Big-Little Net: An Efficient Multi-Scale Feature Representation for Visual and Speech Recognition

Chun-Fu (Richard) Chen, Qianfu Fan, Neil Mallinar, Tom Sercu, Rogerio Feris
IBM T.J. Watson Research Center
{chenrich, qfan, neil.r.mallinar, tom.sercu1, rsferis}@us.ibm.com

Abstract

In this paper, we propose a novel Convolutional Neural Network (CNN) architecture for learning multi-scale feature representations with good tradeoffs between speed and accuracy. This is achieved by using a multi-branch network, which has different computational complexity at different branches. Through frequent merging of features from branches at distinct scales, our model obtains multi-scale features while using less computation. The proposed approach demonstrates improvement of model efficiency and performance on both object recognition and speech recognition tasks, using popular architectures including ResNet and ResNeXt. For object recognition, our approach reduces computation by 33% on object recognition while improving accuracy with 0.9%. Furthermore, our model surpasses state-of-the-art CNN acceleration approaches by a large margin in accuracy and FLOPs reduction. On the task of speech recognition, our proposed multi-scale CNNs save 30% FLOPs with slightly better word error rates, showing good generalization across domains.

1 Introduction

Deep Convolutional Neural Network (CNN) models have achieved substantial performance gains in many computer vision and speech recognition tasks [1][2][3][4]. However, the accuracy obtained by these models usually grows proportionally with their complexity and computational cost. This poses an issue for deploying these models in applications that require real-time inferencing and low-memory footprint, such as self-driving vehicles, human-machine interaction on mobile devices, and robotics.

Motivated by these applications, many methods have been proposed for model compression and acceleration, including techniques such as pruning [5][6][7], quantization [8][9], and low-rank factorization [10][11][12]. Most of these methods have been applied to single-scale inputs, without considering multi-resolution processing. More recently, another line of work relies on dynamic routing, which is a content-adaptive approach, meaning networks use different workloads based on the complexity of the images [13][14][15]. On the other hand, multi-scale feature representations have proven successful for many vision and speech recognition tasks compared to single-scale methods [16][17][18][19]; however, the computational complexity is usually ignored for multi-scale networks.

The computational cost of a CNN model has much to do with the input size. A model, if running at half of the image size, can gain a remarkable computational saving of 75%. Based on this fact, we propose an efficient network architecture by combining multi-scale image and speech information through a multi-branch network. As shown in Fig. 1, our key idea is to use a high-complexity branch (accurate but costly) for low-scale feature representation and low-complexity branch (efficient but less accurate) for high-scale feature representation. The two types of features are frequently merged together to complement and enrich each other, leading to a stronger feature representation than either

Preprint. Work in progress.
Figure 1: Our proposed Big-Little Net (bL-Net) for efficient multi-scale feature representations. (a) The bL-Net stacks several Big-Little Modules. A bL-module include $K$ branches ($K = 2$ in this illustration) where the $k^{th}$ branch represents an image scale of $1/2^k$. ‘M’ here denotes a merging operation. (b) Our implementation of the Big-Little Module includes two branches. The Big-Branch has the same structure as the baseline model while the Little-Branch reduces the convolutional layers and feature maps by $\alpha$ and $\beta$, respectively. Larger values of $\alpha$ and $\beta$ lead to lower computational complexity in Big-Little Net.

The main contributions of our paper are summarized as follows:

- We propose an efficient and effective multi-scale CNN for object and speech recognition.
- We demonstrate that our approach can reduce computation by 33% in models such as ResNet and ResNext on object recognition while improving accuracy with 0.9%. This result outperforms state-of-the-art networks that focus on CNN acceleration by a large margin of accuracy at the same FLOPs.
- We validated the proposed method on a speech recognition task, where we also achieve better word error rates while reducing the number of FLOPs by 30%.

2 Related Work

Model Compression and Acceleration. Network pruning [5, 6] and quantization [8, 9] are popular techniques to remove model redundancy and save computational cost. Another thread of work consists of training a sparse model directly, such as IGCv2 [20] and SCConv [21]. Efficient network architectures like SqueezeNet [22] and MobileNet [23] have also been explored for training compact deep networks. Other methods include knowledge distillation [24], compression with structured matrices [25, 26], and hashing [27]. Dynamic routing [13, 14, 15] has also been explored in residual networks to improve efficiency. These methods operate with single-resolution inputs, while our approach processes multi-resolution data. It could be used in tandem with these methods to further improve efficiency.

Multi-Resolution Feature Representations. The notion of multi-scale feature representation can be dated back to image pyramids [28] and scale-space theory [29]. More recently, several methods have proposed multi-scale CNN-based architectures for object detection and recognition. MSCNN [10], DAG-CNN [31] and FPN [32] use features at different layers to form multi-scale features. Hourglass networks [33] use a hierarchical multi-resolution model for human pose estimation. However, this approach induces a heavy workload as the complexity of each sub-network in their model is equal. Nah et al. [16] and Eigen et al. [34] combine the features from multiple networks working on different resolutions to generate multi-scale features. The overall computational cost grows along with the number of scales, leading to inefficient models. In contrast to existing meth-
We develop a simple, easy-to-implement, yet very efficient and effective network architecture. It
learns multi-scale feature representations by fusing multiple branches with different image scales
and computational complexity. As shown in Fig. [1] we design a multi-scale feature module with the
following principles: (I) each branch corresponds to a single unique image scale (or resolution); (II) the
cost of a branch is inversely proportional to the scale. Note that the principle (II) implies that we use
high-complexity networks at lower resolutions and low-complexity networks at higher resolutions for
the sake of efficiency.

3 Our Approach

We develop a simple, easy-to-implement, yet very efficient and effective network architecture. It
learns multi-scale feature representations by fusing multiple branches with different image scales
and computational complexity. As shown in Fig. [1] we design a multi-scale feature module with the
following principles: (I) each branch corresponds to a single unique image scale (or resolution); (II) the
cost of a branch is inversely proportional to the scale. Note that the principle (II) implies that we use
high-complexity networks at lower resolutions and low-complexity networks at higher resolutions for
the sake of efficiency.

3.1 Big-Little Net

\textit{bL-Net} is a sequence of Big-Little Modules, each one taking input \( x_i \) and producing output \( x_{i+1} \).
Within a bL-Module, assume we have \( K \) branches working on \( K \) scales \([1, 1/2, 1/4, \ldots, 1/2^{K-1}]\).
We denote a feature map \( x_k \) at scale \( 1/2^k \) as \( x_k \), indicating the spatial size of \( x_k \) downsampled by \( 2^k \)
with respect to the original input dimension. We use a weighted sum to combine all branches into a
feature representation at scale \( 1/2^k \). Mathematically, the module’s output \( x_{i+1} \) can be expressed by

\[
x_{i+1} = F \left( \sum_{k=0}^{K-1} c^k S^k \left( f_k \left( x_k^i \right) \right) \right),
\]

where \( f_k(\cdot) \) denotes a sequence of convolutional layers. Typically for higher \( k \), \( f_k \) will have more
convolutional layers having more feature maps. \( S^k(\cdot) \) is the operation that matches the output size of
the branches, either: (1) increasing the number feature maps with a \( 1 \times 1 \) convolution, (2) upsampling
to match the output size of the \( k = 0 \) branch, or both. \( c^k \) indicates the weighting coefficients of each
scale in the merge while \( F(\cdot) \) is an optional final fusion layer like a convolutional layer.

Note that merging the branch outputs happens at the highest resolution and highest number of feature
maps between the branches. Maintaining these large intermediate states avoids information loss. A
crucial aspect of this design is that through consecutive merging and downsampling, the expensive
branches operating at low resolution still have access to the high resolution information, processed
by the cheaper branches in the previous module.

While our design is suitable for any number of networks, in this work we primarily focus on the
case of two networks, i.e., \( K = 2 \). We also experimented with \( K > 2 \) in object and speech
recognition; however, the \( K = 2 \) case provided the best balance between accuracy and computation
(See Appendix A.4 and Section 4.2 for details). Following the principles above, we propose a multi-
network architecture that integrates two branches for multi-scale feature representation. Figure 1(b)
shows an example Big-Little Net architecture.

The module includes two branches, each of which represents a separate network block from a
deep model \( \text{(accurate but costly)} \) and a less deep counterpart \( \text{(efficient but less accurate)} \). The two
branches are fused at the end through linear combination with unit weights (i.e., \( c^0 = c^1 = 1.0 \)).
Before fusion, the low resolution feature map is upsampled using bilinear interpolation to align the
resolution of the feature maps \( (=S^1(\cdot)) \). Similarly, the high resolution feature map has an additional
\( 1 \times 1 \) convolutional layer to increase the number of output channels \( (=S^0(\cdot)) \). Furthermore, since our
design is based on ResNet, we add a residual block to further fuse the combined features (i.e., \( F(\cdot) \)
is a residual block). For convenience, we refer to these two branches as \( \text{Big-Branch} \) (many layers
and channels, low resolution) and \( \text{Little-Branch} \) (fewer layers and channels, high resolution), respec-
tively. We also denote the module as Big-Little Module and the entire architecture as Big-Little Net or bL-Net.

To control the complexity of bL-Net, we introduce two parameters to specify the complexity of the Little-Branch with respect to the Big-Branch. The Big-Branch will typically maintain the structure of the original network, though operating at lower resolution. The Little-Branch (operating at full resolution) will be slimmed and shortened, as controlled by the parameters $\alpha$ and $\beta$. As shown in Fig. 1(b), $\alpha$ controls the reduction in number of channels in the convolutional layers of Little-Branch, to slim the network, and $\beta$ controls the Little-Branch reduction of number of layers, to shorten the depth of the network. As demonstrated later, to control the structure and computational complexity of bL-Net, it is critical to choose $\alpha$ and $\beta$ appropriately (See Table 1 and Table 4). Larger values of $\alpha$ and $\beta$ lead to lower complexity in bL-Net.

### 3.2 Network Merging

We consider two options for merging the outputs of the branches. Merging CNNs at different scales is straightforward, and there are two types of merges in the literature. The first one is a linear combination, which joins features from two networks by addition. The alternative concatenates the outputs of the two networks along the channel dimension and, if needed, subsequently applies a $1 \times 1$ convolution to reduce the number of feature maps. Both merging approaches have their pros and cons. With linear combination, the branches can easily compensate each other, meaning each branch can activate output features not activated by the other branch; however, both feature map resolution (of Big-Branch) and number of channels (of Little-Branch) have to be increased before addition. On the other hand, concatenation only needs to align the feature map size, however requires a $1 \times 1$ convolution reducing the number of channels after concatenation, which is a more expensive operation than the pointwise addition.

While linear combination provides an immediate exchange of the activations of both branches, concatenation relies on the following layers for this exchange. This delay in exchange could possibly be problematic if the information from each branch is destructively altered before merging. For example, a nonlinearity such as ReLU would discard all activations less than zero, effectively ignoring negative features in both branches before merging. Since linear combination does not cause too much overhead and provides better accuracy, we chose linear combination as our merging approach. In Appendix A.4, we empirically show that the linear combination approach performs better than concatenation in object recognition.

### 4 Experimental Results

We conducted extensive experiments, discussed below, to validate the effectiveness of our proposed bL-Net on object and speech recognition tasks. bL-Net can be integrated with many modern CNNs and here we chose ResNet [1] as the primary architecture to evaluate our approach. For simplicity, from now on, we use bL-M to represent the bL-Net with the backbone network from model M. For example, bL-ResNet-50 is the Big-Little net using ResNet-50 as the backbone network.

#### 4.1 Object Recognition

We used the ImageNet dataset [37] for all experiments on object recognition. The ImageNet dataset contains 1,000 classes, 1.28 million training images, and 50k validation images, and it is a common benchmark for object recognition. All details of our experimental setup and network structures for bL-ResNet-50, 101 and 152 can be found in Appendix A.1.

**ResNet as the backbone network.** We experimented with different complexity control factors ($\alpha$ and $\beta$) to better understand their effects on performance. $\alpha$ and $\beta$ control both the structural and computational complexity of the Little-Branch, which determines the overall complexity of bL-Net.

As can be seen in Table 1, all the models based on ResNet-50 yield better performance over the baseline with less computation, clearly demonstrating the advantage of combining low- and high-complexity networks to balance between speed and accuracy. In addition, the small performance gaps between these models suggest that a computationally light Little-Branch ($< 15\%$ of the entire network) can compensate well for the low resolution representation by providing finer high-
resroduction information. We consider $\alpha = 2$ and $\beta = 4$ as the default setting for all of the following experiments.

We further evaluated our approach on deeper models by using ResNet-101 and ResNet-152 as the backbone networks. We see from Table 2 that bL-ResNet-101 and bL-ResNet-152 behave similarly to bL-ResNet-50. As expected, both of them produce better results against the baseline models and achieving notable computational gains. Interestingly, our approach computationally favors deeper models. This is evidenced by the fact that more speedups are observed on bL-ResNet-152 (2.3×) than on bL-ResNet-101 (2.0×) and bL-ResNet-50 (1.4×). This is mainly because deeper models require less computation when operating at low resolution.

ResNet-101 and ResNet-152 present a big structural difference from ResNet-50 in that they have imbalanced residual block distribution across the network. They have most of the residual blocks at the end of the network where the feature maps are significantly downsampled. While such a design may be suitable for a vanilla deep model, it likely limits the ability of the Big-Branch to learn information, since the Big-Branch reduces the resolution by another 2×. To address this issue, we move some blocks to the earlier stages for balance. These changes do not affect FLOPs but reduces the parameters by about 5%. The details of this can be found in Appendix A.2.

We further tested bL-ResNet-101 and bL-ResNet-152 with a deeper Little-Branch ($\alpha = 2$, $\beta = 4$). As can be seen in Table 2 the model outperform their baselines by 0.55%, yet still achieving a considerable reduction of FLOPs by nearly 2×.

**ResNeXt as the backbone network.** We extended bL-Net to ResNeXt, one of the more accurate yet compact network architectures. We experimented with ResNeXt-50 and ResNeXt-101 using the 32×4d setting [38]. In our case, the Big-Branch follows the same setting of ResNeXt; however, we changed the setting of the Little-Branch to 16×8d so that each group convolution of the Little-Branch keeps the same number of input channels to the Big-Branch in the case of $\alpha = 2$.

Table 2 shows the results of ResNeXt and our bL-ResNeXt. bL-ResNeXt-50 achieves a moderate speedup (1.40×) and provides an additional gain of 0.4% in accuracy. However, bL-ResNeXt-101

| Model                  | Top-1 Error | FLOPs (10^9) | Params (10^6) |
|-----------------------|-------------|--------------|---------------|
| ResNet-50             | 23.66%      | 4.09         | 25.55         |
| bL-ResNet-50 ($\alpha = 2$, $\beta = 2$) | 22.72%    | 2.91 (1.41×) | 26.97         |
| bL-ResNet-50 ($\alpha = 2$, $\beta = 4$) | **22.69%** | 2.85 (1.44×) | 26.69         |
| bL-ResNet-50 ($\alpha = 4$, $\beta = 2$) | 23.20%     | 2.49 (1.64×) | 26.31         |
| bL-ResNet-50 ($\alpha = 4$, $\beta = 4$) | 23.15%     | **2.48 (1.65×)** | 26.24         |

Table 1: Complexity study of the Little-Branch ($\alpha$ and $\beta$) for bL-ResNet-50.
gains a much more substantial speedup of 2× and seeing 0.4% improvement in the accuracy. Our approach is expected to perform even better on a deeper network such as ResNeXt-152, based on the experience with ResNet.

We also compared our approach with the baselines under the assumption that a similar number of FLOPs are used. This is achieved by evaluating our approach at a larger image scale, i.e., 256 × 256. As illustrated in Table 2 bl-ResNet and bl-ResNeXt evaluated at 256 × 256 is consistently better than the corresponding baseline while still using fewer FLOPs. The advantage becomes more pronounced with deeper models. For instance, bl-ResNet-152 exceeds the baseline by a large margin of 1.2% while saving 43% FLOPs as compared to ResNet-152. Furthermore, with even fewer FLOPs, bl-ResNeXt-101 boosts the top-1 performance by 0.8%, which is quite impressive given that ResNeXt-101 is a very competitive baseline.

Table 2 also shows the running time on a GPU, the speedup is consistent with the FLOPs reduction, which justifies our design is practical.

4.1.1 Comparison with Related Work

We compared our method with the state-of-the-art works that aim to reduce the FLOPs on ResNet and variants of ResNet, including ResAttNet [39] and WideResNet [40]. The results are shown in Figure 2.

Our bl-Net significantly outperforms all related works regarding FLOPs reduction and accuracy. Comparing to the network pruning approaches (PFEC [6] and LCCL [5]), our bl-ResNet-101 (α = 2, β = 4) is ~ 5% better and uses less computation since bl-Net utilizes a compact way to extract multi-scale features for better performance and saves FLOPs. Comparing to SACT and ACT [15], our bl-ResNet-101 (α = 2, β = 4) improves the accuracy by 5% while using the same number of FLOPs. On the other hand, our bl-ResNet-101 (α = 2, β = 4) outperforms BlockDrop [13] and SkipNet [14] by 3.7% and 2.2% top-1 accuracy at the same FLOPs. Thus, rather than let input data select different paths in run-time to save FLOPs, bl-Net is carefully designed to use efficient multi-scale features to achieve better performance with low FLOPs usage. Regarding SPPPoint [41], under the same FLOPs, our bl-ResNet-101 (α = 2, β = 4) surpass it by 2.2% accuracy; furthermore, to achieve the same performance, it requires 1.8× more FLOPs than our bl-ResNet-101 (α = 2, β = 2).

Our bl-ResNet-101 (α = 2, β = 4) surpasses both WideResNet and ResAttNet by a large margin; at a similar accuracy, our bl-ResNet-101 (α = 2, β = 4) saves 66% FLOPs and 53% FLOPs over both models, respectively, and the same trend could be found for bl-ResNeXt and ResAttNeXt. The advantages of bl-Net comes from the multi-scale feature extraction and effective merge, which leads the big branch to extract important features at a lower resolution while the little branch compensates.

Figure 2: (a) Performance comparison among related works, and bl-Net outperforms all related works. (b) A zoom-in of the bottom-left of (a), the annotations denote the depth of each bl-Net.
Table 3: Different number of merges in bL-ResNet. $m$: number of merges. ($\alpha = 2$, $\beta = 4$)

| Model                  | Top-1 Error | FLOPs (10^9) | Params (10^6) |
|------------------------|-------------|--------------|---------------|
| bL-ResNet-50 ($m = 4$) (baseline) | 22.69%      | 2.85         | 26.69         |
| bL-ResNet-50 ($m = 2$)  | 23.48%      | 2.74         | 26.66         |
| bL-ResNet-50 ($m = 1$)  | 24.57%      | 2.64         | 26.64         |
| bL-ResNet-101 ($m = 4$) (baseline) | 21.80%      | 3.89         | 41.85         |
| bL-ResNet-101 ($m = 7$) | 21.85%      | 5.21         | 44.44         |

with features that can be found at a higher resolution. Furthermore, our work is independent of above works; that is, bL-Net can further improve its efficiency by integrating with them.

4.1.2 Number of Merges in bL-Net

We also analyzed the number of merges we needed in the bL-Net. One big difference between our approach and others is that bL-Net merges multiple times as opposed to only once in most of the other approaches. Below we provide an explanation of why more information exchange is encouraged in our approach and when is the best moment for merging operation.

In the above bL-Net, we merged branches before the feature dimension changes, except for the first stride convolution; thus, we used 4 merges ($m = 4$). We experimented with a different number of merges for bL-ResNet-50 and bL-ResNet-101, and the results are shown in Table 3. Since there are fewer layers in bL-ResNet-50, we reduce the number of merges to show their importance; on the other hand, there are more layers in bL-ResNet-101, so we add more merges to show that those additional merges would similarly not improve the performance anymore.

The accuracy of the bL-ResNet-50 models with less number of merges ($m = 1$ and $m = 2$) is significantly worse than with more ($m = 4$) and they do not save many FLOPs and parameters at all. This justifies frequent information exchange improves the performance. On the other hand, bL-ResNet-101 ($m = 7$) uses more merges; however, it also does not improve the performance and requires more FLOPs, which comes from more merges. This is because the original setting for the amount of merging happened when either the channel number or feature map size is changed, so extra merges happened at the feature dimension. Thus, those extra merges could be redundant since merging at identical dimension could be reduced to one merging. Hence, it empirically proves that merging before dimension is changed is the most effective.

4.2 Speech Recognition

We train ResNet style acoustic models in the hybrid framework on Switchboard+Fisher (2000h) and provide results on Hub5 (Switchboard and Call Home portions). Switchboard is a large dataset with 2000 hours of transcribed speech from 28,000 speakers, which is actively used as benchmark [42, 43] akin to ImageNet in the computer vision community. Our ResNet acoustic models are similar to the state of the art models described in [43], though slightly simplified (less fully connected layers) and trained with a simpler procedure (no class balancing). We provide results only after Cross-Entropy training and after decoding with a small language model (4M n-grams). Gains from this setting are typically maintained in the standard further pipelines like fine-tuning with sequence training, using more complex language models.

Appendix B gives a thorough overview of the architecture of the speech acoustic models. The main difference speech acoustic models have compared to image classification networks, is that striding or pooling only happens along the frequency axis, while along in the time direction we need to output dense predictions per frame [4]. This means that the branches at different resolutions have a fundamentally different view of the signal as it is propagating through the network; the ratio of resolution in frequency (downsampled in the Big-Branch) vs resolution in time (same between branches) is different. We can think about this as the convolutional kernels having different “aspect ratio”’s between branches. Therefore we not only expect FLOP reductions in bL-Net, but expect to have increased representational power. In addition, similar to the case in object recognition (Table 2), we could process the speech signal at higher frequency resolution than what is computationally feasible for the baseline ResNets.
Table 4: Speech recognition results. We present results on Hub5 and the CallHome portion of Hub5, while the RT-02 Switchboard set was used for selecting decode epoch and HMM prior settings.

| Model Description                  | FLOPs ($10^9$) | Params ($10^6$) | WER Avg | Hub5   | Hub5 CH   |
|------------------------------------|----------------|-----------------|---------|--------|-----------|
| 1 Baseline: ResNet-22              | 1.11           | 3.02            | 14.67%  | 11.15% | 18.17%    |
| 2 bL-ResNet-22 ($\alpha = 4, \beta = 1$) | 0.68 (1.63×)  | 3.15            | 14.72%  | 11.24% | 18.18%    |
| 3 bL-ResNet-22 ($\alpha = 4, \beta = 2$) | 0.66 (1.68×)  | 3.11            | 14.47%  | 10.95% | 17.99%    |
| 4 bL-ResNet-22 ($\alpha = 4, \beta = 3$) | 0.65 (1.70×)  | 3.10            | 14.66%  | 11.25% | 18.05%    |
| 5 bL-ResNet-22 ($\alpha = 2, \beta = 3$) | 0.77 (1.43×)  | 3.07            | 14.46%  | 11.10% | 17.80%    |
| 6 bL-ResNet-22 ($\alpha = 4, \beta = 1$) cat | 0.70 (1.58×)  | 3.18            | 14.67%  | 11.31% | 18.00%    |
| 7 bL-PYR-ResNet-22 ($\alpha = 4, \beta = 1$) | 0.98 (1.13×)  | 3.32            | 14.50%  | 11.05% | 17.92%    |

Table 4 shows the results for the different architectures described in Appendix B. Most results are in line with the observations in the object recognition bL-Net. When comparing the baseline ResNet-22 (line 1) to the best bL-ResNet-22 (line 5), we see not only a reduction in FLOPs, but also a modest gain in Word Error Rate (WER). Comparing lines 2-4, we see that increasing $\beta$ (i.e. shorter little branches at full resolution) causes no WER degradation, while reducing the number of FLOPs. From line 5 we see that, similar to the object recognition ResNet results, decreasing $\alpha$ from 4 to 2 (i.e. keeping more feature maps in the full-resolution little branches) is important for performance, even though this increases the FLOPs again. We can summarize the best setting of $\alpha = 2$ and $\beta = 3$ for the little branches at full resolution: make them shorter but with more feature maps. This is consistent with the image classification results. From line 2 vs. line 6, the concatenation merge mode performs similar to the default additive merging, while increasing the number of FLOPs. Line 7 (compare to line 2) shows an experiment with additional branches on the lower layers (See Appendix B). Although there is some gain in WER, the added parameters and compute on the lower layers may not make this a worthwhile trade-off.

4.3 Discussion on bL-Net

From the results of both tasks, we observe the following common insights, which enable us to design an efficient multi-scale network with competitive performance: (I) The Little-Branch can be very light-weight (< 15% of the overall computation), (II) bL-Net performs better when the Little-Branch is wide and shallow (smaller $\alpha$ and larger $\beta$) rather than deep and thin (larger $\alpha$ and smaller $\beta$) under similar complexity, (III) merging is effective when the feature dimension has changed, and (IV) branch merging by addition is more effective than concatenation. (I) is because the Big-Branch can extract essential information, so a light Little-Branch is good enough to provide sufficient information the Big-Branch lacks. Regarding (II), wider networks have been shown to perform better than deep networks while using a similar number of parameters [40]. Finally, (III) is well-discussed in Section 4.1.2. Merging through addition provides better regularization for both branches to learn complementary features to form strong features. The experimental results also show that merging by addition achieves better performance.

5 Conclusion

We proposed an efficient multi-scale feature representation based on integrating multiple networks for object and speech recognition. The Big-Branches gain significant computational reduction by working at low-resolution input but still extract meaningful features while the Little-Branch enriches the features from high-resolution input but with light computation. On object recognition task, we demonstrated that our approach provides approximately 2× speedup over baselines without compromising any accuracy; furthermore, we reduced computation by 33% on object recognition while improving accuracy with 0.7%. This result significantly outperforms the state-of-the-art networks by a large margin in terms of accuracy and FLOPs reduction. Furthermore, when using the proposed method on speech recognition task, we gained 0.2% WER and saved 30% FLOPs at the same time. That evidence showed that the proposed bL-Net is an efficient multi-scale feature representation structure for competitive performance with less computation. In this paper, we chose ResNet as our backbone network but bL-Net can be integrated with other advanced network structures, like DenseNet [44], DualPathNet [45] and NASNet [46] to achieve competitive performance while sav-
ing computations. Furthermore, bL-Net can be integrated with those CNN acceleration approaches to make models more compact and efficient.

Acknowledgments

We would like to thank Paul Crumley and I-Hsin Chung for generously sharing the GPU machines to complete the experiments.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[2] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In The IEEE International Conference on Computer Vision (ICCV), October 2017.

[3] O Vinyals, A Toshev, S Bengio, and D Erhan. Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 39(4):652–663, April 2017.

[4] Tom Sercu and Vaibhava Goel. Dense prediction on sequences with time-dilated convolutions for speech recognition. NIPS End-to-end Learning for Speech and Audio Processing Workshop, 2016.

[5] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[6] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning Filters for Efficient ConvNets. In International Conference on Learning Representation 2017, pages 1–13, March 2017.

[7] Song Han, Huizi Mao, and William J Dally. Deep Compression: Compressing Deep Neural Network with Pruning, Trained Quantization and Huffman Coding. ArXiv, abs/1510.00149, 2015.

[8] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Binarized Neural Networks. In D D Lee, M Sugiyama, U V Luxburg, I Guyon, and R Garnett, editors, Advances in Neural Information Processing Systems 29, pages 4107–4115. Curran Associates, Inc., 2016.

[9] Fengfu Li and Bin Liu. Ternary weight networks. CoRR, abs/1605.04711, 2016.

[10] Wei Wen, Cong Xu, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Coordinating Filters for Faster Deep Neural Networks. In The IEEE International Conference on Computer Vision (ICCV), October 2017.

[11] Yani Ioannou, Duncan P Robertson, Jamie Shotton, Roberto Cipolla, and Antonio Criminisi. Training CNNs with Low-Rank Filters for Efficient Image Classification. ArXiv, abs/1511.06744, 2015.

[12] X Zhang, J Zou, K He, and J Sun. Accelerating Very Deep Convolutional Networks for Classification and Detection. IEEE Transactions on Pattern Analysis and Machine Intelligence, PP(38):1943–1955, October 2016.

[13] Zuxuan Wu, Tushar Nagarajan, Abhishek Kumar, Steven Rennie, Larry S Davis, Kristen Grauman, and Rogerio Feris. Blockdrop: Dynamic inference paths in residual networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[14] Xin Wang, Fisher Yu, Zi-Yi Dou, and Joseph E. Gonzalez. Skipnet: Learning dynamic routing in convolutional networks. In SysML, Feb 2018.

[15] Michael Figurnov, Maxwell D. Collins, Yukun Zhu, Li Zhang, Jonathan Huang, Dmitry Vetrov, and Ruslan Salakhutdinov. Spatially adaptive computation time for residual networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[16] Seungjun Nah, Tae Hyun Kim, and Kyoung Mu Lee. Deep Multi-Scale Convolutional Neural Network for Dynamic Scene Deblurring. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[17] Yanbei Chen, Xiatian Zhu, and Shaogang Gong. Person Re-Identification by Deep Learning Multi-Scale Representations. In The IEEE International Conference on Computer Vision (ICCV), October 2017.

[18] László Tóth. Multi-resolution spectral input for convolutional neural network-based speech recognition. In Speech Technology and Human-Computer Dialogue (SpeD), 2017.

[19] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. Pattern Analysis and Machine Intelligence, 2013.
[20] Guotian Xie, Jingdong Wang, Ting Zhang, Jianhuang Lai, Richang Hong, and Guo-Jun Qi. IGCV2: Interleaved Structured Sparse Convolutional Neural Networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2018.

[21] Quanfu Fan, Chun-Fu (Richard) Chen, and Gwo Giun (Chris) Lee. Sparse Deep Feature Representation for Object Detection from Wearable Cameras. In Proceedings of the British Machine Vision Conference (BMVC). BMVA Press, September 2017.

[22] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. arXiv, pages 1–11, September 2017.

[23] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. ArXiv, abs/1704.04861, 2017.

[24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv:1503.02531, 2015.

[25] Y Cheng, F X Yu, R S Feris, S Kumar, A Choudhary, and S F Chang. An Exploration of Parameter Redundency in Deep Networks with Circulant Projections. In 2015 IEEE International Conference on Computer Vision (ICCV), pages 2857–2865. IEEE, December 2015.

[26] Vikas Sindhwani, Tara Sainath, and Sanjiv Kumar. Structured Transforms for Small-Footprint Deep Learning. In C Cortes, N d Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, Advances in Neural Information Processing Systems 28, pages 3088–3096. Curran Associates, Inc., 2015.

[27] Wenlin Chen, James T. Wilson, Stephen Tyree, Kilian Q. Weinberger, and Yixin Chen. Compressing neural networks with the hashing trick. In Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37, ICML’15, pages 2285–2294. JMLR.org, 2015.

[28] Edward H Adelson, Charles H Anderson, James R Bergen, Peter J Burt, and Joan M Ogden. Pyramid methods in image processing. RCA engineer, 29(6):33–41, 1984.

[29] Tony Lindeberg and Bart M ter Haar Romeny. Linear scale-space i: Basic theory. In Geometry-Driven Diffusion in Computer Vision, pages 1–38. Springer, 1994.

[30] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A Unified Multi-scale Deep Convolutional Neural Network for Fast Object Detection. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, European Conference on Computer Vision (ECCV), pages 354–370, 2016.

[31] Songfan Yang and Deva Ramanan. Multi-Scale Recognition With DAG-CNNs. In The IEEE International Conference on Computer Vision (ICCV), December 2015.

[32] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[33] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked Hourglass Networks for Human Pose Estimation. In Computer Vision – ECCV 2016, pages 483–499, Cham, 2016. Springer International Publishing.

[34] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. In The IEEE International Conference on Computer Vision (ICCV), December 2015.

[35] Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Weinberger. Multi-scale dense networks for resource efficient image classification. In International Conference on Learning Representations, 2018.

[36] Shreyas Saxena and Jakob Verbeek. Convolutional neural fabrics. In Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS’16, pages 4060–4068, USA, 2016. Curran Associates Inc.

[37] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV), 115(3):211–252, 2015.

[38] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated Residual Transformations for Deep Neural Networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[39] Fei Wang, Mengqing Jiang, Chen Qian, Shuo Yang, Cheng Li, Honggang Zhang, Xiaogang Wang, and Xiaou Tang. Residual Attention Network for Image Classification. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[40] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In Edwin R Hancock Richard C Wilson and William A P Smith, editors, Proceedings of the British Machine Vision Conference (BMVC), pages 87.1–87.12. BMVA Press, September 2016.
[41] Jason Kuen, Xiangfei Kong, Zhe Lin, Gang Wang, Jianxiong Yin, Simon See, and Yap-Peng Tan. Stochastic downsampling for cost-adjustable inference and improved regularization in convolutional networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[42] W Xiong, J Droppo, X Huang, F Seide, M Seltzer, A Stolcke, D Yu, and G Zweig. Achieving human parity in conversational speech recognition. arXiv:1610.05256, 2016.

[43] George Saon, Gakuto Kurata, Tom Sercu, Kartik Audhkhasi, Samuel Thomas, Dimitrios Dimitriadis, Xiaodong Cui, Bhuvana Ramabhadran, Michael Picheny, Lynn-Li Lim, et al. English conversational telephone speech recognition by humans and machines. arXiv preprint arXiv:1703.02136, 2017.

[44] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely Connected Convolutional Networks. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.

[45] Yunpeng Chen, Jianan Li, Huaxin Xiao, Xiaojie Jin, Shuicheng Yan, and Jiashi Feng. Dual Path Networks. In Advances in Neural Information Processing Systems 30, pages 4467–4475. Curran Associates, Inc., 2017.

[46] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. CoRR, abs/1707.07012, 2017.

[47] Yuxin Wu. Tensorpack. https://github.com/ppwwyyxx/tensorpack, 2017.

[48] Ilya Loshchilov and Frank Hutter. Sgdr. Stochastic Gradient Descent with Restarts. In International Conference on Learning Representations (ICLR), 2017.

[50] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the Inception Architecture for Computer Vision. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[52] Sam Gross and Michael Wilber. Training and investigating residual nets. http://torch.ch/blog/2016/02/04/resnets.html, 2016.

[53] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. In NIPS-W, 2017.
Appendix

The appendix illustrates the details of our experiments on object recognition and speech recognition and more ablation study.

A  \textit{bL-Net} for Object Recognition

A.1 Experimental Setup

We used the ImageNet dataset for all experiments. We trained all the models by Tensorpack [47], a higher-level wrapper for Tensorflow [48]. All the models were trained with 110 epochs, batch size 256, weight decay 0.0001, momentum 0.9 and Nesterov momentum optimizer. Furthermore, we used cosine learning-rate schedule as [49, 50]. We deployed the popular augmentation technique in [51, 52] to increase the variety of training data, and randomly crop a $224 \times 224$ patch as training image. The validation error is evaluated by resizing the shorter side of an image to 256 and then crop a $224 \times 224$ from the center. Note that the results reported here for the vanilla ResNet and ResNeXt models are difference from those reported in the original paper [1, 52]. Our vanilla ResNet is better than the original paper while vanilla ResNeXt is slightly worse than the original paper.

A.2 Network Structure

This section shows the details of network structures of our bL-ResNet, and the setting of $\alpha$ and $\beta$ is 2 and 4, respectively. To understand how do we design bL-ResNet-50 based on ResNet-50, Table 5 shows the details of network structure. We used a bottleneck residual block as a ResBlock, and a ResBlock, $C$ denotes a block composed of $1 \times 1$, $3 \times 3$, and $1 \times 1$ convolutions, where the first $1 \times 1$ and the $3 \times 3$ have $C/4$ kernels and the last $1 \times 1$ has $C$ kernels.

First, the Big-Branch and the Little-Branch shares a residual block at the transition layer, so the number of residual blocks in each branch will be subtracted by 1. The number of residual blocks in the Little-Branch is defined as $\left\lceil \frac{L}{\beta} \right\rceil - 1$ and at least one, where $L$ is the number of residual blocks in the Big-Branch, and the number of kernels in a convolutional layer would be $\frac{C}{\alpha}$, where $C$ is the number of kernels in the Big-Branch. Thus, for all stages, the number of blocks in the Little-Branch is only one, and the number of blocks in the Big-Branch would be the number of blocks in ResNet-50 minus one.

Table 5: Network configurations of bL-ResNet-50. Output size is illustrated in the parenthesis.

| Layers | bL-ResNet-50 | ResNet-50 |
|--------|-------------|-----------|
| Convolution | $7 \times 7, 64, s2 (112 \times 112)$ | |
| bL-module | $3 \times 3, 64, s2$ | $3 \times 3, 32$ | $3 \times 3, 32, s2$ | $1 \times 1, 64$ | $56 \times 56$ | MaxPooling ($56 \times 56$) |
| bL-module | ResBlock$_B$, $512 \times 2$ | ResBlock$_L$, $128 \times 1$ | ResBlock, $256 \times 3, s1$ | |
| | ResBlock, $256, s2 (28 \times 28)$ | | (56 \times 56) |
| bL-module | ResBlock$_B$, $512 \times 3$ | ResBlock$_L$, $256 \times 1$ | ResBlock, $512 \times 4, s2$ | |
| | ResBlock, $512, s2 (14 \times 14)$ | | (28 \times 28) |
| bL-module | ResBlock$_B$, $1024 \times 5$ | ResBlock$_L$, $512 \times 1$ | ResBlock, $1024 \times 6, s2$ | |
| | ResBlock, $1024 (14 \times 14)$ | | (14 \times 14) |
| ResBlock | ResBlock, $2048 \times 3, s2 (7 \times 7)$ | |
| Average pool | $7 \times 7$ average pooling | |
| FC, softmax | 1000 | |
We analyzed what advantages \textit{bL-Net} could provide as compared to the network which works on low resolution input directly (ResNet-50-\textit{lowres}). As shown in Table 7, ResNet-50-\textit{lowres} reduces lots of computations but its accuracy is not acceptable; however, \textit{bL-Net} (\(\alpha = 2\) and \(\beta = 4\)) achieves a better balance between accuracy and performance. A similar trend is also observed on a deeper model ResNet-101-\textit{lowres}. While such performance is unsatisfying compared to the state of the art, it is quite reasonable and expected given that almost \(3 \sim 4\) reduction of computation are achieved in such a case.

Figure 5 shows the prediction results from \textit{bL-Net} and ResNet-50-\textit{lowres}. When both models predict correctly (a) and (b)), the \textit{bL-Net} provides better confidence for the prediction. Because the object only occupies a small portion of an image, the Little-Branch can still capture the object clearly. On the other hand, when the key features of an object is small, like the shape of beak of a bird (c) and the spots of a ladybug (d), \textit{bL-Net} can easily retain that key feature to predict correctly while ResNet-50-\textit{lowres} provides wrong predicted label.

A.4 Ablation Study on Network Merging and Multi-Branch

Is linear combination better than concatenation? We adopt the simpler addition in \textit{bL-Net}. Nonetheless, if we design the Big-Branch in a way that the output channels is identical to the backbone networks, then the number of kernels in the Big-Branch would be only \(1 - \alpha\) with respect to...
Table 7: Performance of ResNets at different input resolutions.

| Network          | Top-1 Error | FLOPs ($10^9$) | Params ($10^6$) |
|------------------|-------------|----------------|-----------------|
| ResNet-50        | 23.66%      | 4.09           | 25.55           |
| ResNet-50-lowres | 26.10%      | 1.29 (3.17×)   | 25.60           |
| ResNet-101       | 21.95%      | 7.80           | 44.54           |
| ResNet-101-lowres| 24.80%      | 2.22 (3.51×)   | 44.57           |

Table 8: Different scales and merging schemes on bL-ResNet.

| Network                        | Top-1 Error | FLOPs ($10^9$) | Params ($10^6$) |
|--------------------------------|-------------|----------------|-----------------|
| ResNet-50                      | 23.66%      | 4.09           | 25.55           |
| bL-ResNet-50 (addition, $K = 2$) | **22.69%**  | 2.85 (1.43×)   | 26.69           |
| bL-ResNet-50 (concatenation, $K = 2$) | 24.04%      | 2.01 (2.03×)   | 20.57           |
| bL-ResNet-50 (addition, $K = 3$) | 24.12%      | 3.91 (1.04×)   | 27.23           |

the total number of kernels of the backbone network; thus, in this case, the overall bL-Net can be more efficient while the performance degradation could be compromised. We compared the performance of these two different merging schemes in Table 8. Although concatenation approach is more efficient, it performs much worse than addition with a gap of almost 1.5%. This leaves addition as a better choice for bL-Net in both visual and speech tasks.

**More-branch in bL-Net** As mentioned in Section 3.1, our approach can be extended to a scenario with multiple image scales. We experimented with three scales [1/4, 1/2, 1] on bL-ResNet-50 ($K = 3$) where ResNet-50 is served as the Big-Branch at the scale of 1/4 of the original input, i.e. 56 × 56. As indicated in Table 8, a 3-scale bL-Net requires more FLOPs and parameters due to the fact that the overhead in merging more branches is significant for ResNet-50, but even though, it still cannot provide superior performance of a 2-scale bL-Net. This is because the Big-Branch in the 3-scale bL-Net is downsampled aggressively by 4 times, thus substantially degrade the capability of feature representation in the Big-Branch.

Figure 3: Prediction results for bL-ResNet-50 and ResNet-50-lowres. True labels, predicted labels and their probability are listed in the table. When both models predicts correctly ((a) and (b)), bL-ResNet-50 achieves much higher probability; on the other hand, bL-ResNet-50 captures the details on the object and then predicts correctly ((c) and (d)).
Table 9: Network configurations of bL-ResNets applied to speech for acoustic modeling.

| Layers     | bL-Net-22 ($\alpha = 4, \beta = 1$) | bL-Net-22 ($\alpha = 2, \beta = 3$) |
|------------|--------------------------------------|--------------------------------------|
| Convolution| $32 \times T$                        | $5 \times 5, 64, s2$                 |
| T = 49 → 45|                                      |                                      |
| bL-module  | $(3 \times 3, 64)_B \times 2$        | $(3 \times 3, 32)_L \times 1$       |
| T = 45 → 35| $(3 \times 3, 32)_B \times 2$        | $(3 \times 3, 32)_L \times 1$       |
| transition layer | $16 \times T$                         | $(3 \times 3, 64, s2) \times 1$     |
| T = 35 → 33|                                      |                                      |
| bL-module  | $(3 \times 3, 128)_B \times 2$       | $(3 \times 3, 128)_L \times 1$      |
| T = 33 → 23| $(3 \times 3, 128)_B \times 2$       | $(3 \times 3, 128)_L \times 1$      |
| transition layer | $8 \times T$                         | $(3 \times 3, 128, s2) \times 1$   |
| T = 23 → 21|                                      |                                      |
| bL-module  | $(3 \times 3, 256)_B \times 2$       | $(3 \times 3, 256)_L \times 1$      |
| T = 21 → 11| $(3 \times 3, 256)_B \times 2$       | $(3 \times 3, 256)_L \times 1$      |
| transition layer | $4 \times T$                         | $(3 \times 3, 256, s2) \times 1$   |
| T = 11 → 9 |                                      |                                      |
| bL-module  | $(3 \times 3, 512)_B \times 2$       | $(3 \times 3, 512)_L \times 2$      |
| T = 9 → 1  | $(3 \times 3, 512)_B \times 2$       | $(3 \times 3, 512)_L \times 2$      |
| Convolution| $1 \times 1$                         | $4 \times 1, 512$                   |
| Convolution| $1 \times 1$                         | $1 \times 1, 32k$                   |

For each $L$ block, the first $3 \times 3$ convolution is with stride 2 (in frequency), and a bilinear upsampling is applied at the end.

For each $L$ block, a $1 \times 1$ convolution is applied at the end to match feature maps.

$s$: the stride is set to 2 in the frequency dimension (not in time) for the convolutional layer.

$T = T_0 \rightarrow T_1$ indicates that $T_0$ is the size of the time dimension at the start of the given layer, which is reduced to $T_1$ by the end of the layer.

## B bL-Net for Speech Recognition

### B.1 Experimental Setup

In our experiments, we start with an input size of $64 \times 49$, where 64 is the number of logmel filterbanks, calculated for each utterance on-the-fly. We also stack their first and second derivatives to get 3 input channels, resulting in our final input of dimensionality $3 \times 64 \times 49$. Our output is of size $\text{batch\_size} \times 512 \times 4 \times 1$, which is then projected to $\text{batch\_size} \times 512 \times 1 \times 1$ and finally to $\text{batch\_size} \times 32k \times 1 \times 1$ for classification. We then perform softmax cross-entropy over this output space of 32k tied CD states from forced alignment, doing phone prediction on the central frame of the input utterance. We report results after Cross-Entropy training, on Hub5’00 (SWB and CH part) after decoding using the standard small 4M n-gram language model with a 30.5k word vocabulary.

All models were trained in PyTorch [53] over 16 epochs on 2 GPUs with per-GPU batch size 256 (total batch size of 512), gradient clipping 10.0, weight decay $1 \times 10^{-6}$, nesterov accelerated momentum 0.9, and learning rate 0.03 (annealed by $\sqrt{0.5}$ per epoch $\geq 10$).

### B.2 Network Structures

**ResNet-22** Our models follow a ResNet architecture without padding in time [43], which accounts for the fact that padding in time adds undesirable artifacts when processing a longer utterance [4]. Under this constraint, each convolution operation reduces our input sequence in time by $k - 1$, where $k$ is the kernel width used. This effect can be seen in Table 9 in which the time variable, $T'$, is reduced in accordance with the number of convolutions. For similar reasons, when we stride we only do so in frequency, and not in time. For the rest of this section, when we refer to striding we are referring only to striding in frequency. We define our residual blocks as a series of $3 \times 3$ convolutions. When we transition from one stage to the next, we stride by 2 on the first convolution of the following...
Whenever downsampling is performed in the Big-Branch, it is only in the frequency dimension and, whereas we only want
Big-Little Module
bL-Net
with based on the ResNet-22 baseline. The
β
which we take
which is also presented in Table 9. This variant is well-defined in
variants end with the same projection and output layer as Res Net-22. All merges, unless otherwise
3
2
×
3
1
convolution to reduce the number of channels accordingly and fuse the two separate feature
Big-Little Module
bL-ResNet-22
(α = 4, β = 1) Table displays two bL-Net architectures that we experimented
with based on the ResNet-22 baseline. The bL-Net baseline, bL-ResNet-22 (α = 4, β = 1), consists
between each Big-Little Module and is well-defined through the parameters α and β. In
between each Big-Little Module we downsample our input using a transition layer consisting of a
residual block with a single 3 × 3 convolution with stride 2. Therefore, we shift one convolution
operation out of the last residual block in each stage that precedes a transition layer.
Whenever downsampling is performed in the Big-Branch, it is only in the frequency dimension and
not in time. Similarly, bilinear upsampling only occurs in the frequency dimension. All bL-Net
variants end with the same projection and output layer as ResNet-22. All merges, unless otherwise
specified, are through linear combination with unit weights per branch. For comparison, we experi-
mented with a version of bL-ResNet-22 (α = 4, β = 1) using concatenation to merge branches, the
results of which are presented in Table Using concatenation instead of linear combination in this
model results in each stage having more channels after concatenation than the current stage of the
network calls for (i.e. in stage 1 we end up with 64 + 16 = 80 channels at the end of the relevant
Big-Little Module, whereas we only want 64 channels to be outputted). To resolve this, we apply a
1 × 1 convolution to reduce the number of channels accordingly and fuse the two separate feature
maps.
bL-ResNet-22 (α = 4, β = 2, 3) We explored two more models where we fix α = 4, one in
which we take β = 2 and another where β = 3. All Big-Branches in these models are the same as
bL-ResNet-22 (α = 4, β = 1). The difference in each Little-Branch is based on the setting of β.
Since the number of convolutions we use in the bL-Net baseline is uneven in the first three stages, we
take \( \lceil L/3 \rceil \) to be the depth of the Little-Branch, where \( L \) is the depth of the Big-Branch. For \( \beta = 2 \),
the first three stages of the network have a Little-Branch consisting of one residual block with two
3 × 3 convolutions and one residual block with one 3 × 3 convolution. This results in a reduction
of the number of convolutions from 5 in the Big-Branch to 3 in the Little-Branch. For \( \beta = 3 \),
the first three stages of the network have a Little-Branch consisting of one residual block with two 3 × 3
convolutions, resulting in a reduction in the number of convolutions from 5 in the Big-Branch to 2 in
the Little-Branch. For both models, the Little-Branch in the final stage consists of a single residual
block with two 3 × 3 convolutions, since there are only four convolutions in the Big-Branch of the
final stage and \( \lceil 4/3 \rceil = \lceil 4/2 \rceil = 2 \).
In all bL-Net variants in which \( \beta > 1 \), because we can’t pad in time, we see that the time dimension
will get out of sync between the Big-Branch and Little-Branch. Therefore, before merging we need
to match the output size of each branch. To do this, we crop the shallower branches in time to match
the deepest branch (i.e. the Big-Branch will always have a smaller time dimension due to having
more convolutions, so we crop to match it). This is similar to the way the shortcut in ResNet is dealt
with in [43], and does not introduce edge artifacts when processing longer sequences.
bL-ResNet-22 (α = 2, β = 3) The last of our two-branch models is where α = 2 and β = 3,
which is also presented in Table This variant is well-defined in α and β up to the first three stages
of the network. In the last stage, however, we opt not to branch and instead follow identically the
final stage of ResNet-22 with two residual blocks operating at the full input resolution with 512
channels.
bL-PYR-ResNet-22 (α = 4, β = 1) We additionally present results on a pyramidal structure, in
which the first stage of the network operates with four branches, the second with three, the third
with two, and the fourth equivalent to the fourth stage of ResNet-22 (and bL-ResNet-22 (α = 2, β = 3)).
Due to the setting of α = 4, we increased the number of channels in the first stage
Big-Branch to have 256 channels (with the three Little-Branches in this stage having 64, 16, and 4
channels), avoiding a single channel on the smallest branch. Note that the middle branches require
both resolution upsampling and 1 × 1 convolution to match channels. The third stage Big-Little
Module operates on two branches and is identical to the analogous stage in bL-ResNet-22 ($\alpha = 4, \beta = 1$).