Implementation Tool

Arthur J. Redfern
axr180074@utdallas.edu
Oct 24, 2018

0 Outline

1 Logistics

Assigned:	Oct 24
Due:	Nov 14

Submit

1. PDF of spreadsheets as described in the Results Display section for each of the networks in the Network Specification section

2. Code

2 Project Summary

Build a performance prediction tool

Performance == execution time as used here (not accuracy)

Input

- Network specification from high level library
- Parameterized hardware model

Output

- Performance prediction
  - Per layer
  - Total network

3 Network Specification

Select a high level library (TensorFlow, Pytorch, Caffe, …) and associated static graph format

- All of these libraries create a graph describing a network
  - An internal representation or an exchange format for saving / loading models
  - PyTorch pre version 1.0 may have some difficulties with respect to this
- You're going to need to work with this graph for determining layer types and parameters, input, filter and output tensor sizes, …
- It's ok to ask a classmate for help with this part of the project
  - But after getting help make sure you clearly understand how to work with the graph yourself

Find the following models in a format compatible with the selected high level library; a list of references is provided at the end of these notes with helpful places to look

- VGG 19
- MobileNet V1
- ResNet 50
- Inception V3 or V4

These models in the high level library's static graph format will serve as inputs to your performance prediction tool

- We only need the forward pass / evaluation versions
  - Not the training versions with backwards passes
- It's ok to ask a classmate for help with this part of the project
  - But after getting help make sure you clearly understand where / how to find models yourself

In the event that you're having a difficult time getting a static graph in a format that you can programmatically work with, you're allowed to define your own graph format that encompasses all common network structures and operations

A downside of this is that you will have to also create your own graphs of these models
This is a lot of work and not easily scalable to new models so only do this as a last resort
To emphasize, this really should be a last resort

4 Hardware Model

Define your own parameterized hardware model that captures key items needed for performance prediction

Can be a text file your program uses or a list of "defines" at the top of the program file

The following is a list of parameters you need with example values in parenthesis ()

External – internal memory bandwidth in bytes / second (10 GB / second)
Internal memory size in bytes (16 MB)

Matrix multiplication primitive size M, N, K in BLAS notation (32, 32, 32)
Number of matrix multiplication primitives operating in parallel (1)
Number of matrix multiplication primitive completions per second (1e9 / 32)

Vector primitive size N x 1 (32)
Number of vector primitives operating in parallel (1)
Number of vector primitive completions per second (1e9)

These parameters will serve as inputs to your performance prediction tool

5 Performance Prediction

Note that this is very approximate 1st order performance prediction; but it's ok from the perspective of getting a general idea of how a network will perform on hardware and what types of tradeoffs would be beneficial to improve performance

With respect to the network design
With respect to the hardware specification

Determine a memory location for every tensor based on tensor size and internal memory size

- Assume 8 bit elements instead of the 32 bit elements defined in the models
- Assign any feature map that can fit in internal memory to internal memory
- Assign all other feature maps to external memory
- Assign all filter coefficients to external memory
- While not fully optimal this is not far off

Determine the data movement time based on tensor size and data movement parameters

- Input tensor that needs to move from external memory to internal memory
- Filter coefficient tensor that needs to move from external memory to internal memory
- Output tensor that needs to move from internal memory to external memory
- While not fully correct this is not far off
  - E.g., DDR traffic, alignment, transfer lengths and delays are not accounted for; neither the weights or feature maps fitting fully on device are and associated extra data movement is not accounted for

Determine the compute time for every operation

- CNN style convolution based on hardware matrix compute parameters
- Bias addition and ReLU for free
- Pooling based on vector compute parameters
- All other operations based on vector compute parameter speed
- While not fully correct this is not far off
  - CNN style 2D convolution dominates compute so be as accurate as possible for that operation
  - Possibilities of partial layer overlap are not accounted for

6 Results Display

For every operator (node) create a row for a spreadsheet with the following fields

- Operator name

- Input feature map location (external or internal memory)
- Input feature map up sampling ratio
- Number of input feature maps
- Number of input feature map rows
- Number of input feature map cols
- Number of input feature map bytes

Output feature map location (external or internal memory)
Output feature map down sampling ratio
Number of output feature maps
Number of output feature map rows
Number of output feature map cols
Number of output feature map bytes

Filter coefficient location (external memory)
Filter up sampling ratio
Filter grouping
Number of filter rows
Number of filter cols
Number of filter bytes

Input feature map data movement time
Filter coefficient data movement time
Output feature map data movement time
Total data movement time for the layer

Matrix compute time
Vector compute time
Total compute time for the layer

Serial total data movement and compute time for the layer
Total data movement time for the layer + total compute time for the layer
Parallel total data movement and compute time for the layer
Max(total data movement time for the layer, total compute time for the layer)

Create a total row for the whole network for the following fields from above

Input feature map data movement time
Filter coefficient data movement time
Output feature map data movement time
Total data movement time for the network

Matrix compute time
Vector compute time
Total compute time for the network

Sum of serial total data movement and compute time for every layer
Sum of parallel total data movement and compute time for every layer

Choose nice units for data, compute and time such that the formatting looks good, conveys a meaningful level of precision and everything is aligned

7 Extra

Doing the detailed work of performance prediction makes you a better network designer

- You better understand the consequences of CNN style 2D convolution parameters
- You better understand the importance of fitting feature maps on device
- You better understand the importance of minimizing filter coefficient memory
- You better understand that compute dominates early in the network
- You better understand that memory becomes large late in the network
- …

If you want to do more in this area, here are some things to consider

- Report the receptive field size on a per operator basis
    - While not part of performance it's a useful quantity to keep track of when designing networks and thinking about classification accuracy for different size objects in the input image
- Auto generate networks in the chosen library format based on standard building blocks
    - ResNet, ResNeXt, DenseNet, …
    - All of these are based on a tail body head design with the body composed of replicated building blocks
    - Create a generator that parameterizes the repetition of these building blocks and perhaps some other portion of their parameters (e.g., number of channels, grouping, …)
- Modify the graph to merge batch norms
    - These aren't needed for testing and can often be absorbed into neighboring convolutional layers
- Add batching of inputs as a parameter
    - Need to rethink processing
    - When to process samples within the batch serially and when to process together
- Add ability to sweep performance prediction over common variables
    - Input size, data movement bandwidth, internal memory size, compute size and speed, …
- In the data movement time take into account cases where neither the whole input feature map tensor or whole filter tensor for a layer will fit on device internal memory at once
    - More data movement is needed for this case

8 References

Netscope cnn analyzer

https://dgschwend.github.io/netscope/quickstart.html

TensorFlow models

https://github.com/tensorflow/models/tree/master/research/slim
https://github.com/tensorflow/models/tree/master/official

PyTorch models

https://pytorch.org/docs/stable/torchvision/models.html
https://github.com/Cadene/pretrained-models.pytorch
https://github.com/marvis/pytorch-mobilenet

Additional model links

https://modelzoo.co

Additional TensorFlow information

https://www.tensorflow.org/guide/graphs
https://www.tensorflow.org/guide/saved_model
http://web.stanford.edu/class/cs224n/lectures/lecture6.pdf
http://web.stanford.edu/class/cs224n/readings/tensorflow_tutorial_code.zip
https://www.youtube.com/watch?v=PicxU81owCs#t=3m16s
http://web.stanford.edu/class/cs224s/lectures/Tensorflow-tutorial.pdf
https://github.com/pbhatnagar3/cs224s-tensorflow-tutorial

Additional PyTorch information

https://cs230-stanford.github.io/pytorch-getting-started.html
http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture08.pdf
https://pytorch.org/tutorials/
https://github.com/jcjohnson/pytorch-examples
https://github.com/ritchieng/the-incredible-pytorch