REPRODUCIBILITY REPORT OF MULTI-SCALE DENSE CONVOLUTIONAL NETWORKS

Jing Meng(29299675), Xiaoyu Wang(30549647), Yibo Liu(30546842)
School of Electronic and Computer Science, University of Southampton

ABSTRACT

In this project, we reproduce the image classification in the case of limited computing resources. We confirmed that the experiments by G. Huang et al. were reproducible. We compared three different types of networks (MSDNet, DenseNet and ResNet) with two data sets CIFAR-10 and CIFAR-100 for the classification of anytime classification and budgeted batch classification. Experiments show that the accuracy of MSDNet is indeed higher than that of the other two networks.

1 MODELS

1.1 RESNET AND DENSENET

The proposal of ResNet (He et al. (2016)) solves the problem of vanishing gradient when the depth of the network is deepened. Its innovation is the introduction of residual blocks. Except a typical general convolutional network structure, a shortcut from the input to the output is added, which enables information to flow directly through the identity function from non-subsequent layers .

The author of DenseNet (Huang et al. (2017b)) connect all layers directly with each other to ensure maximum information flow between layers in the network. General speaking, the input to each layer comes from the output of all the previous layers. One of the advantages of DenseNet is that the network is narrower and has fewer parameters. The main reason is the design of the dense block. The number of output feature maps (growth rate) for each convolutional layer in the dense block is rather small, which is achieved by the introduction of the bottleneck layer. In order to further reduce the parameters, using the transition block to connect two dense blocks.

1.2 MSDNET

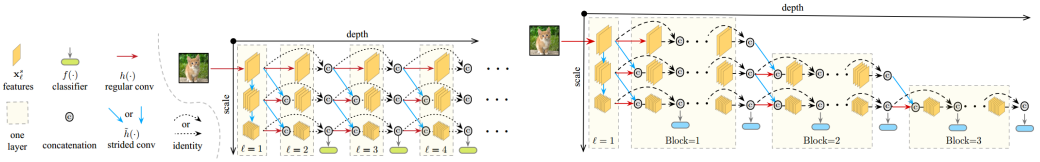
1.2.1 PROBLEM SETUP

The proposal of MSDNet intends to solve the problem for ensuring the accuracy of image classification under the condition of computation resource constrained scenario such as mobile devices and huge amounts of data. The two corresponding proposed questions are anytime classification and budgeted batch classification. The purpose of the questions is to provide both the coarse classification and fine classification on the same model. The solution is to add scale layers vertically based on the existing layers through strided convolution. The scale layers provide coarse features for classification (Huang et al. (2017a)).

1.2.2 ARCHITECTURE

- **The First Layer** The special part is strided convolution as shown in blue line [1(a)], which diagonal connect previous scale layer to next scale layer. Its main function is to provide the means ”seed” on all scales. The feature map of coarse scale is obtained by down sampling.
- **Subsequent layers** The Subsequent layers concatenate previous layers' feature maps following the DenseNet structure. The number of output channels of the three scales is set to be 6, 12 and 24 respectively.
- **Classifiers** Each classifier has two down-sampling convolutional layers with 128 dimensional 33 filters, followed by a 22 average pooling layer and a linear layer. Middle classifiers are added after each block.

- **Network reduction and lazy evaluation** The lazy evaluation is implemented through transition layers, which is added between two adjacent blocks, and reduce number of features by half, wich bottleneck(1×1) convolution 1(b).



(a) MSDNet with three layers (b) Lazy evaluation

Figure 1: MSDNet Architecture

2 Experimental Methodology and Implement Details

The reproducibility project adopts the deductive methodology, follows the algorithms proposed by the original paper step by step to verify the results. We use python as programming language and naturally use Pytorch for deep learning programming. The implementation of the code refers some parts of following code bases git (b) git (a). 1) The original implementation from the author of the paper is *[https://github.com/gaohuang/MSDNet]*. Our code refers the parameters setting of command line for building anytime prediction and budgeted batch computation. 2) *[https://github.com/kalviny/MSDNet-PyTorch]* Our project mainly refer to the code architecture of MSDNet. In addition, We implement the FLOPs calculation for all middle classifiers in once training. For $ResNet^{MC}$ 2(a) and $DenseNet^{MC}$ 2(b). We add early exist classifiers individually in each needed layers as middle classifiers. Finally, we implement all comparison visualisation.

The MSDNet used in anytime-prediction experiments has 24 layers. The classifiers operate on the output of the $2 \times (i+1)^{th}$ layers, with $i = 1, ..., 11$. The 62-layer ResNet consists 10 residual blocks at each spatial resolution: classiers on the output of the 4th and 8th residual blocks at each resolution (plus the nal classication layer). The 52-layer DenseNet consists three 16-layer dense blocks. The six intermediate classiers are attached to the 6th and 12th layer in each block. The other parameters except epochs are all set as paper stated: the optimizer is stochastic gradient descent, the mini-batch size is 64 and initial learning rate is 0.1. We use Nesterov momentum with a momentum weight of 0.9 without dampening, and a weight decay of 10^{-4}. Because of time-consuming of computation and computer resource limitation, we implement on data sets CIFAR10 and CIFAR100, and set epochs as 30 for training. We used Google Colaboratory Notebook for all experiments, with the configuration 12 GB RAM and Nvidia K80 GPU. The execution time is around 2 hours for MSDNet training.

Layer name	Output size	62-layer ResNet
Conv1	16x16	7x7,64,stride=2,padding=3
Conv2_x	8x8	[3x3,64; 3x3,64] x 10
Conv3_x	4x4	[3x3,128; 3x3,128] x 10
Conv4_x	2x2	[3x3,256; 3x3,256] x 10
Classification	1x1	Average pool, 10/10-d for cifar 10/100 fc, softmax

Table 1. Architectures of 62_layer ResNet for Cifar 10/100. Both input size of two datasets is 32 x 32.

(a) $ResNet^{MC}$

Layer name	Output size	52-layer DenseNet
Conv1	32x32	3x3conv, stide=1,padding=1
Dense block 1	32x32	[3x3,12; 3x3,12] x 8
Transition Block 1	32x32	1x1 conv
	16x16	2x2 average pool, stride=2
Dense block 2	16x16	[3x3,12; 3x3,12] x 8
Transition Block 2	16x16	1x1 conv
	8x8	2x2 average pool, stride=2
Dense block 2	8x8	[3x3,12; 3x3,12] x 8
Classification	1x1	8x8 global average pool
		10/100-d fc ,aoftmax

Table 2. Architectures of 52-layer DenseNet for Cifar 10/100. Both input size of two datasets is 32 x 32.

(b) $DenseNet^{MC}$

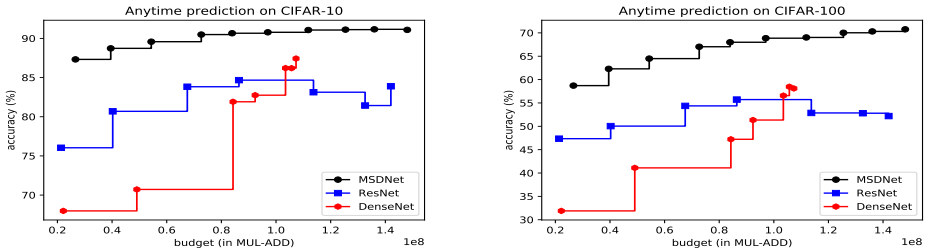
Figure 2: $ResNet^{MC}$ and $DenseNet^{MC}$ Architecture

3 Results and Analysis

3.1 Comparison of three models for anytime prediction

We simply use competitive ResNet and Densenet architectures as the base networks in the process of reproduction. Computational budget is controlled by introducing multiple early-exit classifiers throughout a network, which are applied to the features of the particular layer they are attached to.

On the datasets of CIFAR-10 and 100, the trends of the two images are consistent and the results of the paper are basically reproduced. Under the condition of same flops, the accuracy of MSDNet significantly outperforms the ResNet and DenseNet. Especially at the early layer, when the budget ranges from $0.2-0.3\times10^8$ FLOPs, MSDNet has an obvious advantage. MSDNet achieves 10% and 20% higher accuracy than ResNet and DenseNet respectively on CIFAR 10. And it maintains the advantage at CIFAR100. Although the gap has decreased in the later early, MSDnet still performs better than the other two models. This can prove the fact that MSDNets can abstract low-resolution feature maps that are much more suitable for classification than ResNets or DenseNets after just a few layers. Notably, compared with computational efficient DenseNet, MSDNet achieves the same accuracy at 87% in [3] using only one-fifth of the computations on CIFAR 10 and it is up to 5 times more efficient than DenseNet. This is of great significance for the study of model compression.

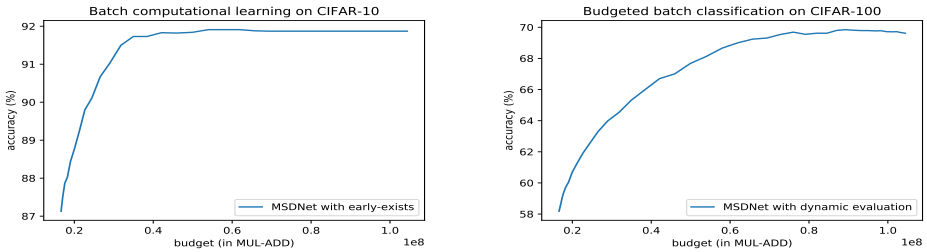


(a) CIFAR10 (b) CIFAR100

Figure 3: Anytime Prediction

3.2 The budgeted batch classification for MSDNet

The basic idea of the implementation of the budgeted batch classification is after training a MSDNet model, use the validation set to find a confidence threshold p for each of the early existing classifiers, and then at test time terminate the computation once a test sample hits the threshold at a certain classifier. Given a value of p, get an average computational cost over a batch of test samples. We train an MSDNet(24 layres) with 7 classifiers with the span linearly increases for efficient batch computation. The result is as follows [4]. From the figures, it shows that batch computation also can achieve the same accuracy as anytime prediction. On CIFAR-10, the accuracy of batch computation is a lit bit higher than anytime prediction, whereas, on CIFAR-100, the result is opposite.



(a) CIFAR10 (b) CIFAR100

Figure 4: Batch Computation

4 Extension

The aim of this part is to verify whether replacing one regular convolutional block with GCN block would reduce the accuracy significantly while reducing the parameters.

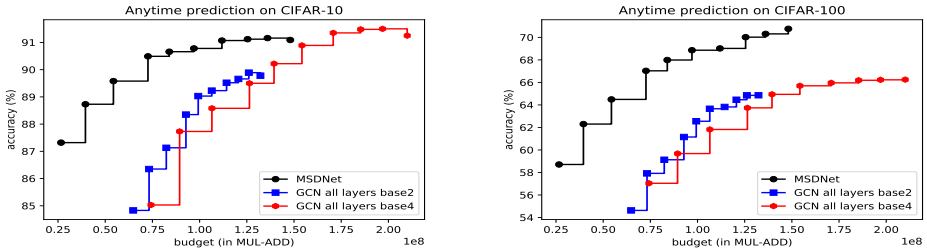
4.1 Global Convolution Network(GCN)

It is known that the effect of stacking smaller filters is higher than bigger filters in the same computational complexity case. The parameters can be greatly reduced by using a smaller kernel filter. The GCN module combines $1\times k+k\times 1$ and $k\times 1+1\times k$ convolutions instead of directly using

larger kernel or global convolution, which enables dense connections within a large $k \times k$ region in the feature map (Peng et al. (2017)).

4.2 Comparison of MSDNet and GCN on all layers

We use the kernel size with 5×5 and base block containing 2 layers and 4 layers respectively for replacing all spatial convolutions in the MSDNet. It can be indicated from [5] that the replacement with GCN reduces the performance for original MSDNet. The accuracy gap in the lower FLOPs is higher than in the higher FLOPs. The degree of accuracy decrease go down with FLOPs increasing. Yet, the accuracy decrease is not big in generally. Another point can be found that the base block containing more layers will result in parameters increasing. The multiple for increasing parameters is almost the same as increasing contained layers.



(a) GCN on CIFAR10 (b) GCN on CIFAR100

Figure 5: MSDNet VS GCN

5 Conclusion

The results of MSDNet paper are reproducible. Different from traditional convolutional networks whose features only become coarser with increasing depth, the MSDNet generates e coarse-scale features in early layers and maintains them throughout. The MSDNet architecture is unprecedented efficient. However, we find that the accuracy of ResNet does not change much with the position of the classifier. The number of 52-layer DenseNet FLOPs differs greatly from the paper. The way of calculating FLOPs is not elaborated and can be confusing. Combining MSDNet and GCN will reduce the performance of MSDNet.

As future work, we would test experiments on ImageNet which we could not complete due to time and resource constraints. In addition, we intend to compare the budgeted batch computation of MSDNet, ResNet and DenseNet and combine the MSDNet with pruning technology to reduce computation further.

References

Condensenet. `https://github.com/ShichenLiu/CondenseNet`, a.

Msdnet-pytorch. `https://github.com/kalviny/MSDNet-PyTorch`, b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Gao Huang, Danlu Chen, Tianhong Li, Felix Wu, Laurens van der Maaten, and Kilian Q Weinberger. Multi-scale dense networks for resource efficient image classification. *arXiv preprint arXiv:1703.09844*, 2017a.

Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017b.

Chao Peng, Xiangyu Zhang, Gang Yu, Guiming Luo, and Jian Sun. Large kernel matters–improve semantic segmentation by global convolutional network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4353–4361, 2017.