AVLSI 2017 FALL FINAL REPORT

ZEI-WEI TONG, BO-HONG CHO, YU-HSIU SUN

Department of Electrical Engineering, National Taiwan University

ABSTRACT

As machine learning has been a prevailing technique in variational realms, the corresponding hardware device requirement has also been put under spotlight. In this work, we investigate how different network architectures and its corresponding hardware computational cost affect on the Tiny ImageNet dataset. Starting with the original model AlexNet, we adapt residue method and depthwise convolution method to it. From experiment, we provide insight into trade-off between accuracy and computation cost and come up with a new distributed model among previous work.

Index Terms— MobileNet, Depthwise Convolution, AlexNet, Tiny ImageNet, Image classification, Efficiency

1. INTRODUCTION

Image classification has been an exciting research field in computer vision. Self-driving system require to distinguish between human and traffic light. Security system require detection on intruder and ignore the floating leaf in front of the camera. During the past, expert try to handcraft clever feature to get the similarity and difference between objects, and work well no matter sunny or cloudy, which significantly make the capture image brighter or darker. Result is far from satisfied. State-of-the-art hand crafted performance on ILSVRC 2010 achieve only 72.2% accuracy (i.e. top-5 accuracy)[1].

At 2012, a new method had been proposed, quickly showed its potential on making acceptable accuracy on image classification task. Empower by new semiconductor production process and GPU acceleration, neural network, once been regarded as impossible to realize due to extreme high computation cost, now achieve astonishing performance among almost all kinds of contest. However, high accuracy usually means large model that can only train on workstation with powerful GPU. The high power consumption and large memory requirement make training process impractical on device like cctv in security system, or on car collision prevention camera system.

Some might argue that we can use off-line training and only inference be applied on device. Obviously it's a appealing solution, and it indeed be practiced on much contemporary application. Nevertheless, we still believe on device training is necessary. First, transmission of model would have security issue. Model use for security system, if known by unauthorized third party, have risk of being "hacked". As shown in [2], single pixel change can lead to miss classified.

Second, on device training can help the system more robust. A general, off-line pre-trained model can't solve unique problem on particular device. In real situation, most of the device handle only limited input condition. A camera in ATM room don't need to detect when neighbor throw garbage into your yard, which might be critical for the camera hang in your front door. On device training can use smaller model that fit best to its input type and surroundings.

Finally, aiming for low power consumption should never be look down, especially in a time when ecologically friendly become a social desperation. The well-known AlphaGo consume over 75kW (only GPU part, even not consider the 1202 CPUs), which almost 3000 times more than the power needed by one person.[3] Our research focus on the performance and efficiency comparison and analysis of state-of-the-art architecture, which is critical for on device training and also off-line training for which power consumption matters.

2. BACKGROUND

2.1. Dataset

We perform our task on Tiny ImageNet[7], the default data set for the course project for Stanford CS231N[8], which contains 10000 pictures of training set, 10000 picture in validation set. The pictures are sourced from 200 classes. Tiny ImageNet is similar to the data set used in ImageNet Challenge (ILSVRC)[9], but the images are downscaled into 64x64x3.

2.2. AlexNet

In 2012, Alexnet[1] was presented, proving the promising advantage of deep convolutional neuron-network in the field of image recognition. The architecture of Alexnet, shown in fig. ,contains five convolutional layers and 3 fully-connected layers, using ReLU as activation function. Before Alexnet, the standard activation function is tanh, which suffered from gradient descent, using ReLU as activation function could effectively avoid this problem, making the training task faster than

Thanks Prof. Andy Wu provide us such a great chance to explore this field.

the counterpart using tanh. Alexnet achieve top-5 error rate of 15.3% on ImageNet in ILSVRC 2012.

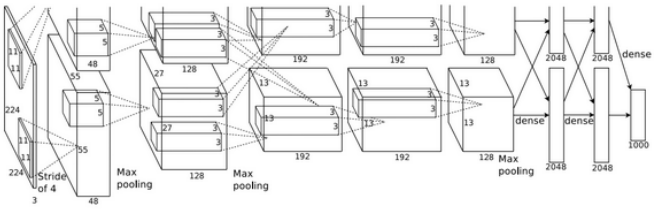

Fig. 1. AlexNet architecture[]

2.3. Residue

After Alexnet, the CNN models used in image recognition became deeper and deeper to pursue higher accuracy, but deeper network are more difficult to train. The main problem that deep networks encounter, is severe vanishing gradients that hampers the convergence of the models. With the network depth increasing, the accuracy get saturated, or even get worse in some cases. In 2014, ResNet[4] present a framework called deep residual learning framework to resolve the problem. The basic building blocks of residual learning framework is shown in fig. . By adding shortcut connection in conventional feed-forward network to bypass one or more layers. The 152-layer residual network won the 2015 ILSVRC competition with top-5 error rate of 3.57%.

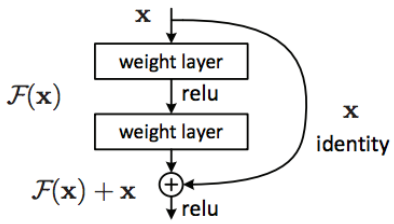

Fig. 2. Residue path[]

2.4. Depthwise Convolution

Depthwise convolution[5] was proposed in the work of Inception v3, which breaks down convolutions into smaller convolutions. While larger spatial filters(e.g. 5 x 5 or 7 x 7) inclined to be disproportionally expensive in terms of computation cost. For example, a 5 5 convolution with n filters over a grid with m filters is 25/9 = 2.78 times more computationally expensive than a 3 3 convolution with the same number of filters. We can replace the the fully connected 5 x 5 convolution by a two layer convolutional architecture: the first layer is a 3 3 convolution, the second is a fully connected layer on top of the 3 3 output grid of the first layer (see Figure 1). Sliding this small network over the input activation grid boils down to replacing the 5 5 convolution with two layers of 3 3 convolution.[Fig.3]

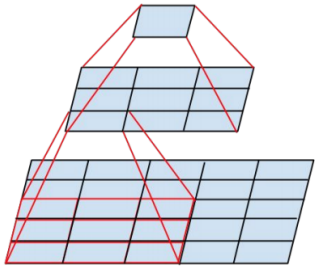

Fig. 3. AlexNet architecture[]

3. PROPOSED METHOD

3.1. Preprocessing

For the various models we trained, we preprocess the Tiny ImageNet dataset by channel wise normalizing among pixels, which means we reduces channel wise mean then divide channel wise standard to each pixel. We augmented the dataset by masking, rotating, mirroring or applying some random noise.

3.2. Architecture

3.2.1. Original Model

Our first model is inspired by Alexnet, shown in fig. ,containing five convolutional layers and one fully connected layers, and max pooling layers after the first and the second convolutional layers. Different from Alexnet, we put a average pooling layer after the fifth convolutional layer, instead of max pooling layer used in Alexnet.The first convolutional layer filters the 64x64x3 input with 32 kernels of size 3x3x3 with stride of 2 pixels. The second convolutional layers takes the input of 31x31x32, and filters it with 64 kernels of size 3x3x32 with stride of 2 pixels. The third convolutional layer has 128 kernels with size 15x15x64. The fourth and the fifth convolutional layers have 128 kernels with size 7x7x128.

3.2.2. Depthwise Convolution

Inspired by the concept of depthwise convolution, which is an effective method to decrease the complexity of CNN network. We change the second to fifth convolutional layers from conventional convolutional layer into corresponding depthwise convolutional layers. If a conventional layer has a filter of NxDxDxM, in our second the conventional layer will be convert into a filter of MxDxDx1 followed by a filter of Nx1x1xM, as shown in fig. . Take the second convolutional layer as an example. Conventionally, the convolutional layers

has 64 layers with size of 31x31x3. In our second model, the output from the first convolutional layers with size of 31x31x32 would first entered a layer containing 32 3x3x1 filters, and then enters a layers containing 64 laters of 1x1x32. The same concept is used in the third, fourth and the fifth convolutional layers.

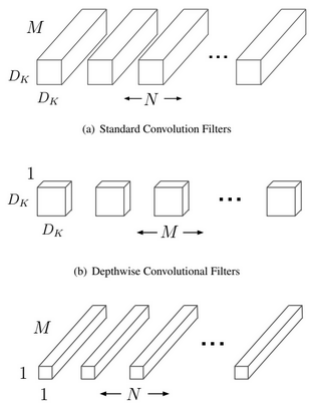

Fig. 4. Depthwise convolution[]

3.2.3. Residue

In our second model, we use depthwise convolutional layers to construct a model with less parameters. Considering the expected degeneration of accuracy coming with a model with less parameters, we perform the concept present in ResNet, by adding shortcut path in the second model. Since the dimension of the end and the start of the shortcut path must be the same, there are only two paths that can be added in our second model. As shown in fig. , a shortcut path bypass the fourth layer, and the other bypass the fifth layers.

4. EXPERIMENT

4.1. Training curves

We implement our task with Keras 2.0 and tensorflow on NVIDIA GTX 1080 Ti. The training work consist of 100 epochs.

In Fig.5 , the training curve of our alexnet-like network is shown. The training accuracy can converge to around 96%, but the accuracy on validation set is limited to about 33%. In our first model. it suffers severe overfitting, which may causes by the small data set we used to train the model, or by the absence of dropout layer.

The training curve of our second model is shown in Fig.7. We can find that the second model has much lower accuracy in training set, which drops from 96% to 48%. As expected the less complex model , since we try to use the linear combination of two less complex layers to replace conventional convolutional layers. The result shows that the linear combination

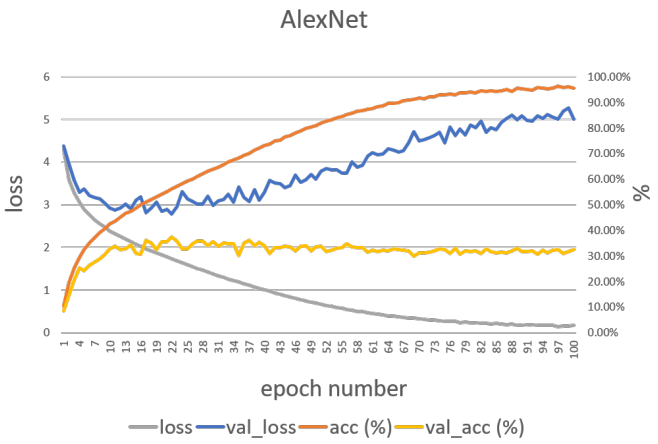

Fig. 5. (a) AlexNet[]

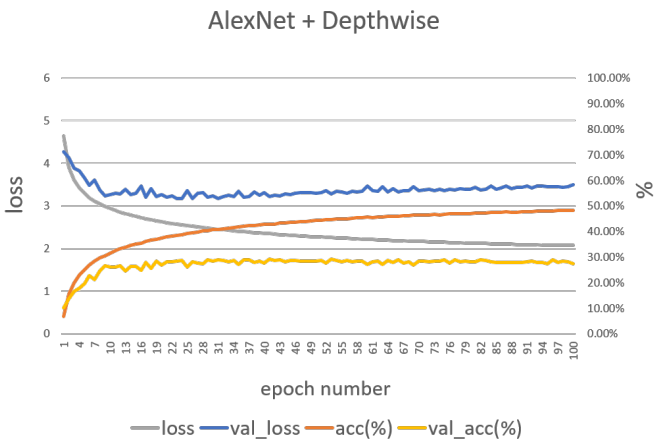

Fig. 6. (b) with depthwise convlution[]

cannot perfectly represent the original convolutional layers. The possible method to improve the degeneration could be making a deeper network or using residual framework as we implement in our third model.

The training curve after adding the residual path is shown in fig.7. Comparing to the previous model, the accuracy on training set and validation set are slightly improved.

In fig. 8, it shows the comparison of the performance of three models. We can tell that, although the depthwise model has a much worse training accuracy, the validation them is almost the same, indicating that using depthwise convolutional layers is a reasonable method to decrease the number of parameters..

4.2. Efficiency

The comparison of the efficiency of our model is shown in fig.9. In our Alexnet-like network, there are 415,656 param-

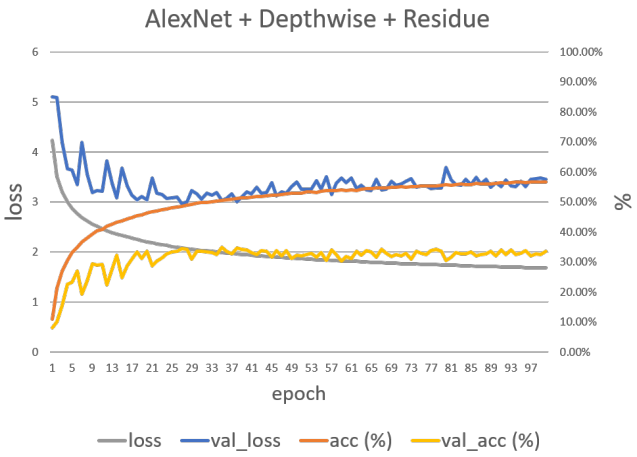

Fig. 7. (c) add residue path[]

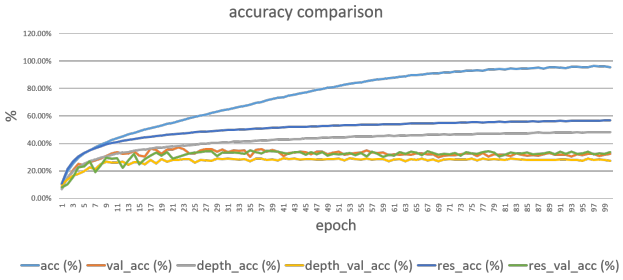

Fig. 8. (d) comparison between three models[]

eters. After the replacement of the convolutional layers, the number of parameter reduced to 74,818, about 18% of the original number. Thanks to the reduced numbers of parameter, the training time per epoch reduced from 34 seconds into 12 seconds. In our third model, although the number of the parameters is equal the second one, the model with residual path suffers a longer training time, since the shortcut path increase the computation of back propagation.

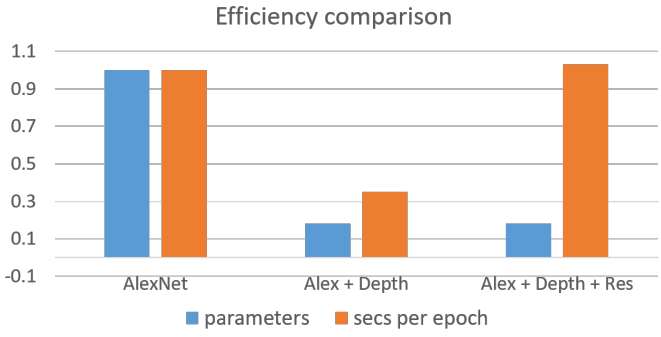

Fig. 9. (d) comparison between three models[]

4.3. Top 5/ Bottom 5 accuracy classes

The top-5 accuracy classes of ours are respectively dungon, trolleybus, bell pepper, scoreboard, black widow which has approximately 70% correct rate. As we can see in the Fig.10, they are easily formulated in the whole figure. For example, dungons picture are formulated by dungon itself and a background of water which is generally blue. Hence, model can easily get the generalization of features. On the other hand, Fig.11 the worst-5 accuracy classes are plunger, Chihuahua, reel, bucket, bow tie. These classes have a similar phenomenon in picture formulation, they are often taken with human which is a great disturbance in generalization of features.

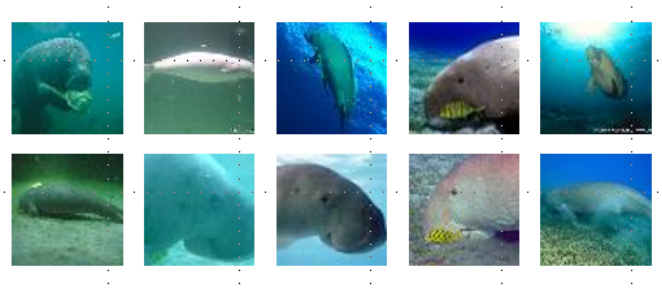

Fig. 10. Best predict class (Dugong)

Fig. 11. Worst predict class (Plunger)

5. CONCLUSION

5.1. Our work

In our research, we provide a clear comparison between traditional convolution layer and depthwise convolution, also the effect of residue on shallow model (i.e. by adding identity path to our model). First, depthwise convolution, based on the assumption of depth tensor can approximate with combination of a set of 2d filter, yield lower accuracy on training data, compare to traditional convolution layer. Though the lower performance didn't show on validation data, this would not be a drawback of the new technique. On the other hand, depthwise make the parameter size much less, providing a

magnificent efficiency improvement on our proposed model. Trying to regain the loss performance due to depthwise, we introduce the residue method in our model. It turns out to have slightly improvement, but with a high computation cost trade-off.

Second, we analysis the classes with highest and lowest accuracy on our model. Classes with high accuracy usually contain "good-training" training images: image with simple background and clear and single topic. The low accuracy, in contrary, often contain many object in single images and with complex background. The low validation performance would be one result of poor training data.

An accuracy trade-off with less computation cost is always the main issue when new algorithm try to apply to real application. We wish our study can provide a new point of view to this trade-off, and inspire more researcher to try to push beyond the limit of technology.

5.2. Future work

Distributed model can achieve higher performance under same scale of computational complexity by combining different processing units or devices. As we can infer from the network in asynchronous distributed model[6], the structure of our model can be implemented as [fig]. There exist numbers of model with same structure, and we define one model as the globat network, which will be our final model used in testing case. In the training process, the model of each sub-models are trained independently. At the end of every T epochs, all the sub-models will update their parameters to the global net (where their structure are same thus the parameters of all networks should fit perfectly) and pull the new updated parameter to each sub-model. Since we have seen the great performance and efficiency on the work in reinforcement learning counterpart, we believe it is a interesting topic to further study in future.

5.3. Reference

6. REFERENCES

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton. *Imagenet classification with deep convolutional neural networks.* . In Advances in neural information processing systems, pages 10971105, 2012.

[2] Jiawei Su, Danilo Vasconcellos Vargas, and Sakurai Kouichi. One pixel attack for fooling deep neural networks. *arXiv preprint arXiv:1710.08864*, 2017.

[3] AlphaGo uses more power than 3000 humans - Draketo. [Online]. Available: http://www.bing.com/cr?IG=BDB020BFD4A84148841B7BC914154206&CID=1E7544C4E1B866C315F44FBDE01767DD&rd=1&h=xjVBYD_xhrTY-2pf19xQu4Kop_ecm86AaHqTj-AXuB0&v=1&r=http%3a%2f%2fwww.draketo.de%2fenglish%2falphago-power&p=DevEx,5067.1. [Accessed:16-Jan-2018].

[4] He, K., Zhang, X., Ren, S., Sun, J *Deep residual learning for image recognition.* . In: CVPR. (2016)

[5] Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. *Rethinking the inception architecture for computer vision.* arXiv preprint arXiv:1512.00567.

[6] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, *Asynchronous methods for deep reinforcement learning,.* in ICML, 2016.

[7] Tiny ImageNet Visual Recognition Challenge, Tiny ImageNet Visual Recognition Challenge[Online]. Available: https://tiny-imagenet.herokuapp.com/. [Accessed: 16-Jan-2018].

[8] CS231n: Convolutional Neural Networks for Visual Recognition, Stanford University CS231n: Convolutional Neural Networks for Visual Recognition.[Online]. Available: http://cs231n.stanford.edu/. [Accessed: 16-Jan-2018].

[9] ImageNet, ImageNet.[Online]. Available: http://www.image-net.org/. [Accessed: 16-Jan-2018].