AN EFFICIENT ALTERNATIVE OF THE IDENTITY MAPPING OF THE ResNet ARCHITECTURE

A Thesis submitted by

ARPAN KUMAR BAG

In the partial fulfillment of the requirements for the degree of

MASTER OF TECHNOLOGY

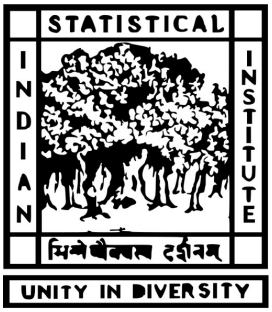
In

COMPUTER SCIENCE

Under the guidance of

Dr. UJJWAL BHATTACHARYA
Associate Professor (Equiv.)
Computer Vision and Pattern Recognition Unit

Indian Statistical Institute, Kolkata



August 31, 2020

Acknowledgement

I sincerely express my gratitude towards my supervisor, Dr.Ujjwal Bhattacharya for providing me with the necessary materials and for all his advice and support. I would also like to sincerely thank all the researchers working in the Handwriting Recognition Lab of the Computer Vision and Pattern Recognition Unit of my institute, Indian Statistical Unit, Kolkata.

Abstract

In the past few years, deep models have made a huge impact in the field of computer vision. Among these deep models, Residual Networks or ResNets have become particularly popular for their simple architecture and efficient performance. Despite the achievement, the skip connection which made the training of a very deep model possible was also considered as a drawback of this model. Some studies have been done on the comparative performance of various types of skip connections. Inspired by the recent work on skip connection which proposed use of ReLU with group normalization as an alternative to the identity skip connection resulting in better performance than traditional ResNet, we have explored use of various activation functions. In this thesis, we propose a different transformation to be used together with the ReLU Group Normalization (RG) connection to improve the performance of Residual networks. We simulated our results on CIFAR-10 and CIFAR-100 datasets. The code developed as a part of this study is available at https://github.com/Arpan142/Arpan_dissertation.

Contents

Chapter 1

Introduction

1.1 Preliminaries

In the recent years it has been seen that increasing the number of layers in the convolution neural network models, produces a much superior performance. Models like Alexnet, VGG, Inception, ResNet, etc. Among all these models ResNet has been experimented with a large number of layers and achieved state of the art benchmark in several tasks including object classification on ImageNet [1] and CIFAR [2], object detection and segmentation on MSCOCO [3] and PASCAL VOC [4]. ResNet quickly caught everyone's attention because of its simple architecture and outstanding performance.

1.1.1 Why deep networks

From circuit complexity theory it can be seen that shallow network may require exponentially more components than deep networks [5]. The ResNet model is made thin and deep which leads to a model with less parameter [5]. A simple example is that a 7×7 convolution has the same receptive field as three 3×3 convolutions.

1.1.2 Residual Networks

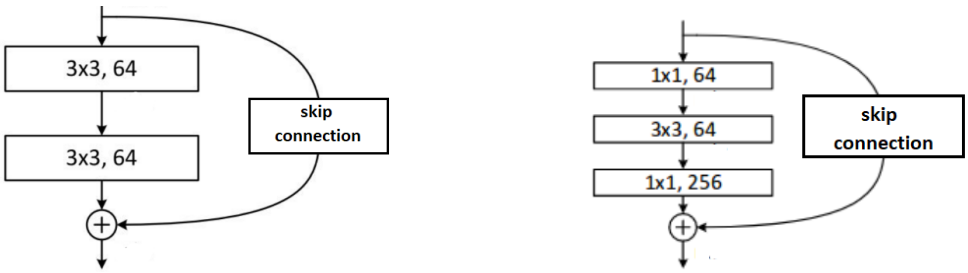
Deep residual networks consists of many stacked "residual units", each unit can be expressed as:

$$y_l = h(x_l) + F(x_l, w_l), \tag{1.1}$$

$$x_{l+1} = f(y_l), \tag{1.2}$$

[6] where x_l and x_{l+1} is the input and output of the lth layer. The h function represents skip connection, F represents the function of a block,w_l represents the weights of the lth block and f represents the final transformation which produces the output of lth block. It has been shown that $h(x_l) = x_l$ performs best among many choices of h [6]. The identity connection provides gradient stability which helps the network to converge in spite of using many layers.

In ResNet, two types of blocks are used. By block, we mean a sequence of operations using some convolution layers and some functions(ReLU, batch normalization, etc.). The depth of a ResNet model is measured by calculating the number of learnable layers, e.g., ResNet-26 means the model has 26 learnable layers. The basic block consists of two 3×3 convolution and the bottleneck block consists of two 1×1 convolution and a 3×3 convolution.



(a) Basic Block (b) Bottleneck Block

Figure 1.1: The basic block consists of two 3×3 convolution. Here the channel dimension stays the same throughout the block. In bottleneck block, the channel dimension of the output of the last 1×1 convolution is 4 times the channel dimension of the output of the first 1×1 convolution.

In ResNet, four convolution groups of blocks are used normally. The authors of [7] called these groups as $conv2, conv3, conv4, conv5$. $conv1$ being the convolution layers at the beginning of the model which do not belong to any building block (basic block or bottleneck block). Each convolution group contains a stack of basic blocks or bottleneck blocks. After each convolution group the channel dimension of the input increases and the height and the width decreases. This is known as downsampling. Throughout this thesis we will be using bottleneck block. So in each bottleneck block there is three learnable layers. At the beginning of ResNet, a 7×7 or a 3×3 convolution is used. Sometimes a pooling layer is used after the first convolution layer. At the end a fully connected layer is used for classification purpose. The performance of ResNet

for classification tasks is measured using top-1 error. The top-1 error means for how many inputs the class predicted by the network, i.e., the highest class probability of the softmax output matches the actual class of the input.

The architecture of ResNet-34 is shown in figure 1.2 . This design is for ImageNet [1] classification where the number of classes are 1000. The input image is passed through a 7×7 convolution with stride 2 which reduces the image height and width by half and the output channel dimension is 64. After each convolution group, downsampling takes place and at the end of the architecture there is an average pooling layer and after that a fully connected layer for classification.

1.1.3 Problems with ResNet

As mentioned in [5] the identity map which helps with gradient stability is also a drawback for ResNet. The identity map does not learn any useful features. Since the gradient flows through the residual connection there is no guarantee that the gradient will go through the residual block weights which means that only the lower level blocks may learn some useful information. This is known as diminishing feature reuse [8]. DenseNet [9] achieves better performance than ResNet with the same number of parameters. A dense-block has direct connection between a layer and its subsequent layers. But DenseNet requires large GPU memory and large computation time for training [10].

1.1.4 Motivation

Skip connections have made a huge impact on deep learning. Skip connections have been used in many applications of NLP [11], computer vision [12], [9] etc. The skip connections help with the gradient flow which deals with the vanishing gradient problem. We have the relation $x_{l+1} = x_l + F(x_l, w_l)$. We are using the notations mentioned in the previous section and ϵ is the error produced by the loss function. Computation of error gradients in backpropagation is shown below.

$$\frac{\partial \epsilon}{\partial x_l} = \frac{\partial \epsilon}{\partial x_{l+1}} \frac{\partial x_{l+1}}{\partial x_l} = \frac{\partial \epsilon}{\partial x_{l+1}} (1 + \frac{\partial F(x_l, w_l)}{\partial x_l}) \tag{1.3}$$

$$\frac{\partial \epsilon}{\partial x_{l+1}} (1 + \frac{\partial F(x_l, w_l)}{\partial x_l}) = \frac{\partial \epsilon}{\partial x_{l+2}} \frac{\partial x_{l+2}}{\partial x_{l+1}} (1 + \frac{\partial F(x_l, w_l)}{\partial x_l}) \tag{1.4}$$

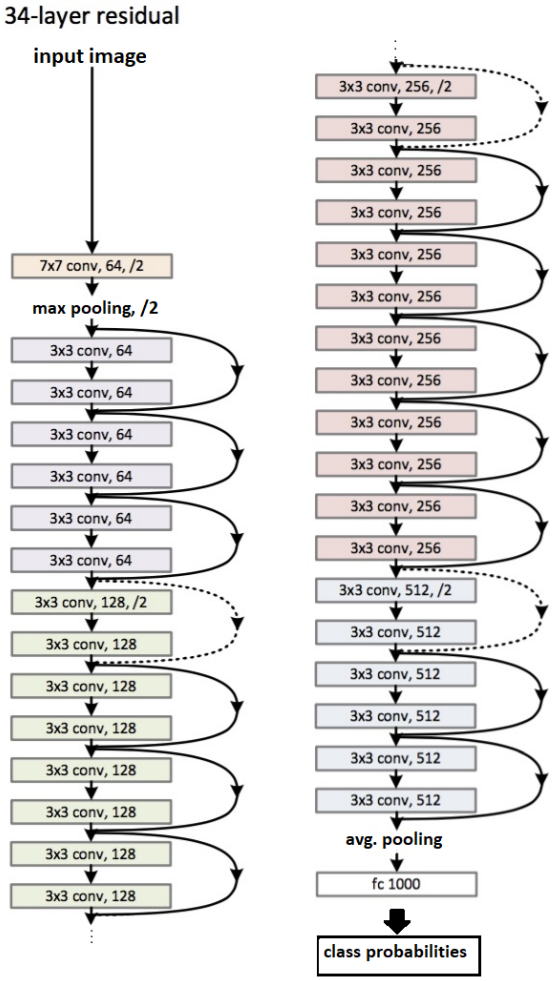


Figure 1.2: ResNet-34 with basic block designed for ImageNet [1] classification. The dotted skip connection means downsampling. For ResNet-34 the convolution group lengths are 3, 4, 6, 3. So that after the first three basic blocks first downsampling takes place. Then again after three blocks and again after five blocks. In the figure /2 means the dimension of the input height and width is getting halved. The numbers after the convolution dimension represent the channel dimension of the output.

$$\frac{\partial \epsilon}{\partial x_{l+2}} \frac{\partial x_{l+2}}{\partial x_{l+1}} (1 + \frac{\partial F(x_l, w_l)}{\partial x_l}) = \frac{\partial \epsilon}{\partial x_{l+2}} (1 + \frac{\partial F(x_{l+1}, w_{l+1})}{\partial x_{l+1}})(1 + \frac{\partial F(x_l, w_l)}{\partial x_l}) \tag{1.5}$$

So if we multiply all these terms we can see that gradient is flowing directly from the last layer to any other layer. The skip connection does not learn any useful feature [5]. Recent work [10] has proposed a simple nonlinear transformation as skip connection which achieves better performance than traditional ResNet. This proves that there is

room for improvement in this area which makes this area a good field for research.

1.2 Activation functions

Activation functions are used at a node of a neural network. That is after multiplying the input with the weight, the result is passed through the activation function which ultimately determines the output. Sometimes only the input can be passed through the activation function without performing any weight multiplication [6], [10]. The activation functions are mainly used to introduce nonlinearities in the network so that the network can capture complex patterns.

1.2.1 Various types of activation function

- **ReLU:** ReLU is defined as

$$ReLU(x_i) = \begin{cases} x_i, & \text{if } x_i > 0 \\ 0, & \text{otherwise} \end{cases}$$

- **GELU:** GELU[13] is defined as

$$GELU(x_i) = x_i\Phi(x_i) = 0.5x_i(1 + erf(\frac{x_i}{\sqrt{2}}))$$

  Here Φ is the cumulative distribution of $\mathcal{N}(0, 1)$ and $erf(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2} dt$

- **Logistic/Sigmoid:** Logistic or Sigmoid can be defined as

$$Sigmoid(x_i) = \frac{1}{1 + e^{-x_i}}$$

- **Softmax:** Softmax function is defined as

$$Softmax(\vec{x})_i = \frac{e^{x_i}}{\sum_i e^{x_i}}$$

- **Tanh:** Tanh function is defined as

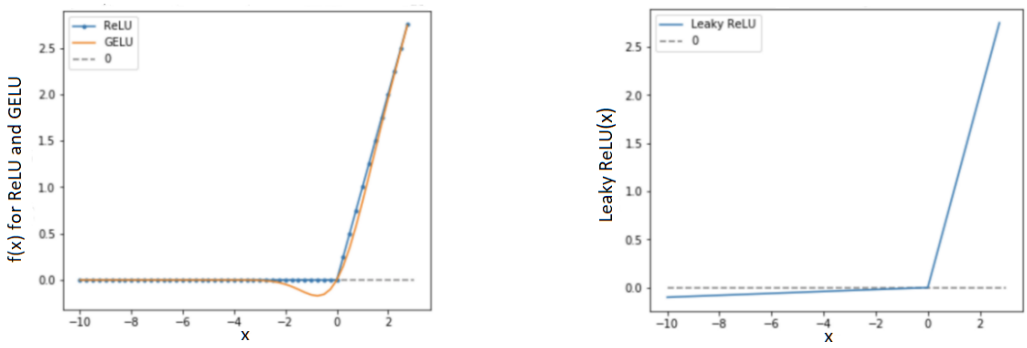
$$Tanh(x_i) = \frac{e^{x_i} - e^{-x_i}}{e^{x_i} + e^{-x_i}} \cdots$$

- **Leaky ReLU:** The Leaky ReLU is defined as

$$Leaky\ ReLU(x_i) = \begin{cases} x_i, & \text{if } x_i > 0 \\ \alpha x_i, & \text{otherwise} \end{cases}$$

  Here α is a hyperparameter.

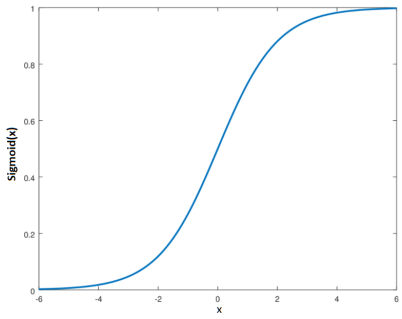
All the activation functions mentioned above except Softmax, take a scalar as input. If these functions are applied on a vector then they are applied on each coordinate of the vector. The input of these functions is denoted by x_i, the ith coordinate of the vector x. For Softmax the input is a vector and the output is also a vector. Among the activation functions ReLU, GELU, Leaky ReLU are unbounded. The range of Sigmoid and Tanh are $(0, 1)$ and $(-1, 1)$ respectively. The sup norm of a softmax output $\in (0, 1]$. Softmax is mainly used at the last layer of a network for classification problem to generate class probabilities.



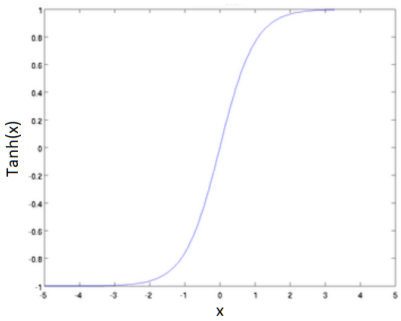
(a) ReLU and GELU

(b) Leaky ReLU

Figure 1.3: The left side represents ReLU and GELU function and the right side represents Leaky ReLU function where $\alpha = 0.01$.



(a) Sigmoid



(b) Tanh

Figure 1.4: The left side represents the Sigmoid function whose range is $(0, 1)$ and the right figure represents the Tanh function whose range is $(-1, 1)$.

1.3 Loss function

For the loss function, we use cross-entropy loss which is the negative log-likelihood of the softmax output. For our context at the end of the ResNet model we have a fully connected layer where the output dimension is the number of classes. The fully connected layer performs a linear transformation to the input. Then applies softmax on the output (which has the dimension same as the number of classes). After performing the softmax we need to pass the class of the input as a target. Then we take the negative log-likelihood of the target component of the softmax vector. The total loss is the average loss across all input in a batch.

1.4 Our Contribution

Our work explores different activation functions and is mainly based on the work of [10]. The previous works show [6] that the use of scaling, gating, 1×1 convolution, Dropout as a shortcut connection impedes the signal propagation leading to higher test error. The latest work [10] uses a simple yet effective shortcut. The authors of [10] used ReLU and group normalization (RG) as a shortcut connection and some variations of it which are discussed in the previous work section. Our work uses this model with an extra transformation on each channel of the input in each layer which results in a better performance. We first tested the transformation on smaller architecture then moved to larger ones.

Chapter 2

Previous Work

As mentioned earlier a lot of work has already been done on deep models, and many methods have arisen for better performance, like auxiliary classifiers [14], weight initialization [12], normalization [15], [16], [17], [18], [19] etc. Among these, we will focus on normalization techniques. Normalization became crucial for better performance of deep networks. There are four types of normalization techniques. Batch, layer, instance and Group. Among these we will focus on batch normalization [15] and group normalization [18]. In any normalization technique we normalize the data at that epoch along a particular axis. In the case of batch normalization [15] the normalization takes place along the batch axis and in case of group normalization [18] we split the channel axis into some number of groups then we normalize these groups along the height and width axis.

(a) Batch Normalization (b) Group Normalization

Figure 2.1: The batch normalization takes place along the batch axis (N axis) and the group normalization takes place along the channel axis (C axis). The H, W axis represents to height and width.

Group normalization is an efficient method when it comes to memory constraint. As the batch size decreases the performance of batch normalization deteriorates [18]. It has

been found that group normalization works better when used in shortcut connection [10]. In [10] the authors proposed three different models. They used bottleneck block in all their experiments and in place of skip connection they used **ReLU** with **group normalization**. The ReLU was for nonlinearity and the group normalization was for extra stability [10]. The 3 different models proposed in [10].

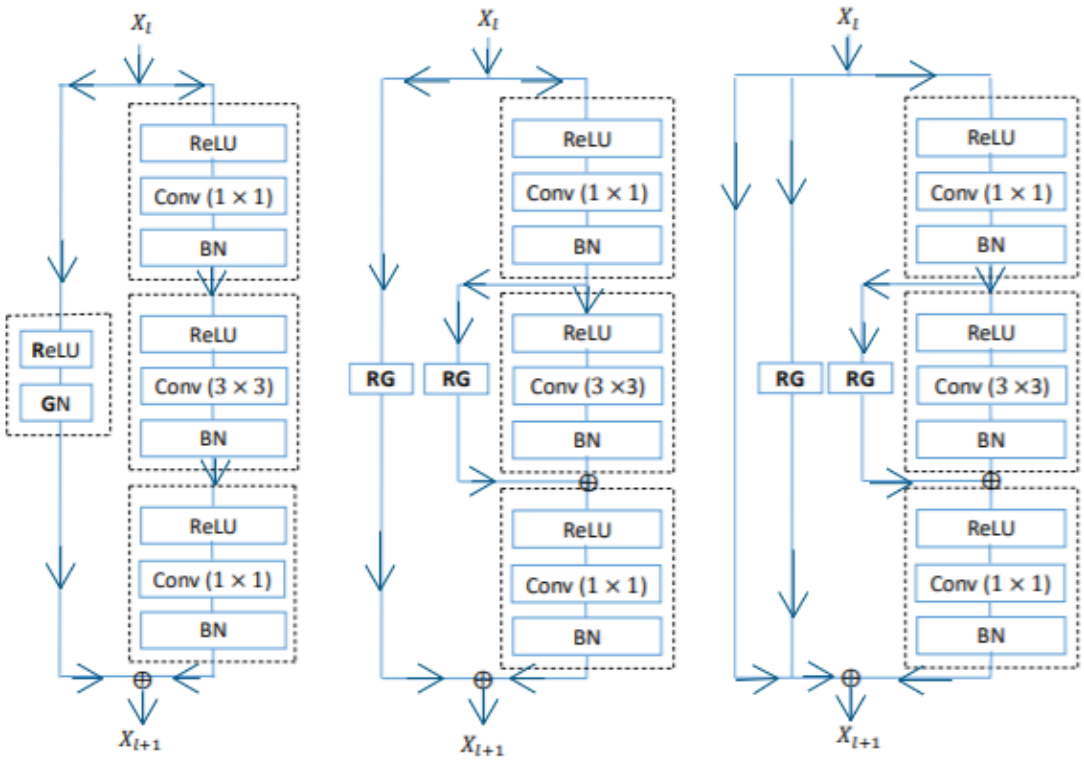


Figure 2.2: Here x_l and x_{l+1} are input and output of the lth layer. In the first model, the authors used one RG shortcut in place of identity skip connection (b-RGSNet [10]). In the second model, they used another inner RG shortcut along with the first model (d-RGSNet [10]). In the third model, they used an extra identity skip connection (Res-RGSNet [10]).

- **b-RGSNet:** In this model, the ReLU+Group Normalization shortcut (RG shortcut) is applied on the input of the block and added with the output of the last batch normalization.

- **d-RGSNet:** In this model, along with the b-RGSNet structure the authors added another RG shortcut to the output of the first batch normalization and added the output of the RG shortcut with the output of the second batch normalization.

- **Res-RGSNet:** This model is same as d-RGSNet with an extra identity skip connection like normal ResNet.

.

Some previous works have also been discussed in [6] as mentioned earlier. The experiments performed in [6], are as follows:

- **Constant Scaling :** In [6] the authors used scaling on the output of the both shortcut connection and final convolution layer of a block (i.e., in both f and F of (1.1)). They experimented with 0.5 scaling. First, they scaled only the skip connection which was not able to converge (by not able to converge they meant that the model has more than 20% top-1 error on test dataset), then they tried scaling both skip connection and residual block connection. The second model was able to converge with a much higher testing error than the baseline model (The baseline model is ResNet with identity skip connection).

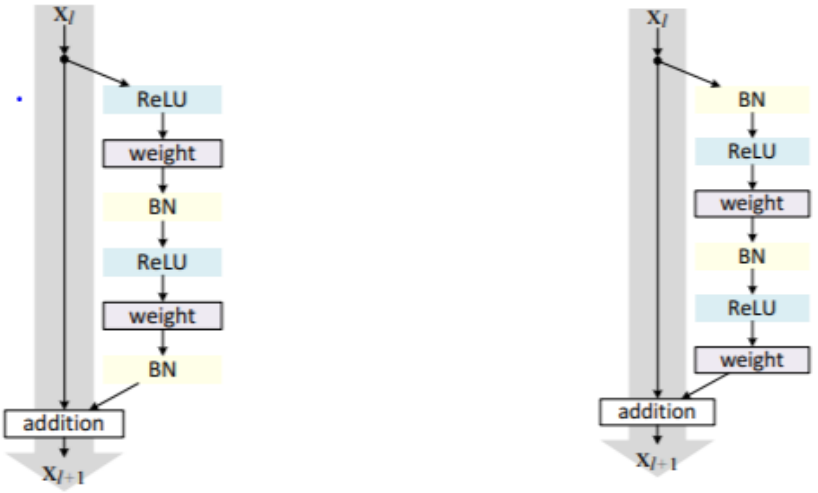
- **Gating :** The concept of gating was first introduced in [8]. The authors of [6] used a 1×1 convolution followed by a sigmoid function. If the gating value is $g(x)$ then they scaled the skip connection with $1 - g(x)$ and the residual block connection with $g(x)$ where x is the input. The authors of [6] were able to converge the model with a specific range of bias initialization, though the end result was worse than the baseline model.

- 1×1 **Shortcut:** The 1×1 convolution as skip connection worked well for relatively shallow model but it did not work well for deeper model (ResNet-110).

- **Dropout Shortcut :** The authors of [7] used Dropout [20] (In neural networks sometimes the hidden units gain highly correlated behavior. This phenomenon is known as co-adaption. It is better for a neural network to have hidden units which can detect independent features. Co-adaptation causes overfitting which affects the performance of the neural network at test time. Dropout is a regularization technique, which prevents co-adaptation. In Dropout at the time of training a weight is multiplied by zero with probability p and is multiplied by 1 with probability $1 - p$, where p is a hyperparameter) with $p = 0.5$, on the skip connection. As the expected value of the output of the skip connection becomes 0.5 times of that input which is like the constant scaling discussed above, Dropout also fails to meet the accuracy of the baseline model.

In [6] the authors proposed another version of ResNet model, called Pre-activation ResNet which performs better than the model proposed in [7]. Throughout this thesis we will be considering the ReLU-only pre-activation [6] model as our base ResNet model. In [6] the authors mentioned five different ResNet models together with the original model [7].We denote the skip connection output as $skip(x)$ where x is the input, batch normalization as BN, and convolution weights as $weight$.

- **original :** $ReLU(BN(weight(ReLU(BN(weight(x))))) + skip(x))$
- **BN after addition :** $ReLU(BN(weight(ReLU(BN(weight(x)))) + skip(x)))$
- **ReLU-Before-addition :** $ReLU(BN(weight(ReLU(BN(Weight(x))))))+skip(x)$
- **ReLU-only pre-activation :** $BN(weight(ReLU(BN(weight(ReLU(x)))))) + skip(x)$
- **full pre-activation :**$weight(ReLU(BN(weight(ReLU(BN(x)))))) + skip(x)$



(a) ReLU-only pre-activation

(b) Full pre-activation

Figure 2.3: Two versions of ResNet model

In pre-activation ResNet, the output after addition of the output of the identity skip connection and F in (1.1), is not affected by any other transformation, unlike the previous models which are mentioned above. In pre-activation ResNet F starts with either only ReLU or with batch normalization + ReLU. This means that input is passed through an activation function at the beginning of a block which is why these models are

called pre-activation ResNet. The model where only ReLU is used as pre-activation is called ReLU-only pre-activation and the one with batch normalization + ReLU is called full pre-activation. Among the five models mentioned above, we only need ReLU-only pre-activation and full pre-activation for our work.

Chapter 3

Our Work

Inspired by the work of [10] we first experimented with various combinations of activation functions and stabilizing methods. Like [10] we used bottleneck block for all ResNet model unlike [7], [6], who used basic block (a stack of two 3×3 convolutions) for ResNets with layers less than 50 and bottleneck block for ResNet with layers more than or equal to 50. Our work can be thought of as an extension of [10]. Though there are many activation functions to try, we considered only GELU [13] and Leaky ReLU. We experimented with different combinations of these functions and normalizations. For all cases, the performance was inferior to the model proposed in [10]. We also used full pre-activation as our basic ResNet model, which did not converge well.

There has been some work on the contribution of skip connection in ResNet performance [6], [21]. Our next intuition was to focus on the distribution of the input of a layer. We are considering that the features are independent of each other. We assume that the input is sampled from multivariate standard normal distribution $\mathcal{N}(0, I)$, where 0 represents the zero vector and I represents the identity matrix. By assuming that the input follows standard normal distribution we thought about focusing on the skewness of the data.

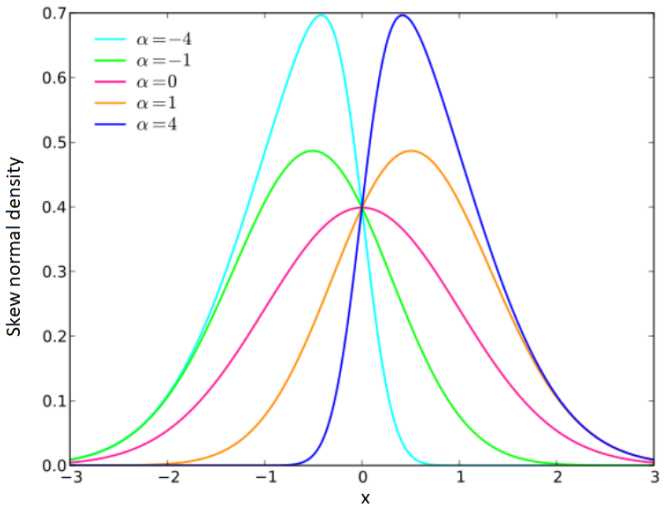


Figure 3.1: Skew normal density function

The probability density function of a skew normal distribution with a parameter α is denoted by $\mathcal{SN}(\alpha)$ if it has the density function

$$f(x) = 2\phi(x)\Phi(\alpha x) \tag{3.1}$$

where $\phi(x)$ denotes the standard normal density function and $\Phi(x)$ standard normal distribution function. If we set $\alpha = 0$, then we get the standard normal density function. So in a neural network, if we are planning to learn the parameter α and if the input follows standard normal distribution, then the network can always set α as zero resulting in no change in the distribution. In this context, we plan to learn α for each coordinate of the input vector. For that, we used a transformation that allows the model to choose how much skewness is best for this model at that layer or is it required at all. As mentioned in [22], suppose we have a n-dimensional random variable $(x_1, x_2, ..., x_n)$ following normal distribution with standardized marginals. The authors of [22] did not assume that the x_i's are independent. We are considering the part that we need for our work. Now let $x_0 \sim \mathcal{N}(0, 1)$ where x_0 is independent of the other x_i's. Let δ_i's $\in (-1, 1)$where $i = 1, 2, ..., n$. Define $z_j = \delta_j \mid x_0 \mid + \sqrt{(1 - \delta_j^2)}x_j$. Then $z_j \sim \mathcal{SN}(\lambda(\delta_j))$, where $\lambda(\delta) = \frac{\delta}{\sqrt{1-\delta^2}}$. The joint density function of Z is given by $2\phi_n(Z;\ \Omega)\Phi(\alpha^\intercal Z)$ where $Z \in \mathbb{R}^n$ and

$$\alpha^\intercal = \frac{\lambda^\intercal \Delta^{-1}}{\sqrt{1 + \lambda^\intercal \lambda}}$$

$$\Delta = diag(\sqrt{(1 - \delta_1^2)}, \sqrt{(1 - \delta_2^2)}, ... \sqrt{(1 - \delta_n^2)})$$

$$\lambda = (\lambda(\delta_1), \lambda(\delta_2), ...\lambda(\delta_n))^\intercal$$

$$\Omega = \Delta(I + \lambda\lambda^\intercal)\Delta$$

Here $\phi_n(Z;\ \Omega)$ denotes the multivariate normal distribution with standardized marginals and correlation matrix Ω.

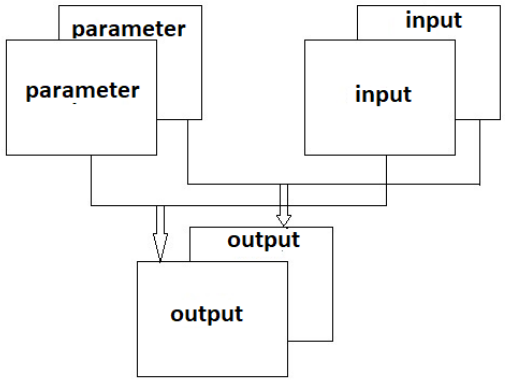


Figure 3.2: This figure represents how our proposed skip-connection works. Here we are multiplying the input tensor and the parameter tensor channel wise. The parameter tensor has the same dimension as the input tensor.

We use the transformation defined by z_j above to transform the distribution of the input. We propose a transformation in which we multiply (Hadamard product) a learnable matrix channel wise and sample a number x_0 from $\mathcal{N}(0, 1)$ in each pass and perform the transformation. Our proposed transformation is defined as:

$$Z = \delta \mid x_0 \mid + \sqrt{(1 - \delta^2)}X,$$

where δ is the learnable matrix, X is the input matrix. Since the matrix norm of matrices which were multiplied is less than 1, the performance of the model using only this transformation was very poor. We then used the linear combination of the two skip connections, one that is our proposed transformation and the second one is the transformation used in [10]. We also added the inner skip connection which was used in d-RGSNet. The performance of the resulting model was superior to Res-RGSNet [10] model which uses the identity skip connection also.

Chapter 4

Implementation Details and Results

4.1 Implementation Details

We mainly focused our work on CIFAR-100 and CIFAR-10 datasets.

4.1.1 Datasets

- **CIFAR-100:** The CIFAR-100 [2] dataset has 60000 colour images of size 32×32, in 100 classes. Each class has 600 images. The training set has 50000 images and test set has 10000 images.
- **CIFAR-10:** The CIFAR-10 [2] dataset is almost same as CIFAR-100, except it has 10 classes and each class has 6000 images.

For CIFAR-100 and CIFAR-10 datasets [2] at first, we used a 3×3 convolution with stride 1 followed by a batch normalization and ReLU. The dimension of the input was $3 \times 32 \times 32$. We used four convolution groups ($conv2, conv3, conv4, conv5$) in our experiment like traditional ResNet. After each convolution group, downsampling takes place (except the last one). For downsampling, we used a 3×3 convolution with stride 2. We ran each model for 200 epochs starting with initial learning rate 0.1 and decreasing the learning rate by a factor of 10 at 82, 124, 164 epochs respectively. We set the weight decay to 0.0005 and batch size to 128.

4.2 Experimental setup

We consider RGSNet as our baseline model. Before coming up with the transformation we looked into some other activation functions which we used with or without some stabilizing techniques, as a skip connection. we tested each combination for ResNet-26. Once the performance improved then we tested the architecture with ResNet-50.

- We started with GELU [13]. We replaced ReLU and group normalization shortcut with GELU. The performance was inferior to the baseline model.
- We used GELU and group normalization in d-RGSNet the performance was inferior to the baseline model.
- We used full pre-activation ResNet as the base model of RGSNet. The performance was further inferior to our baseline result.
- We used group normalization and batch normalization together in b-RGSNet model. The performance was almost the same with the real b-RGSNet.
- We used Leaky ReLU in place ReLU in b-RGSNet. Although the performance was inferior to our baseline model but the difference in performances was not much significant.
- Next, we used our skip connection. We looked into a couple of combinations and finally came up with a combination almost similar to d-RGSNet. We used the linear combination of our skip connection and the RG-shortcut where higher weight is given to our skip connection, as the outer skip connection and for the inner skip connection which starts after the first 1×1 convolution and ends after the 3×3 convolution, we used the RG-shortcut. This model has outperformed d-RGSNet and has achieved a performance as good as Res-RGSNet (which uses the identity skip connection) or even better.

4.3 Simulation Results

For normalization of CIFAR-100 we used ((0.507, 0.487, 0.441),(0.267, 0.256, 0.276)) and for CIFAR-10 we used ((0.4914, 0.4822, 0.4465), (0.247, 0.243, 0.261)). Here the first tuple represents the mean of each channel and the second tuple represents the standard deviation. For CIFAR-100 we mainly focus on models with 26, 50 and 152

layers because we want to compare deep traditional ResNet and shallow RGSNets and our model. It has already been shown that Res-RGSNet-50 is as good as ResNet-200 [10]. For CIFAR-10 we considered the results published in [6]. We ran the RGS models and our models with 26 layers on CIFAR-10. The results of our experiments are shown in Table 4.1 and Table 4.2.

Due to the lack of computational facility, we were not able to simulate the performance of deeper models. We were only able to get result for one such model with 50 layers.

Table 4.1: Comparison on CIFAR-100 dataset

Model Type	Depth	Skip connection	top-1 error
ResNet	26	identity	23.17
ResNet	50	identity	21.43
ResNet	152	identity	20.08
b-RGSNet	26	RG shortcut	21.32
d-RGSNet	26	RG shortcut + inner RG shortcut	20.71
Res-RGSNet	26	RG shortcut + inner RG shortcut + identity	20.49
d-RGSNet	50	RG shortcut + inner RG shortcut	20.00
Res-RGSNet	50	RG shortcut + inner RG shortcut + identity	19.59
d-RGSNet	26	GELU	21.6
d-RGSNet	26	GELU + group normalization	21.78
d-RGSNet	26	GELU + group normalization + batch normalization	23.31
b-RGSNet	26	ReLU + group normalization(full Preactivation ResNet)	26.84
b-RGSNet	26	Leaky ReLU + group normalization	24.86
b-RGSNet	26	0.7× proposed transformation+0.3× RG shortcut	**20.93**
d-RGSNet	26	0.6×proposed transformation+0.4×RG shortcut + RG inner shortcut	**20.54**
d-RGSNet	26	0.7×proposed transformation+0.3×RG shortcut + RG inner shortcut	**20.2**
d-RGSNet	50	0.6×proposed transformation+0.4×RG shortcut + RG inner shortcut	**19.07**

Table 4.2: Comparison on CIFAR-10 dataset

Model Type	Depth	Skip connection	top-1 error
ResNet	110	identity	6.37
ResNet	1001	identity	4.92
b-RGSNet	26	RG shortcut	4.56
d-RGSNet	26	RG shortcut+RG inner shortcut	4.52
b-RGSNet	26	0.7×proposed transformation+0.3×RG shortcut	4.55
b-RGSNet	26	0.6×proposed transformation+0.4×RG shortcut	**4.46**
d-RGSNet	26	0.7×proposed transformation+0.3×RG shortcut+RG inner shortcut	4.67
d-RGSNet	26	0.8×proposed transformation+0.2×RG shortcut+RG inner shortcut	4.64
d-RGSNet	26	0.6×proposed transformation+0.4×RG shortcut+RG inner shortcut	**4.28**

Chapter 5

Analysis of Results and Conclusion

5.1 Analysis of Results

Proposed transformation with ResNet-50 as the base model has achieved 19.17% top-1 error on CIFAR-100. In fact, our proposed model outperforms state-of-the-art Res-RGSNet-50 model (19.59% top-1 error on CIFAR-100 dataset) which uses the identity skip connection to boost its performance. ResNet-152 achieved 20.08% top-1 error on the same dataset. So the proposed model has successfully provided better classification performance than state-of-the-art ResNet models. Also, ResNet-110 and ResNet-1001 achieved respectively 6.37% and 4.92% top-1 error on CIFAR-10 while our proposed model achieves 4.28% top-1 error with 26 layers. Previously all the works have stated that skip connection gives a boost in the performance. We have shown that even without using the identity skip connection we can achieve significantly higher classification performance.

5.2 Conclusion

After studying the work presented at the BMVC, 2019 [10] we thought it might be a good idea to work with the channel transformation because in RGSNets, group normalization also performs its operation along the channel axis only. And since ReLU changes the initial normal distribution to half-normal distribution we thought about adding the skewness to the model. So inter channel transformations can be seen as a possible replacement of the identity skip connection of ResNets.

Chapter 6

Future work

In the future, we are planning to continue the study in the following directions.

- We would like to investigate the classification performance of the proposed model on ImageNet [1] which could not be possible to obtain as a part of the present study due to time and computational facility constraint. Since the dimension of its samples is $3 \times 224 \times 224$, it might be a good idea to downsample the image first then use our proposed transformation and then upsample with transposed convolution to match the input dimension.
- We would also like to obtain the performance of the proposed model on MSCOCO [3] and PASCAL-VOC [4] datasets towards object detection task.
- Gating mechanism will be introduced in the proposed model to enhance its performance similar to the approach used in [8].
- We would also like to explore the use of some alternative transformations capable of capturing the inter-channel dependencies.

Chapter 7

Bibliography

[1] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[2] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European Conference on Computer Vision*. Springer, 2014, pp. 740–755.

[4] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010.

[5] S. Zagoruyko and N. Komodakis, "Wide residual networks," *ArXiv preprint arXiv:1605.07146*, 2016.

[6] K. He, X. Zhang, S. Ren, and J. Sun, "Identity mappings in deep residual networks," in *European Conference on Computer Vision*. Springer, 2016, pp. 630–645.

[7] ——, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[8] R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks," *CoRR abs/1505.00387 (2015), ArXiv preprint arXiv:1505.00387*, 2015.

[9] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.

[10] C. Zhang, F. Rameau, S. Lee, J. Kim, P. Benz, D. M. Argaw, J.-C. Bazin, and I. S. Kweon, "Revisiting residual networks with nonlinear shortcuts." in *Proceedings of the British Machine Vision Conference (BMVC)*, 2019, p. 12.

[11] H. Ren, W. Wang, X. Qu, and Y. Cai, "A new hybrid-parameter recurrent neural network for online handwritten chinese character recognition," *Pattern Recognition Letters*, vol. 128, pp. 400–406, 2019.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1026–1034.

[13] D. Hendrycks and K. Gimpel, "Gaussian error linear units (gelus)," *ArXiv preprint arXiv:1606.08415*, 2016.

[14] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[15] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," ArXiv e-prints, 2015.

[16] D. Ulyanov, A. Vedaldi, and V. Lempitsky, "Instance normalization: The missing ingredient for fast stylization," *ArXiv preprint arXiv:1607.08022*, 2016.

[17] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *ArXiv preprint arXiv:1607.06450*, 2016.

[18] Y. Wu and K. He, "Group normalization," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 3–19.

[19] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 901–909.

[20] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *ArXiv preprint arXiv:1207.0580*, 2012.

[21] T. Liu, M. Chen, M. Zhou, S. S. Du, E. Zhou, and T. Zhao, "Towards understanding the importance of shortcut connections in residual networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 7892–7902.

[22] A. Azzalini and A. D. Valle, "The multivariate skew-normal distribution," *Biometrika*, vol. 83, no. 4, pp. 715–726, 1996.