A parallel version of deep learning

XIAOTIAN GAN 50207987

content

A brief summary of deep learning and google research’s work

A approximation parallel algorithm of deep learning

Experiment result and future work

content

A brief summary of deep learning and google research's work

A approximation parallel algorithm of deep learning

Experiment result and future work

What is deep learning

Deep learning is a class of machine learning algorithms that:

use a cascade of many layers of nonlinear processing units

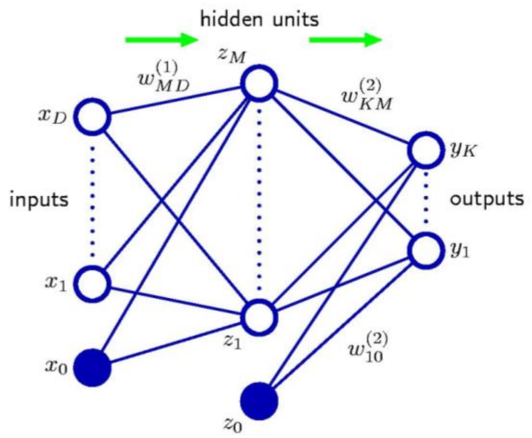
for feature extraction and transformation.

Deep learning is very popular

Deep learning is very slow

A easy sample of deep learning

A Neural Network



Can be viewed as a generalization of linear models

$$y_k(\boldsymbol{x}, \boldsymbol{w}) = \sigma\left(\sum_{j=1}^{M} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right)$$

A instruction for google research’s work

Draw a TensorFlow graph

Find the dependency

Add node to change dependency

Controlling Data Communication and Memory Usage

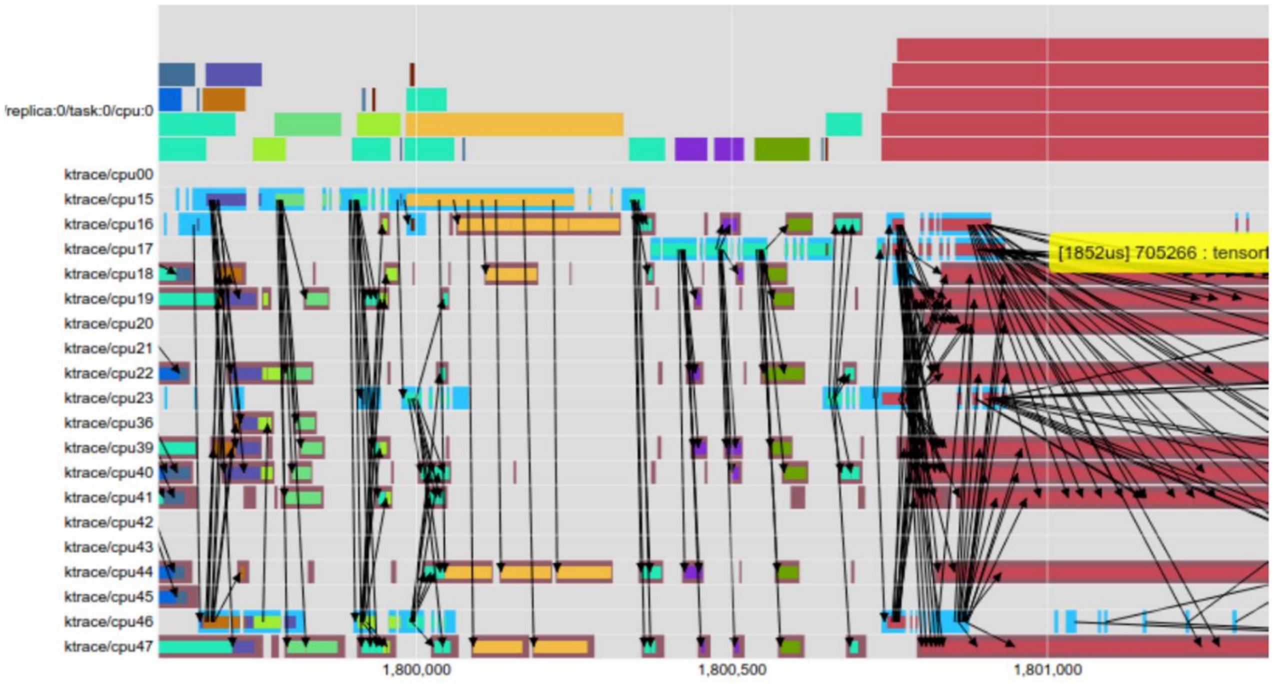


Figure 12: EEG visualization of multi-threaded CPU operations (x-axis is time in μs).

content

A brief summary of deep learning and google research's work

A approximation parallel algorithm of deep learning

Experiment result and future work

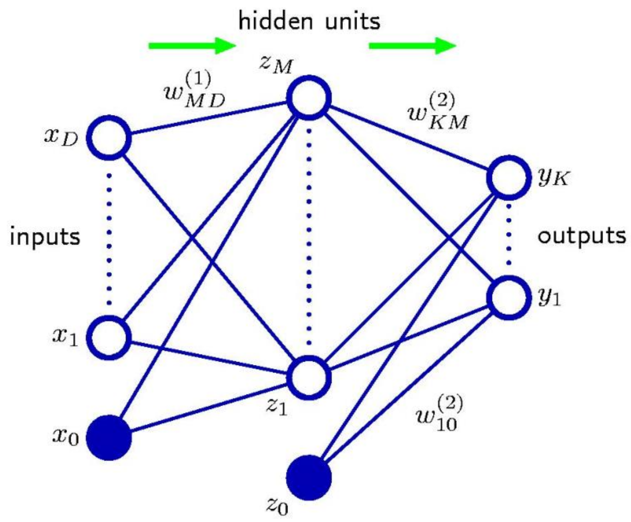
What I want to do at first

Use gradient decent to update the parameter

Try to update a parameter more efficiently

--------by update parameter at same time

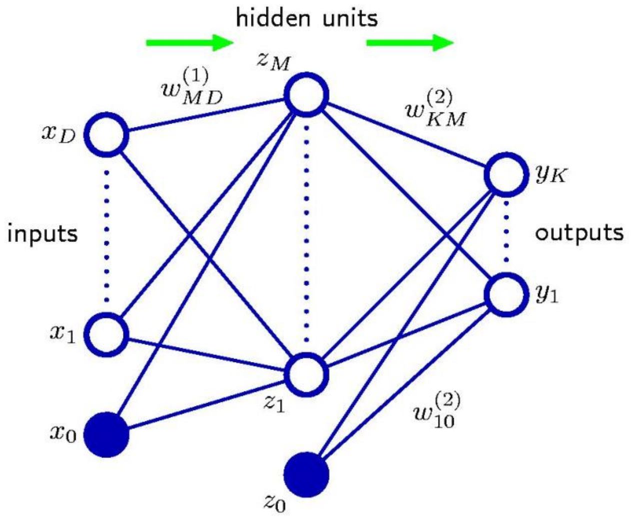
--------by sorting the data in a special way



Why Hard to implement

There are too much dependency!

You can not training one layer's weight without other layers



$$\mathrm{w}^{(\tau+1)} = \mathrm{w}^{(\tau)} - \eta \nabla E(\mathrm{w}^{(\tau)})$$

$$E(\boldsymbol{w}) = -\sum_{\mathrm{n}=1}^{\mathrm{N}} \sum_{\mathrm{k}=1}^{\mathrm{K}} \{t_{nk} \ln y_{nk} + (1 - t_{nk}) \ln(1 - y_{nk})\}$$

where y_{nk} denotes $y_k(\boldsymbol{x}_{\mathrm{n}}, \boldsymbol{w})$

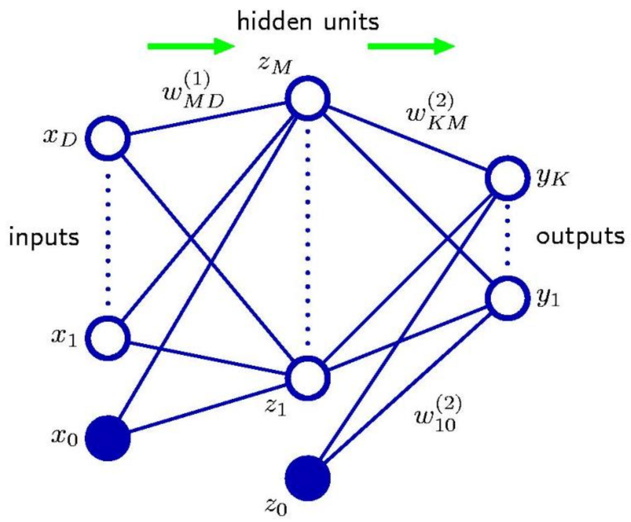
Solution: A approximation algorithm

Try to update the weight later in each layer

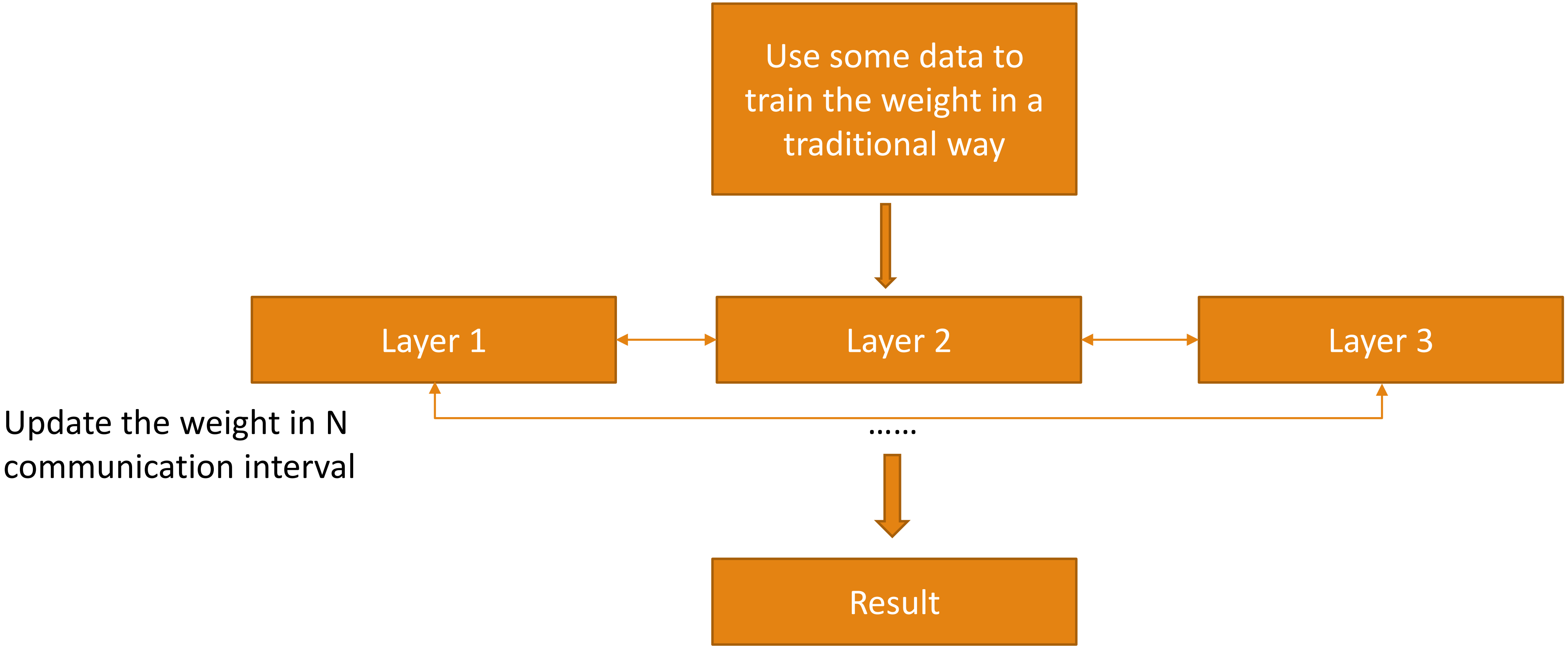
Reduce the relevance between different layers

Let each layer to update it's weight by itself

After training for a period communicate with other layer and update the global weight



approximation algorithm



content

A brief summary of deep learning and google research's work

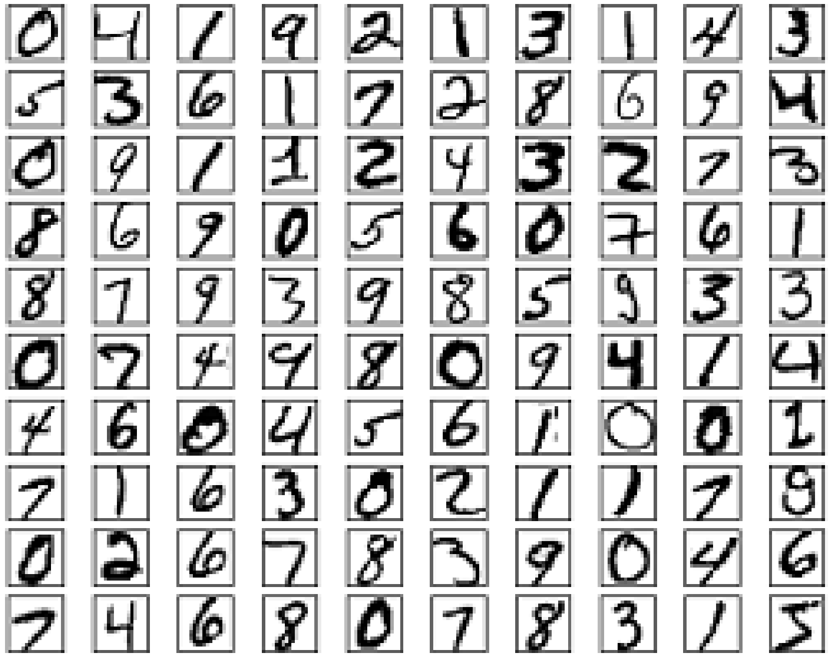
A approximation parallel algorithm of deep learning

Experiment result and future work

Experiment-------- MNIST

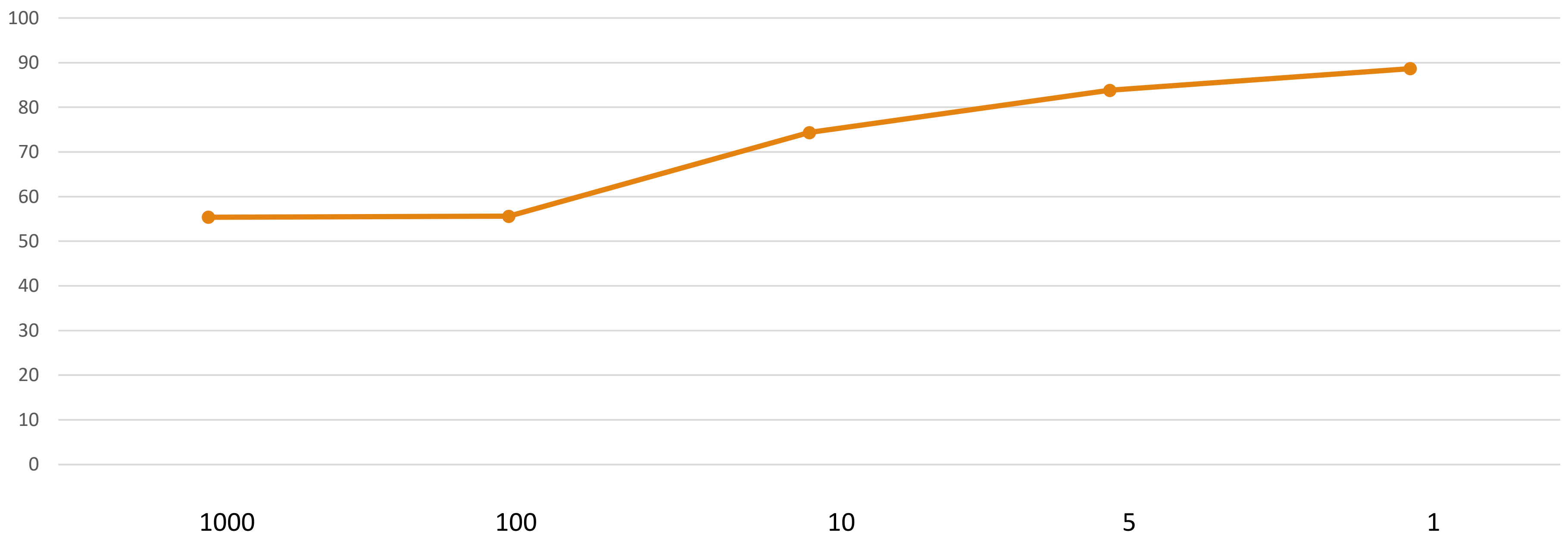
The MNIST database is a large database of handwritten digits that is commonly used for training various image processing systems

The MNIST database contains 60,000 training images and 10,000 testing images



Result of different communication interval

comm interval	1000	100	10	5	1
accuracy	55.39	55.61	74.35	83.83	88.68



Speed up

Former time: data size*training time per piece of data

New algorithm's time: $\frac{\text{Former time}}{\text{the layer of the model}} + \frac{\text{data size}}{\text{comm interval}} * \text{communication time}$

Conclusion

Approximation algorithm can not do a good job when communication interval is really big

When communication interval is small the algorithm could be slow

When there is only one or two layer the speed up is not very obvious, but when it come to more layers speed up will be much more obviously

Future work

Try to implement on more layers

Try to use more core to implement it

Reference

Yoshua Bengio. Learn Deep Architecture for AI. Foundations and Trends in Machine Learning, 2009

Jeffrey Dean, Greg S. Corrado, etc. Large Scale Distributed Deep Networks. NIPS, 2012

TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems (Preliminary White Paper, November 9, 2015)

Thank!