Discussion on the Project of Analyzing Running Time of Stochastic Gradient Methods
Some General Comments

- You should analyze your results
- Some just present tables or figures, but we want more insight
The Analysis I

- You should identify the major operations
- Then check percentage of these operations
- From the results you draw some conclusions
- Let’s consider a good example (from P2 of Davis Cho’s report in the UCLA 2019 course)
- I will show good examples this year later
The Analysis II

Forward

Eq. 1: \( \phi_{pad}(Z^m_{i}) \)

Eq. 2: \( W^m \phi(\cdot) \)

Eq. 3: \( Z^{m+1}_{i} = \text{mat}(P^m_{\text{pool}} \text{vec}(\sigma(S^m_{i}))) \)

As can be seen, the vTP function from the backward process takes a majority of the run time of the whole program, as shown in figure 1c. The vTP function is used in both the first equation for pooling as well as the third equation mentioned in the slides. The next most time consuming portion would be the forward feeding equation 2, denoted as F Eq.2 in Figure 1c.

Given that the vTP function is a vector matrix multiplication function, indeed the prediction that the next most costly function is the bottleneck is true in Matlab optimizes matrix matrix operations well.
A typical analysis is as follows. In the forward procedure,

\[
W^m \text{mat}(P^m P_{\text{pad}} \text{vec}(Z^{m,i})) = W^m \phi(\text{pad}(Z^{m,i}))
\]

\[
\phi(\text{pad}(Z^{m,i})) : \mathcal{O}(l \times h^m h^m d^m a^m_{\text{conv}} b^m_{\text{conv}})
\]

\[
W^m \phi(\cdot) : \mathcal{O}(l \times h^m h^m d^m d^{m+1} a^m_{\text{conv}} b^m_{\text{conv}})
\]

\[
Z^{m+1,i} = \text{mat}(P_{\text{pool}}^m \text{vec}(\sigma(S^{m,i})))
\]

\[
\mathcal{O}(l \times h^m h^m d^{m+1} a^m_{\text{conv}} b^m_{\text{conv}})
\]
The Analysis IV

- From this complexity analysis matrix-matrix products are the bottleneck
- Roughly,
  
  \[ d^m \] times

  more than others at layer \( m \)
- However, we see that matrix-matrix products take less than half (feedforward part)
- Thus optimized BLAS is very effective
- It is easier to optimize the computationally heavy part
But this also means that probably there is still room to improve other operations such as

$$\phi(\text{pad}(Z^{m,i}))$$
Why Checking Main Steps is Important

- We see that some simply list the percentage of each function
  - feedforward $??\%$
  - max pooling $??\%$
- But the problem is that max pooling is part of feedforward
- Thus you cannot say much from such a list of results
By default Tensorflow uses single
Thus we should use single in MATLAB too
But some may forget to use the option -f type to specify the use of single
Before a rough answer of this issue, let’s discuss some differences between Tensorflow and our MATLAB-based implementation

- Automatic differentiation
- Computational graphs
Automatic differentiation I

We will discuss this in regular lectures
Let’s borrow a description from https://deepnotes.io/tensorflow “Tensorflow approaches series of computations as a flow of data through a graph with nodes being computation units and edges being flow of Tensors (multidimensional arrays).”

This means that we must build a graph first before the execution

This is a bit unnatural

For example, to do a matrix product
C = A*B;

we cannot just write the above statement like in MATLAB.

We need two steps

- But using a graph does have some advantages
- One is the effective parallel computation
- Operations that are independent to each other (e.g., they need different input data) can be conducted in parallel
- Therefore, operations can be scheduled in a more efficient manner.
In contrast, our MATLAB code is a procedural setting.

For example, in the feed forward process for function evaluation we have

\[ S^{m,i} = W^m \phi(\text{pad}(Z^{m,i}))_{h^m h^m d^m \times a^m_{\text{conv}} b^m_{\text{conv}}} + b^m 1_T a^m_{\text{conv}} b^m_{\text{conv}}, i = 1, \ldots, l \]

and

\[ Z^{m+1,i} = \text{mat}(P^{m,i}_{\text{pool}} \text{vec}(\sigma(S^{m,i})))_{d^{m+1} \times a^{m+1} b^{m+1}}, i = 1, \ldots, l \]

Remember that this is over all data (in a batch).
Thus if a graph has been constructed, easily (1)-(2) can be done in parallel.
Now let’s go back to this issue
We think there are some possible reasons
Some MATLAB operations are not efficiently implemented
You have seen that index manipulation is time consuming
We will try to make improvements in the next project
Tensorflow’s setting by computational graph leads to better overall optimization?
Tensorflow may have used some optimized packages dedicated to neural networks.

For example, they may use

Intel MKL-DNN:

https://github.com/intel/mkl-dnn
Please have both ID and name on the first page

You should use the latex template

I have no choice but to give you 0 point as we have said that many times