frac{dE}{dO_5}\frac{dO_5}{dw_5} \quad = -\varepsilon\frac{dE}{dO_5}\frac{dO_5}{db_5}$$
$$\Delta w_5 = \varepsilon(er)(O_5)(1-O_5)O_4$$
$$\Delta b_5 = \varepsilon(er)(O_5)(1-O_5)$$

$$er = (t^n - O_5^n)$$
$$E = \frac{1}{2}\times\left(t^n - O_5^n\right)^2$$
$$\frac{dE}{dO_5} = (-er)$$

Figure 7: Parameterized ResNet_5 Network

The accuracy for the two Networks are shown in Table 4

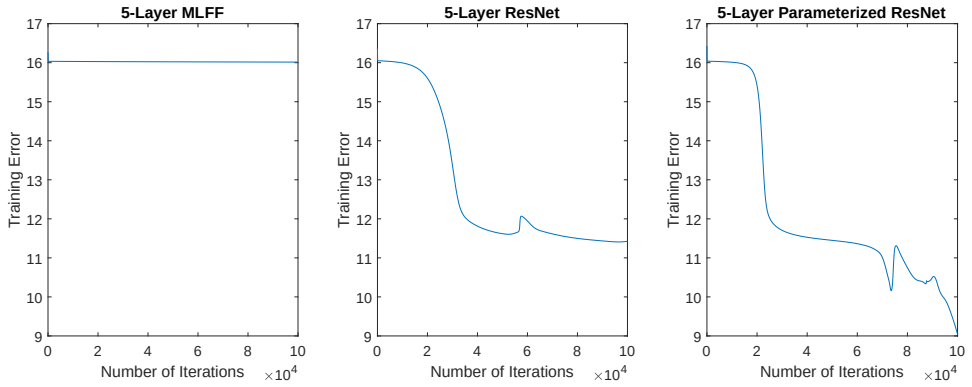


Figure 8: Training and Validation comparision for 5-Layered MLFF, ResNet and Parameterized ResNet

Model	5-Layer MLFF	5-Layer Resnet	5-Layer Parameterized ResNet
Training Error	16.01	11.42	9.04
Validation Error	3.73	0.97	0.64

Table 4: Accuracy: MLFF ResNet and Parameterized ResNet

2.5 Ensemble Learning

Ensemble Training involves training multiple models with different parameters. The parameters might vary in different starting points in the weight space (i.e., different random initialization) or with different hyper-parameters, and these models are intelligently combined. The intelligent combination here is letting the models with a better performance, lower training or validation error, to breed, i.e., copy their weights to other models and continue training these models. At the next checkpoint, again an evaluation is made, and the weights from the best performing model are copied to other models.

Here an ensemble of 5 models. They start at different random initial points and have different learning rates.

And as designed the Ensembeled model would have the weights that produce the minimum error amongst the 5 models it was trained with.

3 Implementations and Results

As there are many hyper-parameters that can be tuned and many models implemented, the Results for this assignment were generated by maintaining similar conditions amongst models to ensure they can be compared. This being said, individual models can be fine tuned to outperform the result published here.

As larger models are trained for large iterations, maintaining a static learning isn't practical. If the learning rate is too large, then the system starts to destabilize after some iterations, and conversely if a very small learning rate is chosen, the system is stable but the solution reached in the specific number of iterations is still not optimal. Thus for the 4 and

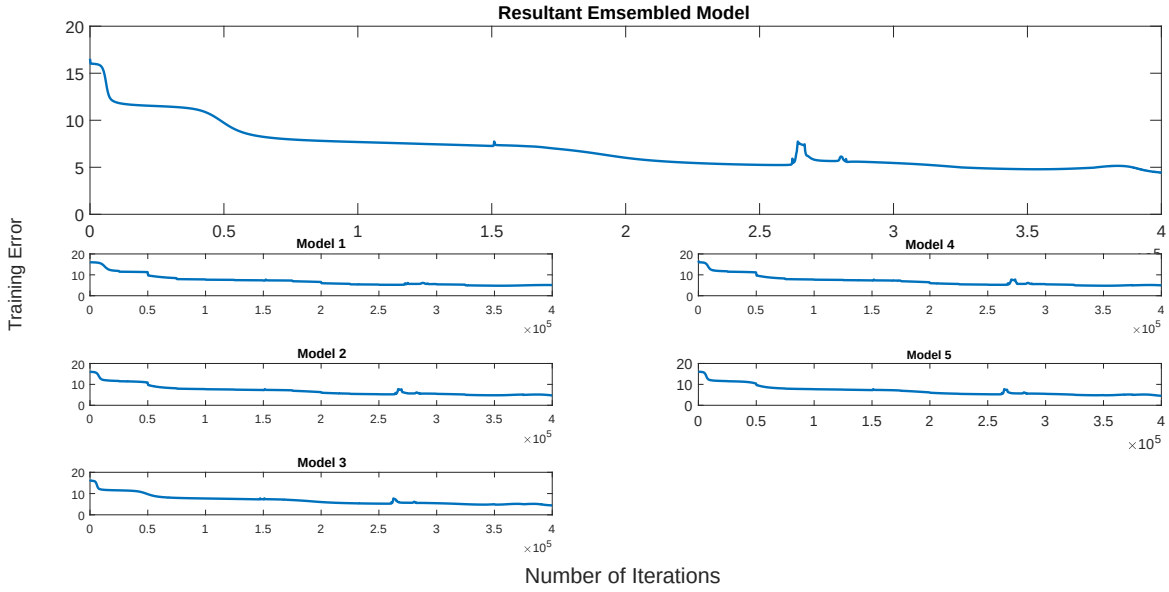


Figure 9: Emsembled Model

Model	Training Error	Validation Error
2-Layer Starter Model	5.02e-4	0.19
4-Layer MLFF Net	13.73	2.06
4-Layer ResNet	10.91	1.70
4-Layer ResNet with LR decay	4.90	**0.18**
4-Layer Parameterized ResNet with LR Decay	7.42	0.44
5-Layer MLFF Net	9.29	1.04
5-Layer ResNet	11.11	1.19
5-Layer ResNet with LR decay	5.13	0.49
5-Layer Parameterized ResNet with LR Decay	10.02	0.62

5 layer networks a learning rate decay of 10% every 50,000 iterations was also applied.

The entire Implementation aims to outperform the standard 2-Layer starter A-D converter provided, but for reasons unknown only one out of the 8 models trained outperformed the starter model.

It can be noted that though the 2-Layer model performs well on the validation set, its corresponding training error is 5e-2. Thus, very small error or very small deltas are being back propagated, leaving very less scope for improvement. While the 4-Layer ResNet still has a substantially higher error, leaving scope for improvement and showing that the neurons aren't saturated yet. This might lead us to believe that with more iterations the model might find a substantially better solution.

- Though most models failed to outperform the 2-Layer Starter model, it can be concluded that 'Deeper Models' may not perform better.
- Deeper models tend to have lower validation errors

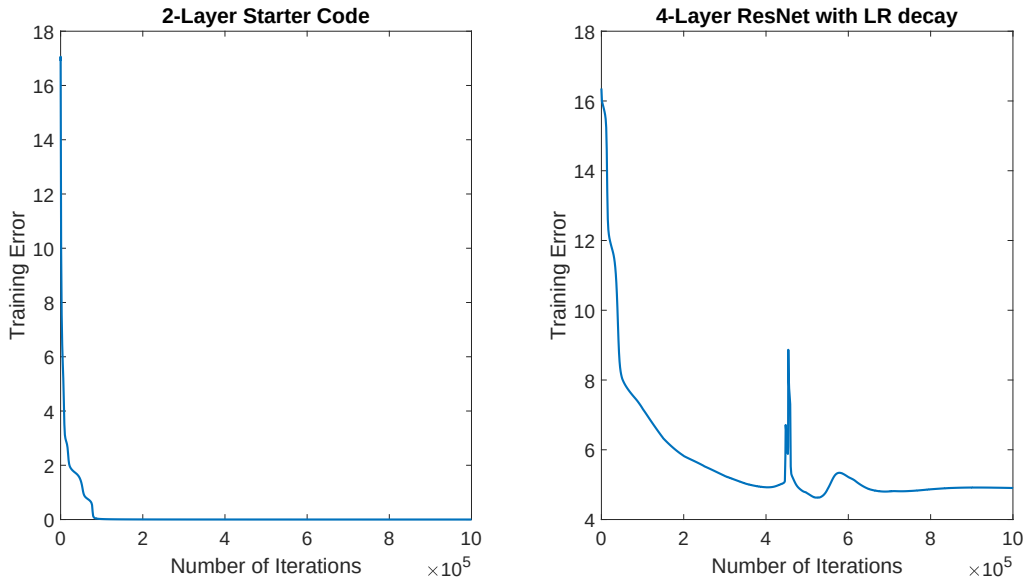


Figure 10: Training of the Starter Model and Best Validated Model

- The Residual Networks, almost consistently outperform the conventional MLFF Networks

An ensembled model of this 4-Layer ResNet with LR Decay produced a validation error of

4 Conclusions

Residual Networks create alternate paths for the gradients/residuals to propagate through the network. Modifications to these ResNets seem to yield, similar (if not better) results. The parameterized ResNet can find applications in determining the efficiency of these residual connections. For instance, if on multiple instances it is found that these Jump-ahead connections take zero (or very small) weights, then it can be argued that jump-ahead connections are not preferred over Feed Forward Networks. In any case, these jump-ahead/residual connections seem to yield better mappings for the A-D mapping than their MLFF counterparts.

This concept can find best applications in very deep Convolutional Neural Nets, where there are between 50 and 150 layers. So as a proof of working of this concept the ResNet implementation on Keras can be easily modified to incorporate these changes. Then the modified ResNet model can be trained to classify a standard dataset, say MNIST, and compare how close (or better) the modified versions fare. As it is a classification problem with a deeper CNN, the parameterized ResNet would have more parameters to establish a better mapping.