Project 02 – Networks

Arthur J. Redfern
arthur.redfern@utdallas.edu
Mar 30, 2020

---

1 Logistics

- Due to our class migrating to an online format for the remainder of the semester, the previously planned Test 2 is being replaced with Project 2, assigned Mar 30, 2020 and **due Apr 10, 2020**
- This is an individual project, no help from others is allowed
- The use of any and all online resources is allowed

2 Goals

- The primary goal of this project is to test a portion of your network design, training and implementation knowledge
- A secondary goal of this project is to improve your research paper writing skills

3 Setup

- You built a time machine and went back to late 2012, just after AlexNet won the ILSVRC, but before any subsequent CNN papers appeared
- You still have all of your knowledge with respect to 2020 CNN design, training and implementation methods and want to share this knowledge with the 2012 world; however, you don't want anyone to know that you're from the future and got here via time machine

4 Project

- **Design:** So you're going to write a research paper as if you're the inventor of a new to 2012 network design
    - In your own words introduce a MobileNet sequential design, a ResNet residual design, an Inception parallel design or a DenseNet dense design, … and provide some motivation / explanation of the design

  - Pick the 1 network structure you like the most or are most interested in sharing with the research community in 2012, others network structures can wait until their discovery sometime in the future
  - Consider parameterizing the design you introduce to create a family of networks
  - Modifications to the known to 2020 network structure are allowed; for example, if you'd like to show a slightly more regular Inception style parallel structure and not include the additional training heads, that's ok
  - SE and other complementary modifications are optional
  - Don't forget to describe your head (decoder) design
- Training: Introducing a fancy new network design doesn't do any good if you can't train it, so you're also going to share your 2020 knowledge network training strategy, again, pretending to be the inventor of any new to 2012 methods that you use to improve convergence and generalization
  - Possibilities include weight initialization methods, batch randomization and size selection, data augmentation, residual connections, batch norm, weight regularization terms, weight update method, … remember to provide some motivation / explanation of any new to 2012 methods that you use
  - Include training data loss and validation data accuracy curves
  - Note that this means you need to implement your network and training methods in TensorFlow, PyTorch, … and train it in Colab, AWS, … saving the required data
  - Feel free to make use of the code on the class GitHub page as a starting point
- Implementation: And finally, you're going to predict the inference performance of your network on an architecture that consists of an infinite external memory, 1 GB/s DDR bus for data movement between external memory and internal memory, 1 MB of internal memory, 1 TFLOPS of matrix compute and 10 GFLOPS of generic host compute
  - For each layer determine a location for the input and output feature maps in that layer; any feature maps that fully fit in internal memory assign to internal memory and assign all other feature maps to external memory
  - For each layer assign all filter coefficients to external memory
  - Determine a data movement time for each layer as the sum of
    - The time it takes to move input feature maps from external memory to internal memory (if necessary)
    - The time it takes to move filter coefficients from external memory to internal memory
    - The time it takes to move output feature maps from internal memory to external memory (if necessary)
  - Determine a compute time for each layer
    - CNN style 2D convolution and matrix multiplication as 2x the number of MACs in the operator / 1 TFLOPS (number of ops divided by ops per second = seconds)
    - All other operations as the number of ops in the operator / 10 GFLOPS (number of ops divided by ops per second = seconds)
  - Sum the data movement and compute times for all layers to get a predicted performance time to process a single input; mention this is a 1st order

approximation and many architectures can improve on this by parallelizing data movement and compute, …
    - This can be done on a spreadsheet and ~ copied into the paper

5 Practicalities

- Keeping with the spirit of 2012, the limited widespread availability of big compute and the limited amount of time you have for this project (comically, considering the time machine that the project is predicated on)
    - You can do all of the above on CIFAR-10
    - But if you'd like to use a larger dataset, that's also fine
    - Irrespective of the data set you choose you should optimize the network structure specifically for the input image size of the data set you select and also mention how the structure would be modified for the input image size used in other data sets (e.g., ImageNet)
    - Note that the input image size you choose also affects feature map placement for the implementation which affects the data movement time which affects the predicted performance

6 What To Turn In

- A pdf of your paper in the provided template format filled in with the above content
- A link to your code on either Colab or GitHub; include this somewhere in the paper
- It's ok to migrate the template to the editor of your choosing if you prefer something other than MS Word; irrespective of editor you choose, generate and upload a pdf
- Use the created eLearning Project 2 for turning in the paper, just like homework assignments