Learning Connections in Multi-branch Residual Networks

Karim Ahmed  
Department of Computer Science  
Dartmouth College

–Project Proposal–

1 Problem

Deep residual learning [3] have led to a series of successful results for image classification. By adopting a residual learning framework, it has become possible to train very deep convolutional neural networks. The degradation problem caused by the increased depth of the network is no longer a serious obstacle for training very deep models [4].

In addition to the importance of the depth and the width of deep convolutional networks, there have been some attempts to study the effect of another dimension called the "Cardinality" or in other words using multiple parallel branches at the same depth instead of a single layer. Recently, Xie et al. [7] showed through an extensive set of experiments, that using multi-branch residual blocks leads to better performance than using a single branch residual block. Figure 1 shows this multi-branch residual block architecture. This multi-branch block allows the learning of different transformations and features represented by the branches.
By adopting a branched architecture as shown in Figure 1, the large number of **static branch connections** that represent different features within each block reveals another dimension, which is the "Connectivity" of the block(s). In other words, finding out the best combination of connections (features) within or between block(s). **The importance of the "Connectivity" in the multi-branch network architectures, comes from different observations. First**, each residual block may need different set of features. We argue that allowing each block to select different set of features at different point during the training stage, will lead to a better performance by adding a regularization effect. This is similar to the effect of the dropout \[6\] technique that is used for reducing overfitting in neural networks by preventing complex co-adaptations on training data. **Second**, at a certain depth the residual block may need different number of branches than at another lower or higher depth. Varying the number of branches within each block, can add flexibility and adaptive capacity to the network. **Third**, each branch within each block can learn better features in case of using its own distinct residual connection. Allowing each branch within each block to have its own residual connection, will help in learning more discriminative features for the branches. **Finally**, if we know the suitable "Connectivity" of each residual block, we could then prune the network architecture by dropping the unnecessary connections and branches to reduce the complexity of the network, and using the new pruned architecture to efficiently train other datasets.

Motivated by these observations, I propose new method to learn the connections in multi-branch residual networks. The new proposed method can be considered a special case of learning the architecture of deep neural networks.

**2 Proposed Method**

![Proposed residual block](image)

In this project, I propose to learn the branch connections in the multi-branch residual networks, by utilizing a new parametric gating component that acts as binary mask. This gate allows only the best suitable input branches to pass through the next block. For this purpose, I propose two different residual block architectures that use this gating mechanism.

The first proposed residual block architecture is "**Type A**", shown in Figure 2. This block has a larger number of blocks \((M)\), than the number of blocks \(C\) in the baseline architecture (Figure 1). The branches in this proposed block are followed by a gate component that is responsible to learn which branches (set of different features) are the most suitable for the next block. Two variants will be adopted for "Type A" model. The first is training the gates in all blocks to learn the best \(M\) branches
out of the total $C$ branches; whereas the second approach is to train each block to select different number of branches.

The second proposed residual architecture is "Type B", shown in Figure 3. This architecture is more flexible and dynamic than "Type A". It allows each branch within each block to have its own residual input, which encourages the features of each branch to be more discriminative than the other branches. Additionally, each branch has its own gating component, which provides more flexibility to connect each branch in one block to a different branch in the preceding or the following block.

The proposed gate component is a learned binary vector $g^b = [g_1^b, g_2^b, \ldots, g_M^b]^T \in \{0, 1\}^M$ specifying the input branches taken into consideration. If $g_m^b = 1$, then the activation tensor produced by the $m$-th branch, where $1 < m < M$, is fed as input to the next layer following the gate. If $g_m^b = 0$, then the input from the $m$-th branch is ignored. Assuming that $E_m$ is the input activation tensor from the $m$-th branch to the gate, the output tensor of the gate $F$ is calculated by the following equation:

$$ F = \sum_{m=1}^{M} g_m^b \cdot E_m. $$

For training the gate component, I will adopt a procedure inspired by the algorithm proposed in [1] to train neural networks with binary weights. During training, a real-valued weights $g^r \in [0, 1]^M$ are maintained in addition to the binary weights. In brief, the training of the gate component consists of three steps: 1) forward propagation, 2) backward propagation, and 3) parameters update. In the forward step, the real-valued weights will be stochastically binarized (by drawing samples from a multinomial distribution) during the forward propagation and backward propagation (steps 1 and 2), whereas during the parameters update (step 3), the method updates the real-valued weights. The proposed training procedure assumes that there are $C$ active input branches in the gate, where $C$ is a predefined integer hyper-parameter with $1 \leq C \leq M$. This training procedure will be applied to the
first variant of model "Type A", and to the model "Type B". For the second variant of "Type A", I will use a predefined threshold on the values of real-valued weights instead of using $C$ the number of active input branches.

3 Data sets

3.1 CIFAR-100

CIFAR-100 [5] is a dataset of 32x32 color images, and 100 classes. The training set contains 50,000 examples and the test set includes 10,000 images. This is dataset is suitable to conduct a comprehensive study using different network architectures and settings.

3.2 ImageNet-1K

ImageNet [2] dataset includes images of 1000 object categories. The training set consists of 1.28M photos, while the validation set contains 50K images. I will train on the training set and use the validation set to assess performance.

4 Goals by the milestone due date:

- Implementation of the proposed gate component, and the training algorithm.
- Implementation of the proposed models (Type A and Type B).
- Conducting preliminary experiments on CIFAR-100 dataset using the proposed models (Type A and Type B), to determine which is the best model.

References


