Inverted Residual Structure with SE and Conditional Convolution

Hsien Chi Kuo
hxk180054@utdallas.edu

Abstract

In this work, we propose an accuracy-latency balanced ImageNet structure which is inspired by MobileNetV2, MnasNet, MobileNetV3, and EfficientNet. We will show the network design, including standard building block, SE and conditional convolution as well as the result of after implementation.

1. Introduction

Accuracy is a factor of measuring model performance. The traditional way of increasing the accuracy is by scaling, that is, adding more neurons to the network. However, a large network takes huge amount of resources and time to train, making it not feasible on mobile or embedded devices. Over the past several years, better architecture such as MobileNetV2 [2], MnasNet [3] etc have achieved a good trade-off between accuracy and latency.

In this body of work, we introduce a relatively simple ImageNet. Firstly, we will give a brief description of our standard block, showing the identity and residual paths. Secondly, we will implement the Squeeze and Excitation method to the block. Last, we further improve the network by using conditional convolution. Table 2 shows the parameter for each standard building block. Table 3 shows the accuracy and training time. Table 4 shows the Rep per operation, MAC and coefficient counts.

2. Related Work

The network in this paper was heavily inspired by SENet [1], MobileNetV2 [2], MnasNet [3], CondConv [4], MobileNetV3 [5] and EfficientNet [6].

MobileNetV2 [2] used inverted residuals which its input and output of the residual block are thin bottleneck layer. It also removes non-linearities in the narrow layers to maintain representational power.

MnasNet [3] added squeeze and excite operations to the inverted residuals and used neural architecture search to optimize the network depth and width and propose a novel factorized hierarchical search space to enable layer diversity.

MobileNetV3 [5] replaced the swish nonlinearity with a hard swish nonlinearity, improved the design of the final encoder stages and used a new neural architecture search to optimize the depth and width. We present through experiment demonstrating on wide range of cases and mobile phone.

EfficientNet [6] refined the mobile baseline network and focused on the combined scaling of network input resolution, depth and width to design larger networks. It proposes a scaling method that uniformly scales all dimensions of depth, width and resolution using a compound coefficient.

A variant of EfficientNet replaced the convolution operations with conditional convolution operations. Conditional convolution operations increases the size and capacity of a network without sacrifice the efficient inference.

3. Design

It is a relatively standard ImageNet structure in figure 1 and table 1 with the 1st 2 stride by 2 operations change to a stride by 1 operation as the image size has 1/4 the rows and cols of typical ImageNet images. The network contains 3 parts, which are tail, body and head. The tail includes a 3x3 convolutional layer, a batch norm process and a ReLU.function. The body composed of 5 building blocks, which we will further elaborate in next paragraph. Lastly, the head consists of a global average pooling, Linear layer with bias.

Figures 2[A] shows the standard building block. For the residual path, firstly, the input go through 1x1 convolutional layer, batch norm, ReLU and increases the number of channel to 4 times of the original input channels. Secondly, it will go through another FxF/S fully grouped convolutional layer, batch norm, ReLU, where the output feature map size might shrink when S is not 1. Finally, it will be forward to another 1x1 convolutional layer and batch norm where the number of output channels decreases to the output channel we specified. If input channel Ni equals output channel No, and S = 1, then the input will go through identity path and later on add to the output of residual path.

Figure 2[B] shows that Squeeze and Excitation(SE) can also be implement between second and third step of residual path. We see that the SE process contains 3 steps. First step, we run through a global average pool which flattens the feature map. Second step, it goes through Linear layer with Bias and ReLU function, decreasing the number of channels by rank reduction ratio R. Last step we forward the output of previous step to a Linear layer with bias and sigmoid function. (see figure 3 for SE details)

Figure 2[C] shows how we implement SE and conditional convolution block. The conditional convolution uses 3D/4D convolution weight tensors Wm from M experts combined to a single 3D/4D weight tensor W via a learned input dependent weighted sum with fully grouped convolution or not fully grouped convolution.

There are a few key points on differences with this design and EfficientNet: this design only used 3x3 filters in the fully grouped convolution, modified the stem width, modified the depth and width of various blocks, modified the inverted residual expansion ratio, simplified the nonlinearity choice to ReLU (or Sigmoid where appropriate), only included 3 levels of down sampling as the cropped input is 3x56x56.

For implementations based on the SE enhanced building block, the internal rank reduction ratio R = 4.

For implementations based on SE and conditional convolution enhanced building blocks, the number of experts M = 4.

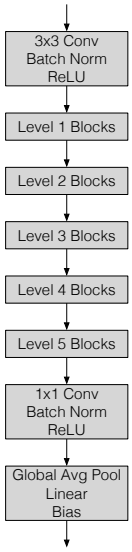

Figure 1: Network structure; the linear layer output dimension and bias dimension is the number of classes

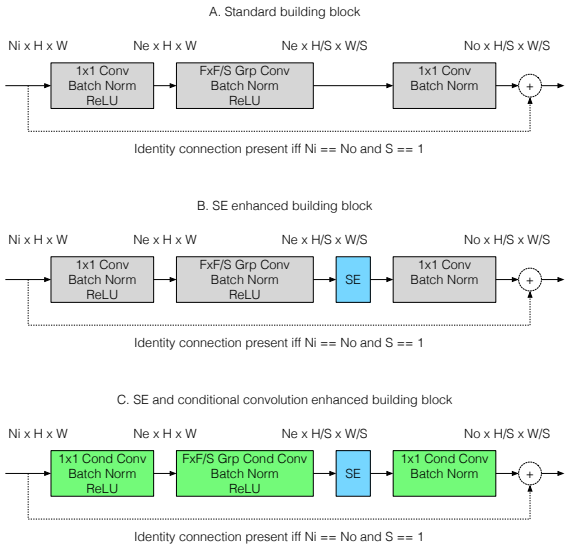

Figure 2: [A] Standard building block, [B] SE enhanced building block and [C] SE and conditional convolution enhanced building block

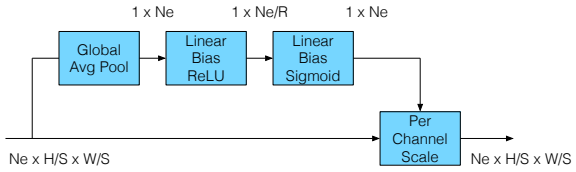

Figure 3: Squeeze and excite uses a learned input dependent per channel weighting to re weight input feature maps with internal rank reduction R

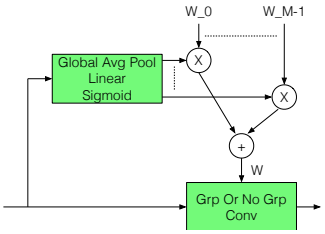

Figure 4: Conditional convolution uses 3D / 4D convolution weight tensors W_m from M experts combined to a single 3D / 4D weight tensor W via a learned input dependent weighted sum with fully grouped convolution / not fully grouped convolution

Re-peat	Input NixWxH	Operation
1	3x56x56	Conv (3x3/1), Batch Norm, ReLU
1	16x56x56	Block (Ne=4Ni, F=3, S=1, ID=True)
1	16x56x56	Block (Ne=4Ni, F=3, S=1, ID=False)
1	24x56x56	Block (Ne=4Ni, F=3, S=1, ID=True)
1	24x56x56	Block (Ne=4Ni, F=3, S=2, ID=False)
2	40x28x28	Block (Ne=4Ni, F=3, S=1, ID=True)
1	40x28x28	Block (Ne=4Ni, F=3, S=2, ID=False)
3	80x14x14	Block (Ne=4Ni, F=3, S=1, ID=True)
1	80x14x14	Block (Ne=4Ni, F=3, S=2, ID=False)
4	160x7x7	Block (Ne=4Ni, F=3, S=1, ID=True)
1	160x7x7	Block (Ne=4Ni, F=3, S=1, ID=False)
1	320x7x7	Conv (1x1/1), Batch Norm, ReLU
1	1280x7x7	Global Avg Pool, Linear, Bias
1	1x100	Output

Table 1: Network specification; block is either a [A] standard building block, [B] SE enhanced building block or [C] conditional convolution enhanced building block

4. Training

Table 2 includes a summary of all training hyper parameters. Note that this is a ~ generic ImageNet training routine such as you would find in RegNetX/Y [7]. Training routines that use more complex data augmentation, additional data, different train and test resolutions, more epochs, … can achieve higher accuracies.

Table 3 includes final training results and figure 5 shows a plot of the per epoch accuracy and loss curves.

Parameter	Value
epoch	55
learning rate initialize epoch	5
learning rate final epoch	50
initial learning rate	0.02
final learning rate	0.002

Table 2: Training hyper parameters

Block	Training time	Accuracy
Standard	5.5 hours	72.08
SE enhanced	3.5 hours	73.44
SE and cond conv enhanced	Optional	Optional

Table 3: Training final results

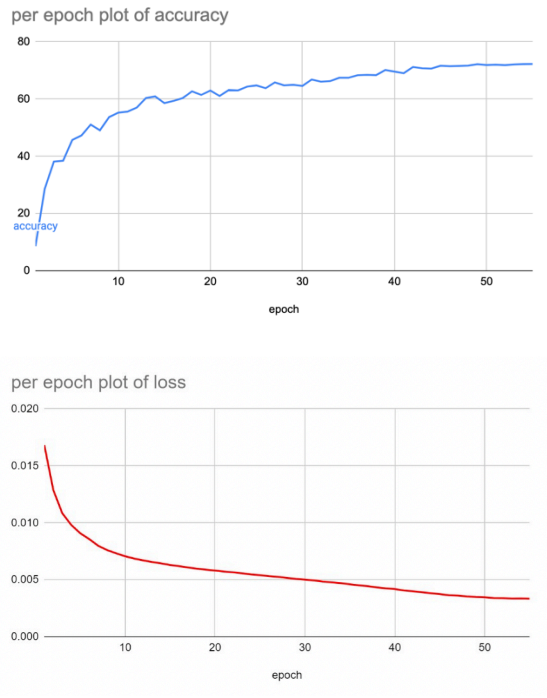

Figure 5: Training per epoch accuracy and loss curves

5. Implementation

Table 4 shows per operation MACs and number of filter coefficients for the stem convolution, convolutions in all standard blocks (taking into account repeats) and the head convolution and matrix multiplication, along with their sum for the full network.

Operation	Rep	MAC	Filter Cx
Conv (3x3/1), Batch Norm, ReLU	1	1354752	432
Block (Ne=4Ni, F=3, S=1, ID=True)	1	8228864	2624
Block (Ne=4Ni, F=3, S=1, ID=False)	1	9834496	3136
Block (Ne=4Ni, F=3, S=1, ID=True)	1	17160192	5472
Block (Ne=4Ni, F=3, S=2, ID=False)	1	10913280	7008
Block (Ne=4Ni, F=3, S=1, ID=True)	2	22328320	28480
Block (Ne=4Ni, F=3, S=2, ID=False)	1	7808640	20640
Block (Ne=4Ni, F=3, S=1, ID=True)	3	31799040	162240
Block (Ne=4Ni, F=3, S=2, ID=False)	1	7667520	79680
Block (Ne=4Ni, F=3, S=1, ID=True)	4	41269760	842240
Block (Ne=4Ni, F=3, S=1, ID=False)	1	15335040	312960
Conv (1x1/1), Batch Norm, ReLU	1	20070400	409600
Global Avg Pool, Linear, Bias	1	0	0
Total	–	193770304	1874512

Table 4: Per operation and total MAC and filter coefficient counts for all trainable operations

6. Conclusion

In this paper, we propose a relatively simple ImageNet, which is similar to those networks listed in the related work. We implement the network starting from standard building block, then add SE layer and conditional convolution layer to it. Hyperparameter and accuracy are shown in Table 2 and 3 respectively. Finally, Table 4 shows the block repeat times, MAC, and coefficient count.

This network gives a basic structure of ImageNet. In the future, we can continue try out different filter sizes, different widths, depths and repeats of different blocks, different residual expansion factors, different SE rank reduction factors, different numbers of mixtures of experts, and different training strategies to increase the accuracy of our network.

References

[1] J. Hu et. al., "Squeeze-and-excitation networks," arXiv:1709.01507, 2017.

[2] M. Sandler et. al., "MobileNetV2: inverted residuals and linear bottlenecks," axXiv:1801.04381, 2018.

[3] M. Tan et. al., "MnasNet: platform-aware neural architecture search for mobile," arXiv:1807.11626, 2018.

[4] B. Yang et. al., "CondConv: conditionally parameterized convolutions for efficient inference," arXiv:1904.04971, 2019.

[5] A. Howard et. al., "Searching for MobileNetV3," arXiv:1905.02244, 2019.

[6] M. Tan and Q. Le, "EfficientNet: rethinking model scaling for convolutional neural networks," arXiv:1905.11946, 2019.

[7] I. Radosavovic et. al., "Designing network design spaces," arXiv:2003.13678, 2020.