PADERBORN UNIVERSITY
The University for the Information Society

Faculty for Computer Science, Electrical Engineering and Mathematics

Faculty for Computer Science, Electrical Engineering and Mathematics
Paderborn University
Communications Engineering
Prof. Dr.-Ing. Reinhold Häb-Umbach

Seminar Report

Deep Residual Learning

by

Varun Nandkumar Golani
Matr.-No.: -

Supervisor: Prof. Dr. Reinhold Häb-Umbach, M. Sc. Christoph Böddeker
Filing date: 21.03.2021

Declaration

This Seminar Report, with the title "Deep Residual Learning", is the result of my own work and includes nothing that is the outcome of work done in collaboration, except where specifically indicated. It has not been published yet or submitted, in whole or in part, for a degree at any other university.

Paderborn, 21.03.2021 _____

(Varun Nandkumar Golani)

Contents

1 Introduction

Neural Network(s) (NN) have been in the literature for quite some time now [Wik21] and recently Deep Neural Network(s) (DNN) are gaining a lot of attention because of the new technique Deep Residual Learning (DRL) to train deeper networks which produce state-of-the-art results as compared to other baseline models on various tasks like image recognition [He+16], speech recognition [HDHU16], image steganalysis [WZL18], small footprint keyword spotting [TL18], age and gender estimation [Lee+18] etc. The questions that come to mind are Can't we use the same training procedure as that of a NN?, Why do we need DRL? To answer the first question, yes DNN can be trained with the same training procedure as that of NN but as networks get deeper the accuracy gains from the depth of the network become insignificant or the depth affects the accuracy negatively and this problem is called the degradation problem (not overfitting) [He+16]. To answer the second question, to solve this degradation problem DRL is used [He+16]. DRL is described in detail in Section 2.1.

Let's discuss the degradation problem [He+16] in a bit more detail. Consider two NN, Network 1 which has a shallower network architecture than Network 2 and Network 2 has extra identity mapping layers and the rest of the layers are same as the learned Network 1. Network 2 is called a constructed solution because it uses learned layers from Network 1. Now we know that this constructed solution (Network 2) exists, it should imply that network with the same depth as Network 2 (or deeper than Network 2) should produce no lower accuracy as compared to Network 1. But as per [He+16], the solvers are not able to find better solutions than the constructed solution (Network 2) or they are not able to find better solutions in feasible time.

The report is structured as follows. In Chapter 2 the fundamental concepts required to understand this report are described. In Chapter 3 some literature related to the topic DRL are summarized. In Chapter 4 the DRL architectures used for some tasks are explained in detail. In Chapter 5 the experiments related to one of the task image recognition [He+16] (Section 4.1) are presented. Chapter 6 summarizes and concludes this topic.

2 Fundamentals

2.1 Residual Learning

Residual Learning (RL) [He+16] is a recent technique in the literature to train DNN, and it is effective technique to train deeper networks.

Consider $\mathcal{H}(x)$ to be the unknown mapping function that can be approximated by some stacked layers (or the entire network), here x denotes the input to the first layer. Instead of approximating $\mathcal{H}(x)$, the network approximates a residual function $\mathcal{F}(x) = \mathcal{H}(x) - x$. The original function $\mathcal{H}(x)$ can then be obtained by $\mathcal{F}(x) + x$. The normal intuition is the deeper the network, the better should be the performance but there is a degradation problem which is described in [He+16] and due to this problem the observed behavior is quite the opposite. RL solves the degradation problem, and it is easier to optimize the residual function $\mathcal{F}(x)$ than the original function $\mathcal{H}(x)$. The intuition behind the working of the residual connections is that even if the some layers don't learn anything i.e. weights and biases for these layers are zero, due to the residual shortcut connections the same activation (from the layer before these zero weights and biases layers) is carried to the further layers and the depth doesn't hurt the performance of the network and in turn solves the degradation problem [He+16].

The Figure 2.1 depicts a two layer residual building block with ReLU activation. It is assumed that the dimensions of $\mathcal{F}(x)$ and x (identity mapping as in Figure 2.1) are the same and formally it can be expressed as in equation 2.1. If this assumption is not true then a linear projection W_s is carried out to match the dimensions and formally it can be expressed as in equation 2.2.

$$y = \mathcal{F}(x, \{W_i\}) + x \tag{2.1}$$

where x and y are the input and the output vectors respectively, $\mathcal{F}(x, \{W_i\})$ is the residual function that is to be learned.

$$y = \mathcal{F}(x, \{W_i\}) + W_s x \tag{2.2}$$

2.2 Convolutional Neural Networks

Convolutional Neural Network(s) (CNN) [WZL18] have been a popular choice for many image related tasks [He+16; SZ15; WZL18] and the DRL architectures discussed in Chapter 4 use some/all the following layers.

2.2.1 Convolution Layer

This layer performs convolution of input images with one or many filters (kernel size is mostly set to 3×3, 5×5, 7×7) and in turn these new feature maps can be used for further processing

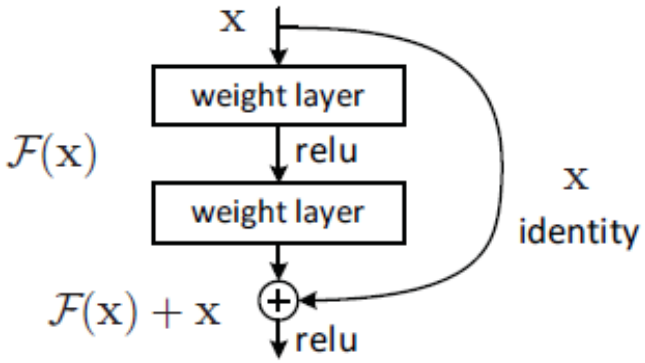

Figure 2.1: A building block for RL. Image Source: [He+16].

by the next layer (see equation 2.3).

$$F_j^{l+1} = \sum_i W_{ij}^l F_i^l + b_j^l \tag{2.3}$$

where F_j^l is the j^{th} feature map of the l^{th} layer, W_{ij}^l is the convolutional kernel and b_j^l is the bias. The filters W_{ij}^l, bias b_j^l can be learned by the back-propagation algorithm. Different structures in images can be extracted by the learned filters for accurate modeling.

2.2.2 Nonlinear Mapping/Activation Layer

This layer performs the nonlinear transformation of the input feature map and can be expressed as in equation 2.4.

$$F_j^{l+1} = f(F_j^l) \tag{2.4}$$

where $f(.)$ denotes the point-wise nonlinear function, for e.g. ReLU, sigmoid, tanh etc. Without the nonlinear mapping, any NN is equivalent to one single layer NN and that's why nonlinear mapping is required for CNN. Nonlinear mapping is useful for extracting more complex correlations in images.

2.2.3 Pooling Layer

This layer reduces the dimension of the input feature map i.e. it reduces the size of the extracted features and can be expressed as in equation 2.5.

$$F_j^{l+1} = pool(F_j^l) \tag{2.5}$$

where $pool(.)$ is the pooling function. Pooling aggregates the input feature maps and makes the representation compact. Normally two types of pooling are used in CNN models: maximum pooling and average pooling. Maximum pooling outputs the maximum value in a local region. Average pooling outputs the average value of a local region. Pooling can capture large distance correlations in images as it makes the feature map smaller.

2.2.4 Batch Normalization Layer

As the name suggests, this layer normalizes all the data inputs x_i present in a training batch $\mathcal{B}$ to y_i. This can be expressed as in equation 2.6.

$$y_i = \gamma \hat{x}_i + \beta \tag{2.6}$$

where γ and β are the parameters used for batch normalization and $\hat{x}_i$ is expressed as in equation 2.7.

$$\hat{x}_i = \frac{x_i - E_B(x_i)}{\sqrt{Var_B(x_i)}} \tag{2.7}$$

where $E_B(x_i)$ and $Var_B(x_i)$ are the mean and variance of x_i with respect to the training batch $\mathcal{B}$. A NN using Batch Normalization is not sensitive to the parameter initialization and the speed of convergence is faster than a NN without Batch Normalization.

3 Related Work

He et al. (2016) [He+16] proposed a residual DNN architecture of up to 152 layers for Image Recognition on the ImageNet dataset and ensemble of these residual networks (of different depths) produced state-of-the-art results with an error of 3.57% on the ImageNet test set. Many experiments were conducted on the CIFAR-10 dataset and residual networks of depth over 100 and 1000 layers were trained. Heymann et al. (2016) [HDHU16] used a Wide Residual BLSTM Network(s) (WRBN) as a back-end for Speech Recognition and overall the complete system achieved the best reported results as a single system on the CHiME-4 challenge. Wu et al. (2018) [WZL18] trained a deeper CNN model with residual learning which is called Deep Residual learning based Network(s) (DRN) for the task Image Steganalysis, and it can recognize the state-of-the-art steganographic algorithms with high accuracy and outperforms some baselines in Image Steganalysis. Tang et al. (2018) [TL18] used DRL and dilated convolutions for the task Small Footprint Keyword Spotting and the model achieved state-of-the-art performance when compared to Google's CNN models on the Google Speech Commands benchmark dataset and as per the experiments in [TL18] on the depth and width of the model, all of their models have fewer parameters than Google's CNN models and most of them outperform the Google's CNN models. Lee et al. (2018) [Lee+18] proposed a deep residual network for the task age and gender estimation and their network architecture is very similar to Residual Network (ResNet)-50 [He+16] and on the Adience benchmark [LH15], their baseline outperforms the model in [LH15].

4 Deep Residual Learning Tasks & Architectures

4.1 Image Recognition

The residual DNN architecture that is used for image recognition from [He+16] is described here and can be seen in Figure 4.1. Each convolution block in the architecture can be formally represented as $f \times f$ conv, n_c where f stands for the filter size and n_c stands for the number of channels. The layers in the architecture are a 7×7 conv, 64 layer, a pooling layer, many 3×3 conv layers with increasing channel sizes up to 512, average pooling layer and a fully connected layer with softmax which outputs 1000 units which are the final outputs of the DNN. The dark/dotted curved arrows indicate the residual connections, the difference is that the dark arrows are the connections where no dimension change is required (see equation 2.1) and the dotted arrows are the connections where an increase in dimension is required. Two options are discussed in [He+16] to increase dimensions: (A) Pad extra zeros for increasing dimensions while keeping the rest of the input the same, this option does not introduce any extra parameter. (B) Perform linear projection (see equation 2.2). To elaborate option (B), let the input to the residual connection be of dimension a and the required output be of dimension b, therefore W_s will be a matrix of dimension $b \times a$ and addition of two b dimension vectors (one vector from the output of the previous layer + one vector from the product of W_s and a dimensional input) will result in a b dimension vector.

The number of filters in each convolutional layer is the same if the output feature map size is the same (for instance in Figure 4.1, the first six 3×3 conv, 64 layers, the number of filters are 64 for all) else if the output feature map size is halved then the number of filters is doubled (for instance in Figure 4.1, the next eight 3×3 conv, 128 layers, the number of filters are $64 \times 2 = 128$ for all). Downsampling is performed by the convolutional layers with a stride of 2. This 34 layer residual network (3.6 billion FLOPs) [He+16] has less filters and lower complexity as compared to one of the baselines VGG nets (19.6 billion FLOPs for VGG-19) [SZ15].

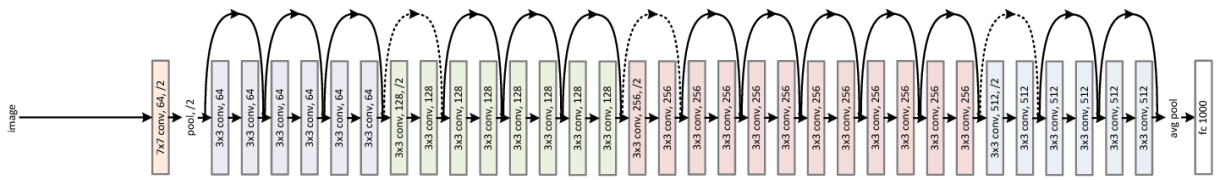

Figure 4.1: Architecture of a 34 layer (3.6 billion FLOPs) residual network used for image recognition. Image Source: [He+16].

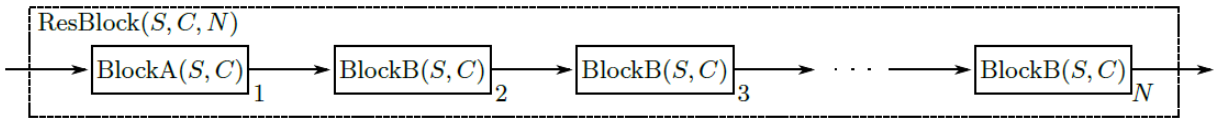

Figure 4.2: An in-depth view of the ResBlock(S,C,N) in Figure 4.4. Here S denotes the stride, C denotes the number of output channels and N denotes the number of blocks in ResBlock(S,C,N). Image Source: [HDHU16].

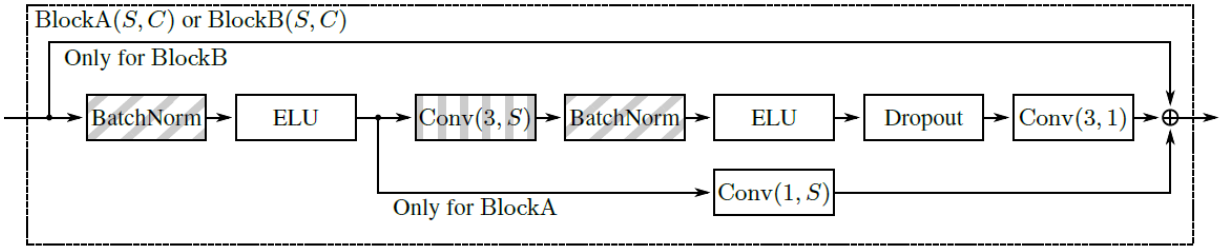

Figure 4.3: An in-depth view of BlockA(S,C) and BlockB(S,C) in Figure 4.2. In this Figure Conv(A,S) blocks are used where A stands for the filter size $A \times A$, S stands for the consecutive striding. A zero padding of size $(A-1)/2$ in both the directions is also used in this Conv(A,S) block. Image Source: [HDHU16].

4.2 Speech Recognition

The WRBN [HDHU16] is used as a back-end for speech recognition and can be seen in Figure 4.4. The architectures/block diagrams in figures 4.2, 4.3, 4.4 will be explained in the following subsections.

4.2.1 Wide Residual BLSTM Network(s)

The architecture/structure of WRBN can be seen in Figure 4.4. Firstly, a Conv layer with filter size 3×3 with stride S is used. Following the Conv layer, three ResBlock blocks with different strides are used. Starting with 80, the number of output channels are doubled for every ResBlock after that i.e. for the second (160) and the third (320) ResBlock. A Residual Block(s) (RB) is discussed in detail in the next Subsection 4.2.2. The output after all the RB and the Batch Normalization layer is a tensor with dimension of $B \times 320 \times 10 \times T$ where B is the mini-batch size and T is the number of frames. The input to the first BLSTM layer is of the dimension $T \times B \times 320$ which is the result of weighting and combining the previous output for each channel. Two BLSTM layers are used in this architecture/structure with 512 units for both directions, for the first BLSTM layer the output is merged by addition and for the second BLSTM layer the output is merged by concatenation. To avoid overfitting, Dropout is used for both the BLSTM layers. There is an additional Dropout used after the second BLSTM layer for hidden-hidden transitions. Finally, there are two feed-forward layers Batch Normalization and Exponential Linear Unit (ELU) activation.

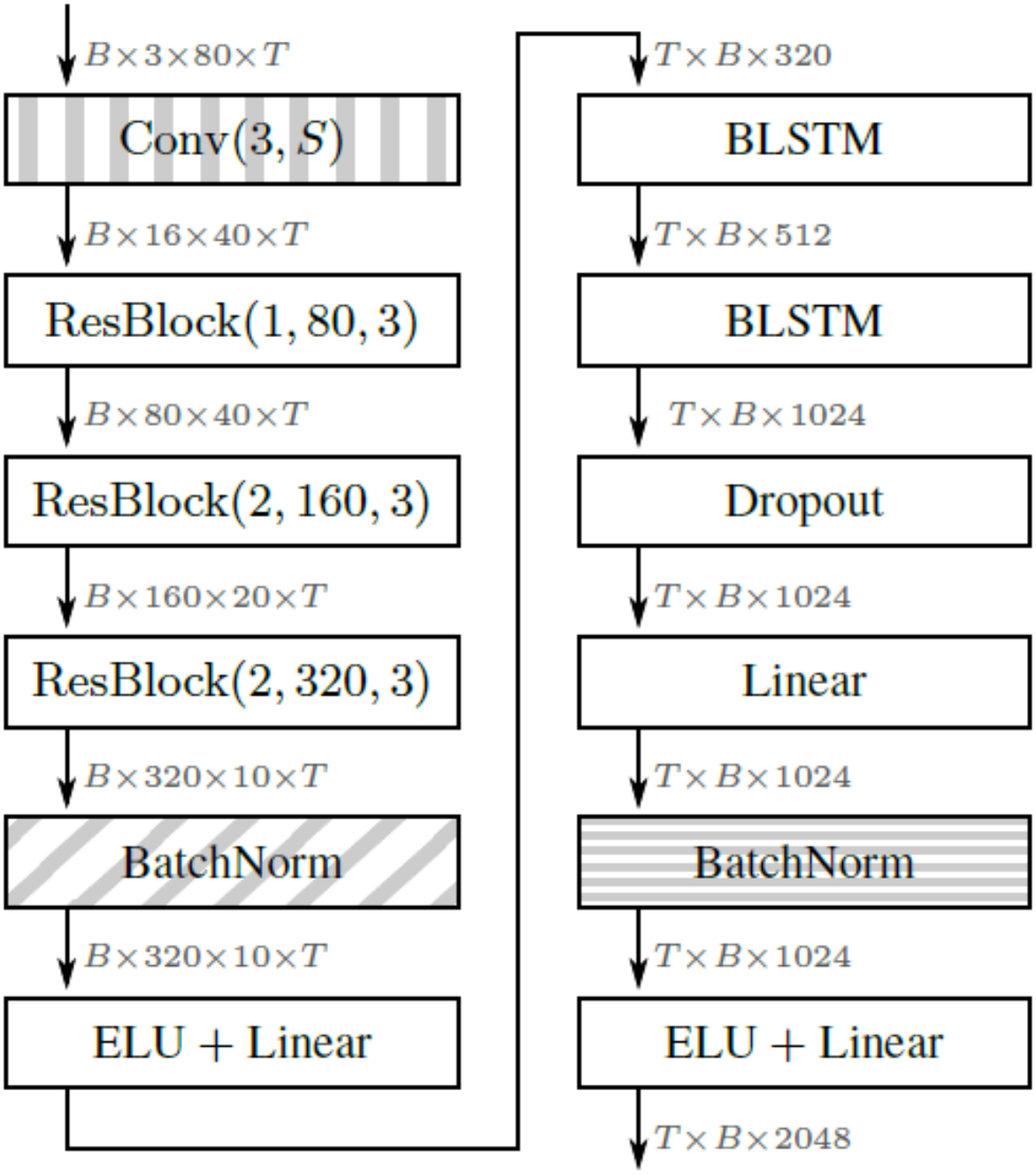

Figure 4.4: An WRBN architecture overview of the back-end used for speech recognition. The dimensions on each arrow are the dimensions of the tensors where B denotes the mini-batch size and T denotes the number of frames. Image Source: [HDHU16].

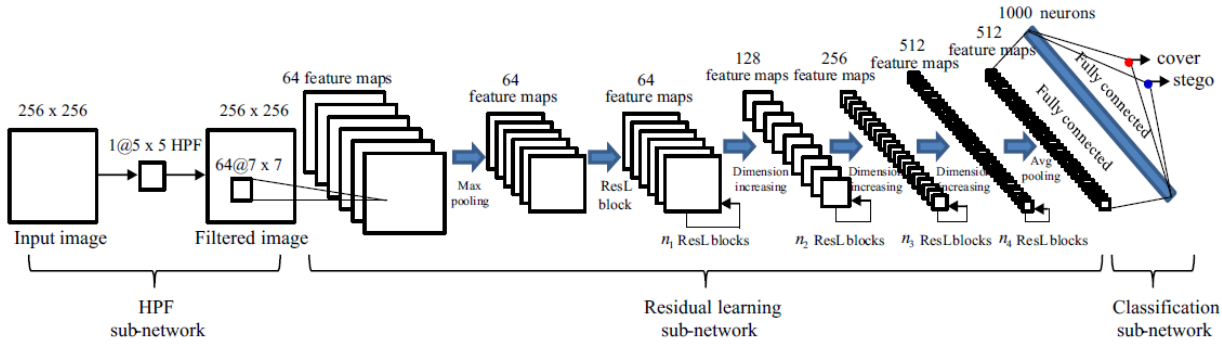

Figure 4.5: Architecture of DRN for Image Steganalysis. Image Source: [WZL18].

4.2.2 Residual Block and BlockA/BlockB

A RB (See Figure 4.2) consists of one block of BlockA at the start and $N-1$ blocks of BlockB. As it can be seen from Figure 4.3, the difference between BlockA and BlockB is of the residual connection. BlockA (BlockB) has a residual connection which is similar to the dotted (dark) arrows as discussed in Section 4.1. For BlockA, a convolution operation with filter size of 1×1 with stride S is performed. The vertically striped block Conv$(3, S)$ and the diagonally striped BatchNorm blocks are the same as in Figure 4.4. The Batch Normalization block normalizes the output from the previous convolution. The ELU block applies an ELU non-linearity to the output from the previous block. Before the Conv$(3, 1)$ block, Dropout is used with $p = 0.5$.

4.3 Image Steganalysis

The task in Image Steganalysis is to differentiate between images with and without secret messages [WZL18]. A DRN [WZL18] architecture used for Image Steganalysis is discussed here and can be seen in Figure 4.5. The complete network is divided into three sub-networks for different tasks and are described in the following subsections.

4.3.1 High-Pass Filtering Sub-Network

High-Pass Filtering (HPF) sub-network extracts the noise components from input cover/stego images. As discussed in [WZL18], the output image obtained by preprocessing with HPF have a narrow dynamic range and a large signal-to-noise ratio (SNR) ratio between the weak stego and the image signal. This is the reason to input the extracted noise components to the next sub-network which is described in Subsection 4.3.2. Formally, let I denote the input image, k be the HPF kernel and the noise component n can be expressed as in equation 4.1. In the Figure 4.5, the dimension of the kernel denoted as $p@q \times q$ means p filters of the size $q \times q$. The dimension of the kernel for HPF is $1@5 \times 5$. The KV kernel is used and k can be expressed as in equation 4.2.

$$n = I * k \tag{4.1}$$

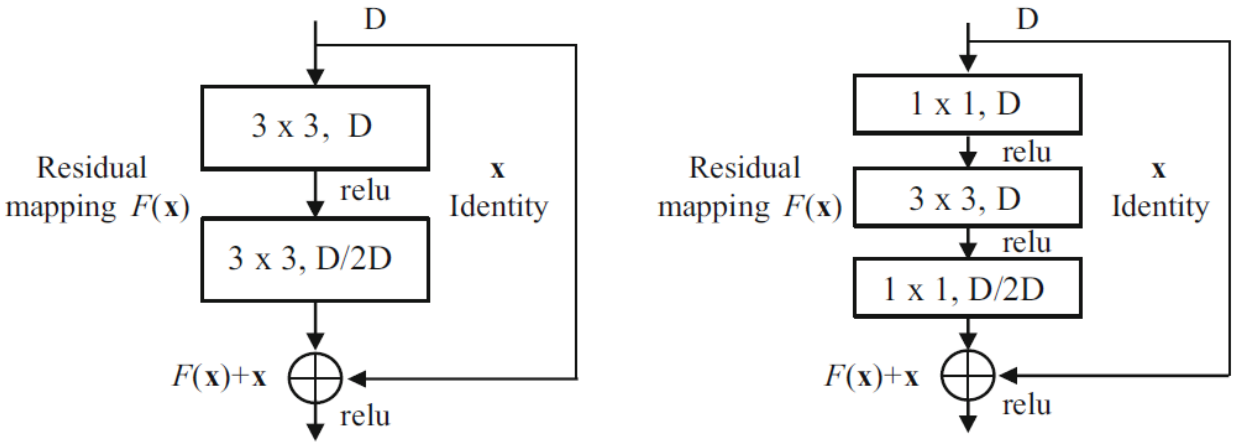

Figure 4.6: A non-bottleneck and a bottleneck building block for DRN. In both the blocks, the dimension of the output can be kept the same (doubled) and the block would be called ordinary residual (dimension increasing). Image Source: [WZL18].

where $*$ is the convolution operation.

$$k = \frac{1}{12}\begin{pmatrix} -1 & 2 & -2 & 2 & -1 \\ 2 & -6 & 8 & -6 & 2 \\ -2 & 8 & -12 & 8 & -2 \\ 2 & -6 & 8 & -6 & 2 \\ -1 & 2 & -2 & 2 & -1 \end{pmatrix} \tag{4.2}$$

4.3.2 Residual Learning Sub-Network

This sub-network automatically extracts features from cover and stego images for classification in the further sub-network which is described in Subsection 4.3.3. Convolution (dimension of the kernel is $64@7 \times 7$) is applied on the filtered image from the previous sub-network generating 64 feature maps which are used for further processing. Following every convolution layer, there is a ReLU activation layer, a maximum pooling layer and a batch normalization layer which are not shown in Figure 4.5.

A non-bottleneck block (see Figure 4.6 (a)) consists of two 3×3 convolution blocks and a bottleneck block (see Figure 4.6 (b)) consists of two 1×1 convolution blocks and a 3×3 convolution block. As per [He+16; WZL18], the bottleneck blocks are used for deeper networks and reduce the time taken to train the deeper networks. The feature map is down-sampled by a factor of 2 if the feature maps are doubled for the dimension increasing block [He+16; WZL18].

The above principle is used for further processing in the DRN and the feature maps are transformed from 64 to 512 (see Figure 4.5).

4.3.3 Classification Sub-Network

The role of this sub-network is to map the extracted features from the previous sub-network (Subsection 4.3.2) to binary labels. This sub-network uses a fully connected neural network model. The number of neurons is set to 1000.

4.4 Small Footprint Keyword Spotting

The task of Keyword Spotting is to recognize a small set of predefined keywords in user's speech which is normally used for a mobile phone intelligent agent or a smart home consumer device like Amazon Alexa [TL18]. A footprint of a model is measured using two quantities: the number of parameters and the number of multiplications required for one full feedforward pass [TL18]. Combining the two definitions together for the task of Small Footprint Keyword Spotting, the preferred models are those that can solve the task of Keyword Spotting and have lesser value for the two quantities described above. Before feeding the input to the DNN with DRL (see Figure 4.7), the feature extraction and the input preprocessing takes place and these processed Mel-Frequency Cepstrum Coefficient (MFCC) frames are fed as inputs to the network (for more details about feature extraction and input preprocessing refer to [TL18]).

Let's discuss the zoomed in part in Figure 4.7. The first 3×3 convolution layer has weight component $W \in \mathbb{R}^{(m \times r) \times n}$ (no bias component) where m is the width of the feature maps, r is the height of the feature maps and n denotes the number of feature maps. After the convolution layer, there is a ReLU activation which is followed by a batch normalization layer. With RB, a (d_w, d_h) convolution dilation is also used to expand the receptive field of the network and is depicted in Figure 4.8. The very first convolution layer (same weights W as described above) in Figure 4.7 is used for matching the dimensions as the input of the residual connection should have the same dimensions as the output. An extra convolution layer with batch normalization layer is also added after the residual blocks (see Figure 4.7).

The base model is called `res15` (15 layer DNN with DRL) consists of six RB, $n = 45$ feature maps. Dilation is used with $d_w = d_h = 2^{\left\lfloor \frac{i}{3} \right\rfloor}$ and as a result the total receptive field is 125×125. Normally, in residual network architectures the whole output after each layer is zero padded and at the end it is average-pooled and forwarded to the fully-connected softmax layer. Many variants of these DNN with DRL can be obtained by changing the width (number of feature maps) and depth (depth of the network) of the model. The considered depth variants are `res8`, `res26` including `res15` and the considered width variants are `res8-narrow`, `res26-narrow`, `res15-narrow` [TL18]. The `-narrow` suffix is appended to the model if the width of the model is reduced and for these narrow variants $n = 19$. The most compact model in [TL18] is the `res8-narrow` which has 19.9K parameters and 5.65M number of multiplications required for one full feedforward pass. There is always a tradeoff between high detection accuracy and a small footprint [TL18].

4.5 Age and Gender Estimation

The inputs to the deep residual networks [Lee+18] (see Figure 4.9) are images (faces) obtained by using face detectors and after preprocessing the size of the input images are 224×224.

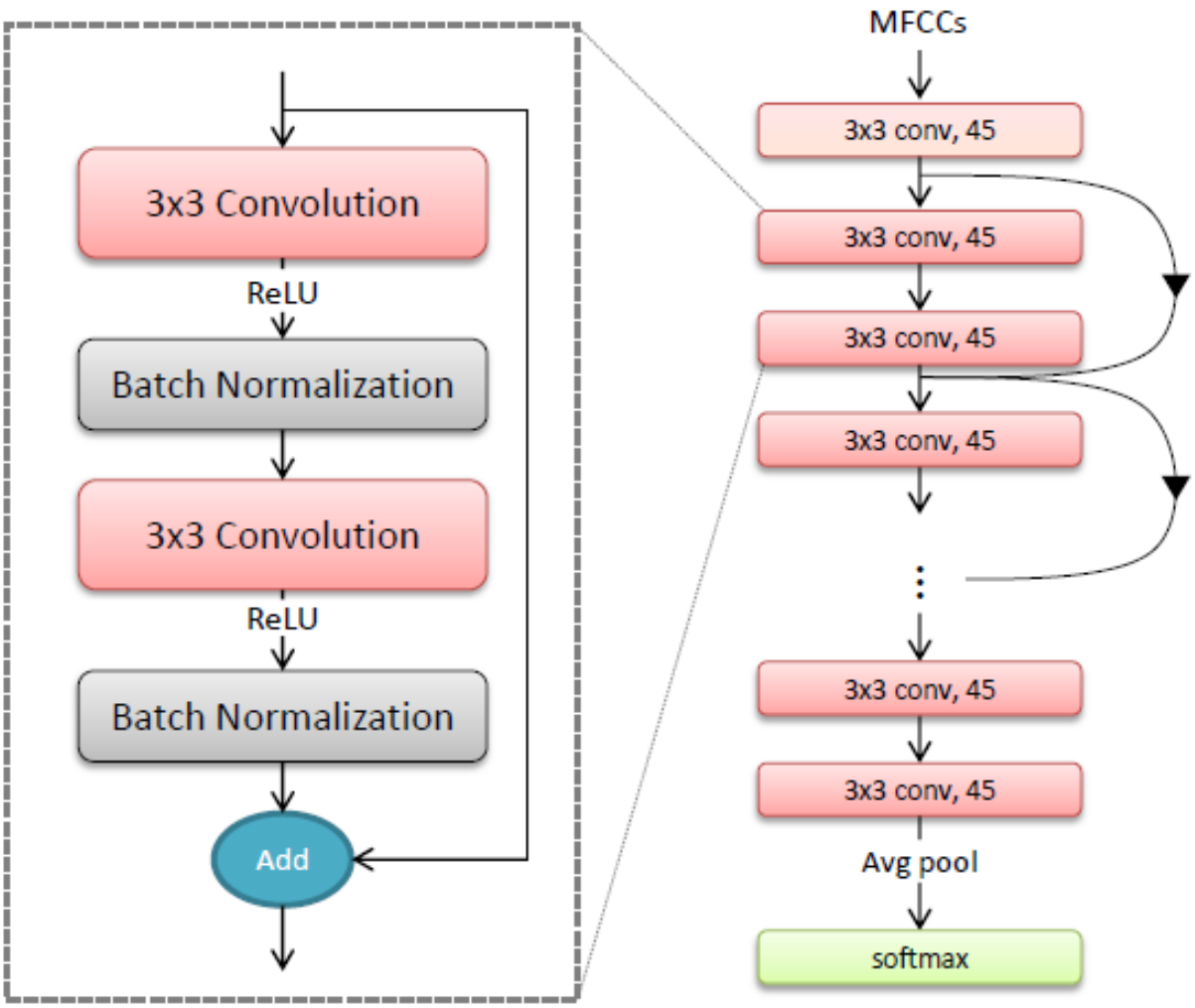

Figure 4.7: Architecture of DNN with DRL for Small Footprint Keyword Spotting with a zoomed in RB. Image Source: [TL18].

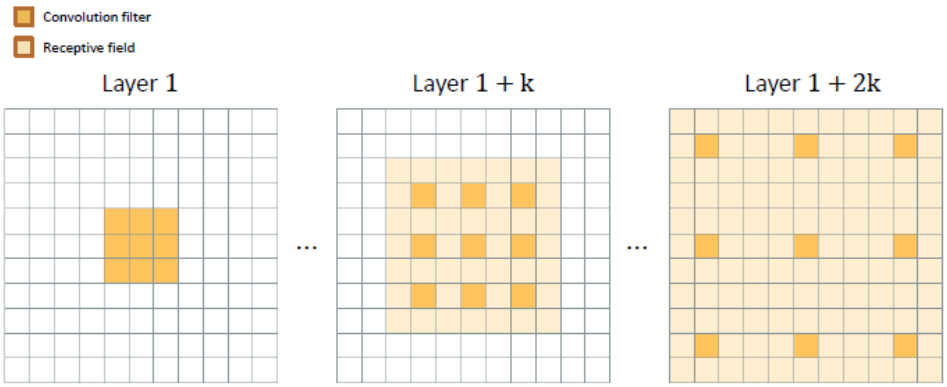

Figure 4.8: Exponentially increasing dilated convolutions for $k = 1$. Image Source: [TL18].

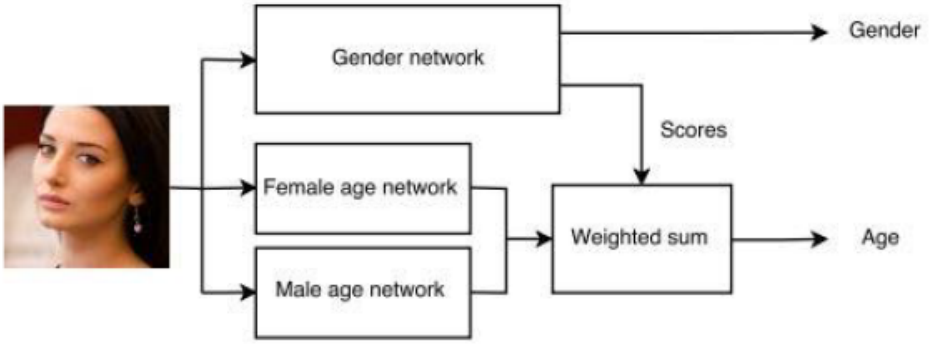

Figure 4.9: Block diagram for the task age and gender estimation. Image Source: [Lee+18].

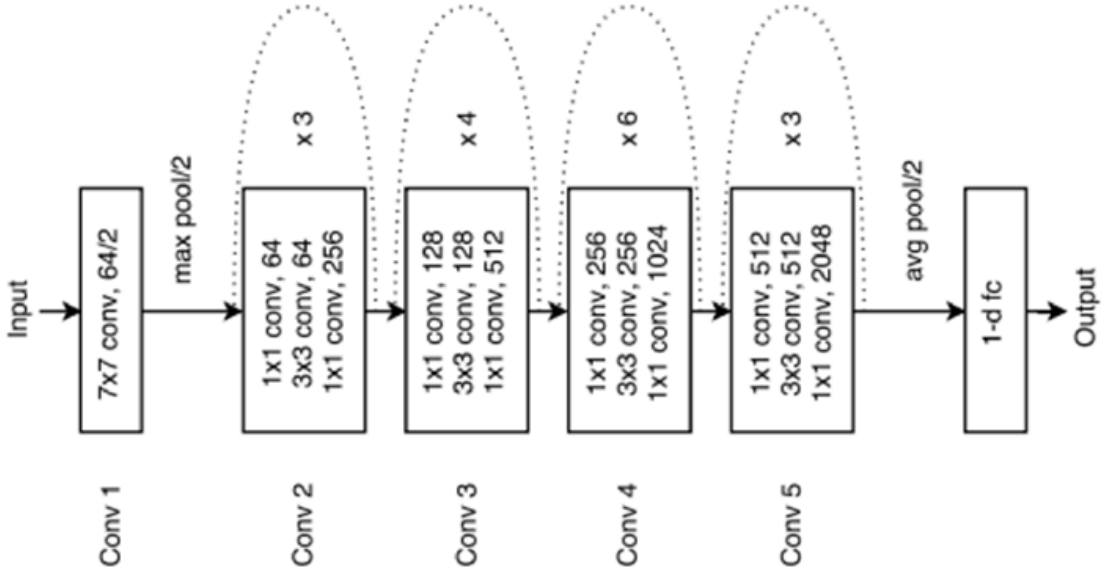

Figure 4.10: Architecture of a 50-layer deep residual network used for all the networks in Figure 4.9. Image Source: [Lee+18].

Three deep residual networks are used to accomplish this task which are Gender, Female age and Male age networks. All the deep residual networks use a model architecture which is almost similar to ResNet-50 [He+16]. The only difference is in the last layer, instead of a 1000-d fully connected, softmax layer (see Figure 5.1, ResNet-50), a 1-d fully connected layer (see Figure 4.10) is used. For more details on ResNet-50, refer Section 4.1 and Subsection 5.1.2. The end output from the age networks (Female age and Male age) is the final age which is weighted sum of the outputs from both the networks and the weights (output scores) are taken from the Gender network (see Figure 4.9). Age is estimated using regression and the mean absolute error is used as a loss function for the model.

5 Experiments

All the experiments and the results in this chapter are taken from [He+16] (refer Section 4.1 for the detailed explanation of the task and the architecture) because this paper contains the most interesting experiments and all the DNN architectures with RL in Chapter 4 are inspired from/motivated from/related to/similar to [He+16]. The plain network can be constructed from Figure 4.1 by removing the dark/dotted curved arrows and the residual network considered is same as Figure 4.1. The experiments are classified based on the datasets used and are described in the following sections.

5.1 ImageNet Classification

This section contains the experiments on the ImageNet 2012 classification dataset [Rus+15] with 1000 classes. The exact implementation should be referred from [He+16] (Subsection 3.4 Implementation). All models are trained on 1.28M training images and the evaluation is done on 50k validation images. The test server gives a final result by evaluating the trained residual network on 100k test images. The error measures used for this dataset are top-1 and top-5 error rates.

5.1.1 Degradation Problem

The first experiment is conducted on the 18-layer and 34-layer plain networks. The detailed architectures for all the variants of the residual networks can be found in Figure 5.1. As seen from Figure 5.3, the validation error (top-1 error) is higher in case of 34-layer plain network as compared to 18-layer plain network. As per the analysis in [He+16], this problem is known as degradation problem and the chances of the problem being caused by vanishing gradients is improbable. The plain networks use Batch Normalization, and it is confirmed by [He+16] that the forward and backward signals do not vanish. Deep plain networks have exponentially low convergence rates and that is the reason why deeper plain networks have higher training error and in turn higher validation error [He+16].

The same experiment as above is conducted on the 18-layer and 34-layer Residual Networks (ResNets). For ResNets, identity mapping is used for all the shortcut connections and option (A) (refer Section 4.1) is used to increase dimensions. As seen from the results in figure the validation error (top-1 error) is higher in case of 18-layer ResNet as compared to 34-layer ResNet which means that DRL has solved the degradation problem.

5.1.2 Deeper Networks

The experiments that are described in this section are based on deeper architectures ResNet-50/101/152. Because of the constraints on the training time, each RB on the left (see Figure 5.2) is replaced by the RB on the right and this type of design is called as the bottleneck

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 5.1: Detailed architectures for variants of the residual networks (also refer Figure 4.1, 5.2). Image Source: [He+16].

design. Consider the bottleneck RB on the right (see Figure 5.2) for a 256 dimensional input. Firstly, the 1×1, 64 layer scales the input down to 64 dimensions. Secondly, the 3×3, 64 layer (bottleneck) is the same as the one of the layers on the left. Finally, the 64 dimensional output from the previous layer is scaled back up to 256 dimensional which is the output of the bottleneck RB. According to [He+16], the identity shortcuts without any parameters make the model more efficient if bottleneck RB are used because if identity shortcuts are replaced with linear projections (refer equation 2.2) for bottleneck RB, the time complexity and the size of the model (parameters) are doubled. Three options are experimented with for shortcut connections: options (A), (B) are the same as in Section 4.1 and for both keep the rest of the shortcut connections as identity mappings, (C) Perform linear projection for all shortcuts connections.

For constructing ResNet-50 using ResNet-34, every 2-layer RB is replaced with this new 3-layer bottleneck RB and the resulting residual network is ResNet-50 (refer figures 4.1, 5.2). For ResNet-50, option (B) is used to increase dimensions. Similarly, by adding more 3-layer bottleneck RB, ResNet-101 and ResNet-152 are constructed. The results of all the described single-model ResNets and different baselines can be seen in Table 5.1.

5.1.3 Comparison with other state-of-the-art ensembles

Table 5.2 compares the ensemble of ResNet and other state-of-the-art ensembles and the ensemble of ResNet performs the best with the top-5 error rate of **3.57%** on the ImageNet test set and this result secured the 1st place in ILSVRC'15 [He+16]. The ensemble of ResNet consisted of six residual networks of varying depths (the depth for two of the residual networks was 152-layer).

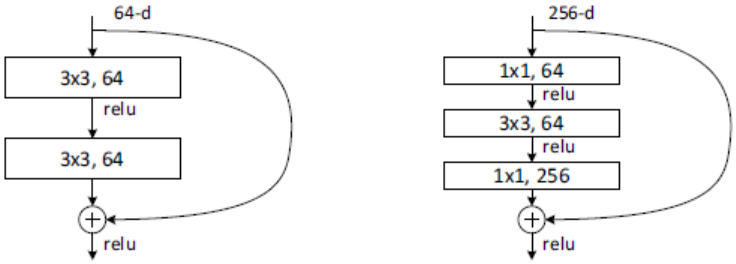

Figure 5.2: Left: A RB which is used in Figure 4.1 after the first 7×7 conv, 64 layer. Right: A bottleneck residual block used for deeper architectures ResNet-50/101/152. Image Source: [He+16].

	plain	ResNet
18 layers	27.94	27.88
34 layers	28.54	**25.03**

Figure 5.3: Results (top-1 error) of validation on ImageNet dataset for plain and ResNets. ResNets and their equivalent plain networks have the same number of parameters (Option A, refer Section 4.1). Image Source: [He+16].

Table 5.1: Single-model results (% error) of ResNets with other baselines on the ImageNet validation set. The symbol † means that the reported results are on the ImageNet test set. Table Source: [He+16].

method	top-1 err.	top-5 err.
VGG [SZ15] (ILSVRC'14)	-	$8.43^{\dagger}$
GoogLeNet [Sze+15] (ILSVRC'14)	-	7.89
VGG [SZ15] (v5)	24.4	7.1
PReLU-net [He+15]	21.59	5.71
BN-inception [IS15]	21.99	5.81
ResNet-34 (B)	21.84	5.71
ResNet-34 (C)	21.53	5.60
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	**19.38**	**4.49**

Table 5.2: Results (% error) of **ensembles**. The top-5 error is communicated by the test server after evaluating the trained model on the ImageNet test set. Table Source: [He+16].

method	top-5 err. (**test**)
VGG [SZ15] (ILSVRC'14)	7.32
GoogLeNet [Sze+15] (ILSVRC'14)	6.66
VGG [SZ15] (v5)	6.8
PReLU-net [He+15]	4.94
BN-inception [IS15]	4.82
ResNet (ILSVRC'15)	**3.57**

5.2 CIFAR-10 Classification

This section contains the experiments on the CIFAR-10 dataset [Kri09]. This dataset consists of 50k training images and 10k testing images with 10 classes. The evaluation is done on the test set. The residual network architecture is similar to Figure 4.1. The inputs (after performing the per-pixel mean subtraction for each image) to the network are 32×32 images. The starting layer of the residual network is the 3×3 convolution layer. After that, $2n$ layers each with the feature map size of $\{32, 16, 8\}$ are stacked i.e. total $6n$ layers with the number of filters $\{16, 32, 64\}$ respectively. The convolutions have a stride of 2. At the end, the residual network has global average pooling, a 10-way fully-connected layer and softmax. In total the residual network has $6n + 2$ weighted layers. The residual shortcuts are used for pairs of 3×3 convolution layers i.e. $3n$ residual shortcut connections (option (A) is used for shortcut connections, refer Subsection 5.1.2). The exact implementation should be referred from [He+16] (Subsection 4.2 CIFAR-10 and Analysis).

5.2.1 Degradation Problem

Different values of $n = \{3, 5, 7, 9\}$ were experimented with, which lead to networks of depth 20, 32, 44 and 56 respectively. For plain networks (see Figure 5.4 Left), degradation problem was observed which is similar to Subsection 5.1.1 which suggests that for deeper plain networks optimizing the network parameters is a problem [He+16]. For ResNets (see Figure 5.4 Middle), the degradation problem is solved and the observed results are quite similar to Subsection 5.1.1.

5.2.2 Deeper Networks more than 1000 layers

A deeper network is constructed by substituting $n = 200$ which results in ResNet-1202. This 1202 layer network has a training error $< 0.1\%$ (see Figure 5.4 Right) and the test error is 7.93% [He+16]. The test error of ResNet-110 is lower than ResNet-1202 and they both have comparable training error. As per the analysis in [He+16], this is due to overfitting and 19.4M parameters are too many parameters for a small dataset CIFAR-10 [Kri09]. This experiment did not use any maxout/dropout, and they incorporate regularization by design i.e. by using deep and thin architectures.

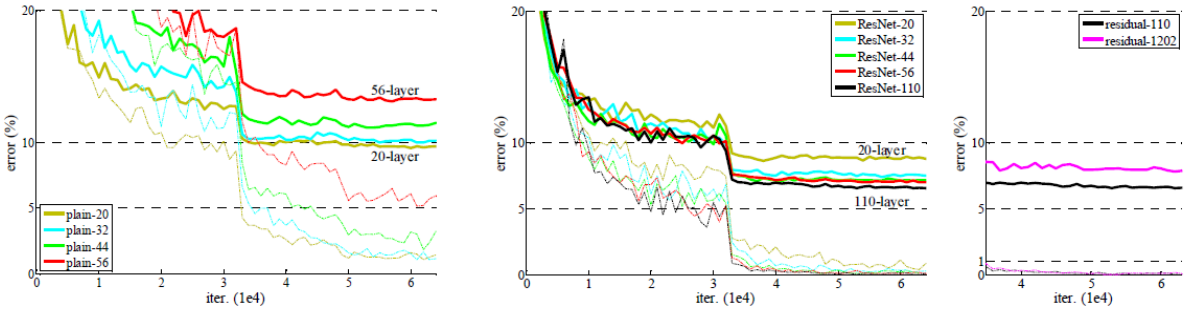

Figure 5.4: Results (error %) on CIFAR-10 dataset. In all the figures training error is denoted by dashed lines and testing error is denoted by bold lines. Left: Plain Networks (the error is higher for the deeper plain networks). Middle: ResNets (the error is lower for the deeper ResNets). Right: ResNet-110 and ResNet-1202. Image Source: [He+16].

6 Summary & Conclusion

6.0.1 Summary

This report introduced a recent technique called DRL which is used to train DNN by introducing shortcut connections within the network architecture. DRL solves the degradation problem and DNN trained using DRL perform better (in terms of accuracy) as compared to the DNN trained without using DRL [He+16].

Most of the applications (tasks) for which DRL is used are from the domains image recognition [He+16; WZL18; Lee+18] and speech recognition [HDHU16; TL18]. This report surveys recent literature related to DRL and details five tasks which use DRL and these are as follows: image recognition [He+16], speech recognition [HDHU16], image steganalysis [WZL18], small footprint keyword spotting [TL18], age and gender estimation [Lee+18].

This report presented some interesting experiments from [He+16] on two datasets ImageNet 2012 classification [Rus+15] and CIFAR-10 [Kri09] and almost all the variants of DNN trained with DRL outperform the other baseline models on both the datasets. For all the other described tasks, almost all the variants of DNN trained with DRL outperform other baseline models used for the respective task.

6.0.2 Conclusion

In general, DNN using DRL are easier to optimize as compared to the normal DNN (plain networks) [He+16]. DNN using DRL can exploit the depth of the DNN which results into more accurate models but this does not mean that the deeper the network the better the accuracy. DNN with higher depths have more parameters which makes the network more prone to overfitting (especially on smaller datasets). For instance, ResNet-1202 (7.93%) performs worse (in terms of testing error) than ResNet-110 (6.43%) on the CIFAR-10 dataset [Kri09] and both have almost the same training error [He+16].

List of Figures

List of Tables

Acronyms

CNN Convolutional Neural Network(s).

DNN Deep Neural Network(s).

DRL Deep Residual Learning.

DRN Deep Residual learning based Network(s).

ELU Exponential Linear Unit.

HPF High-Pass Filtering.

MFCC Mel-Frequency Cepstrum Coefficient.

NN Neural Network(s).

RB Residual Block(s).

ResNet Residual Network.

ResNets Residual Networks.

RL Residual Learning.

SNR signal-to-noise ratio.

WRBN Wide Residual BLSTM Network(s).

Bibliography

[HDHU16] J. Heymann, L. Drude, and R. Haeb-Umbach. "Wide Residual BLSTM Network with Discriminative Speaker Adaptation for Robust Speech Recognition". In: *Computer Speech and Language*. 2016.

[He+15] K. He et al. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *2015 IEEE International Conference on Computer Vision (ICCV)* (2015), pp. 1026–1034.

[He+16] K. He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778.

[IS15] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*. ICML'15. Lille, France: JMLR.org, 2015, 448–456.

[Kri09] A. Krizhevsky. *Learning multiple layers of features from tiny images*. Tech. rep. 2009.

[Lee+18] S. Lee et al. "Age and gender estimation using deep residual learning network". In: *2018 International Workshop on Advanced Image Technology (IWAIT)* (2018), pp. 1–3.

[LH15] G. Levi and T. Hassner. "Age and gender classification using convolutional neural networks". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2015), pp. 34–42.

[Rus+15] O. Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *Int. J. Comput. Vision* 115.3 (Dec. 2015), 211–252. ISSN: 0920-5691. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y.

[SZ15] K. Simonyan and A. Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

[Sze+15] C. Szegedy et al. "Going Deeper with Convolutions". In: *Computer Vision and Pattern Recognition (CVPR)*. 2015. URL: http://arxiv.org/abs/1409.4842.

[TL18] R. Tang and J. Lin. "Deep Residual Learning for Small-Footprint Keyword Spotting". In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2018), pp. 5484–5488.

[Wik21] Wikipedia contributors. *Artificial neural network — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1010367251. [Online; accessed 8-March-2021]. 2021.

[WZL18] S. Wu, S. Zhong, and Y. Liu. “Deep Residual Learning for Image Steganalysis”. In: *Multimedia Tools Appl.* 77.9 (May 2018), 10437–10453. ISSN: 1380-7501. DOI: 10.1007/s11042-017-4440-4. URL: https://doi.org/10.1007/s11042-017-4440-4.