Impacts of Add-based Modules on Model Performances

Jiawei Gu, Yuzhe Chen, Yifei Zhang

Tsinghua-Berkeley Shenzhen Institute, Tsinghua University, Shenzhen University Town, Guangdong 518071, China

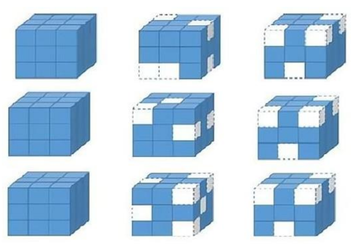
ABSTRACT

Convolutional Neural Networks (CNN) are widely used in various image-processing tasks. However, to achieve high inference accuracy, many CNNs incorporate large-scale models, leading to a huge amount of calculation and large power consumption. On the other hand, Adder-based modules simplify the computation by changing multiplication to addition operations, and can be applied to replace the convolution part of the neural network. In this project, we examine the effects of such modules on different network structures, regarding inference speed and accuracy.

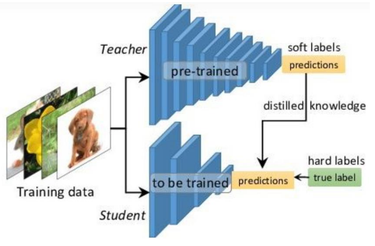
INTRODUCTION

In order to achieve better results in image processing tasks, deep convolutional CNN networks are usually applied. However, these convolution operations often introduce a large number of calculations, causing a heavy computational workload. This seriously affects the deployment of these models to peripheral and mobile devices, such as smartwatches, smartphones, drones, VR glasses, and other mobile terminals. The high requirement of computational resources and supporting devices severely limits the models' processing capabilities and usages on portable mobile terminals. In order to facilitate model deployment, one usually needs to compress the model. The classic convolution can be regarded as a measure of the similarity of the input in essence. We consider another way to perform calculations in order to effectively reduce the calculations in model reasoning.

Two common methods of model compression



Pruning



Knowledge Distillation

METHOD

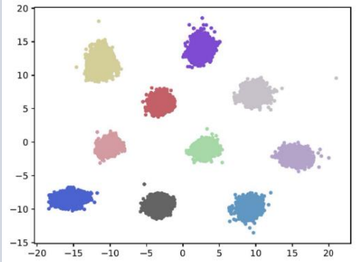
AdderNet model:

- Replaces the **multiplication** by **addition**.
- Mathematical formulation:

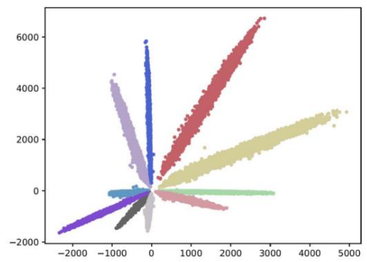
$$\mathrm{Y(m,n,t)} = \sum_{i=0}^{d}\sum_{j=0}^{d}\sum_{k=0}^{c_m} \mathrm{X(m+i,n+j,k)} * \mathrm{F(i,j,k,t)} \longrightarrow Y(m,n,t) = -\sum_{i=0}^{d}\sum_{j=0}^{d}\sum_{k=0}^{c_m} |X(m+i,n+j,k) - F(i,j,k,t)|$$

- An alternative way of **similarity measurement** -- achieves similar effects as **convolution** operation
- We evaluate AdderNet on two datasets (CIFAR-10, CIFAR-100).

Visualization of features in different networks



(a) AdderNets



(b) CNNs

Parameter update scheme:

In CNNs, the partial derivative of output features Y with respect to the filters F is calculated as:

$$\frac{\partial Y(m,n,t)}{\partial F(i,j,k,t)} = X(m+i,n+j,k)$$

In AdderNets, the partial derivative of Y with respect to the filters F is:

$$\frac{\partial Y(m,n,t)}{\partial F(i,j,k,t)} = \mathrm{sgn}(X(m+i,n+j,k) - F(i,j,k,t))$$

However, **signSGD** almost never takes the direction of steepest descent and the direction only gets worse as dimensionality grows.

$$\frac{\partial Y(m,n,t)}{\partial F(i,j,k,t)} = X(m+i,n+j,k) - F(i,j,k,t)$$

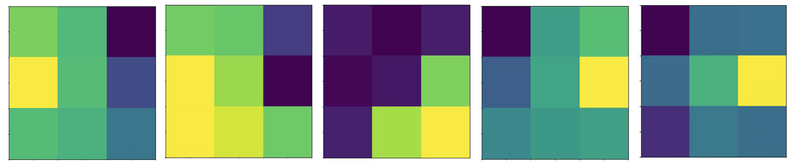
To this end, we clip the gradient of X to [−1, 1] to prevent gradients from exploding. Then the partial derivative of output features Y with respect to the input features X is calculated as:

$$\frac{\partial Y(m,n,t)}{\partial X(m+i,n+j,k)} = \mathrm{HT}(F(i,j,k,t) - X(m+i,n+j,k))$$

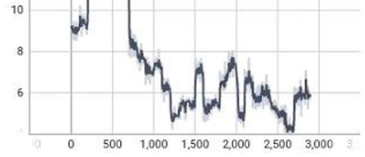
$$\mathrm{HT}(x) = \begin{cases} x & \text{if} \quad -1 < x < 1, \\ 1 & x > 1, \\ -1 & x < -1. \end{cases}$$

RESULT

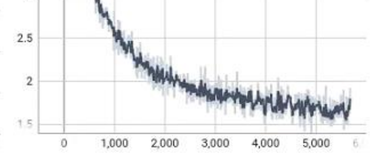
Visualization of convolutional kernels -- similar to CNNs, able to extract features.



loss curves: cifar-10, 400 epochs

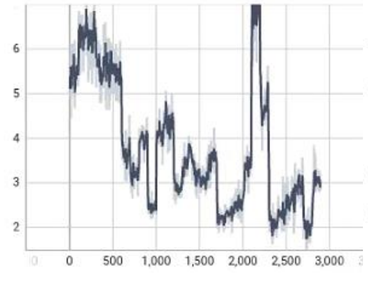


(a) Test loss

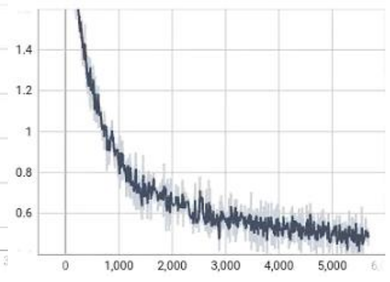


(b) Training loss

loss curves: cifar-100, 400 epochs



(a) Test loss



(b) Training loss

REFERENCES

[1] AdderNet: Do we really need multiplications in deep learning?[C]//Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020: 1468-1477.

[2] Gholami A, Yao Z, Kim S, et al. Ai and memory wall[J]. RiseLab Medium Post, 2021, 1: 6.