Project 02 – Networks

Arthur J. Redfern
arthur.redfern@utdallas.edu
Oct 12, 2020

---

1 Logistics

- Assigned Oct 12, 2020 and due Oct 26, 2020
- This is an individual project, no help from others is allowed
- The use of any and all online resources is allowed
- All code should be written by you – do not simply copy code from others

2 Goals

- The network section of the course built on the theory introduced in the math section and looked at the practice of the design, training and implementation of xNNs; specifically
    - An encoder (stem and body) decoder (head) approach to the design of CNNs for image classification based on common stem and head structures and serial, parallel, dense and residual body building blocks
    - A comprehensive look at training methods for improving convergence and improving generalization via regularization
    - A practical look at implementation including methods for estimating performance and methods for improving it via network, software and hardware design
- In this project you will use the above techniques to design, train and estimate the complexity of a CNN for image classification

3 Project

3.1 Image Classification (Max Grade 100/100)

Use PyTorch to design, train and implement a network for the classification of the images in the modified variant of ImageNet provided in the Data directory of the class GitHub page. The data set was created by down sampling the original ImageNet images such that their short side is 64 pixels (the other side is >= 64 pixels) and only 100 of the original 1000 classes were kept. Template code is provided for data download and formatting appropriate for PyTorch. Start from that template.

Design a RegNetX-200MF network modified for the smaller image size as follows:

- Set stride = 1 (instead of stride = 2) in the stem
- Replace the first stride = 2 down sampling building block in the original network by a stride = 1 normal building block
- The fully connected layer in the decoder outputs 100 classes instead of 1000 classes
- All of the other blocks in RegNetX-200MF stay the same

The original RegNetX-200MF takes in 3x224x224 input images and generates Nx7x7 feature maps before the decoder, this modified RegNetX-200MF will take in 3x56x56 input images (cropped from the provided data set) and generate Nx7x7 feature maps before the decoder.

Train the network using the training data to achieve the highest accuracy possible on the testing data. For reference, each epoch took ~ 112 s on Sep 27, 2020 using a free GPU runtime in Google Colab.

Implementation complexity for this network is strongly determined by the CNN style 2D convolution MACs and parameters. Calculate the number of MACs and parameters for each CNN style 2D convolution operation (ignore the bias terms and nonlinearities) and the sum for the whole network. Assume that the input to the network is a 3 x 56 x 56 input image.

3.2 Impress Me (Max Grade ???)

If you do something extra beyond what is required above, maybe you can earn some extra points. As a general rule, the more impressed I am by what you do and the quality by which you do it, the more points you can get. If everyone does the same extra thing, I'll be less impressed by it.

4 What To Turn In Via eLearning

- Upload individual files, not a zip file
- Don't write any comments in eLearning
- A Python file I can cut and paste into Google Colab, run and reproduce your results
    - For the required portion of the project call this file cnn.py
    - For the extra / impress me portion of things call this file extra.py (optional)
- A pdf file that describes your submission and is structured as follows
    - Call this file cnn.pdf
        - This file should be short (I don't want any unnecessary text)
        - This file should clearly describe the below items in the below format
    - Section 1 – Design
        - Include 3 figures (either drawn by hand or in a drawing program)

        - The network structure
        - The network standard building block
        - The network down sampling building block
    - Include 1 table
        - Describes the parameters (e.g., channels, groups, repeats, …) to create RegNetX-200MF from the generic network structure, standard building block and down sampling building block
- Section 2 – Training
    - Include 1 table
        - All hyper parameters and associated values
    - Include 2 plots
        - Training data loss vs epoch
        - Testing data accuracy vs epoch
    - List the final accuracy
- Section 3 – Implementation
    - Include 1 table
        - A list of each CNN style 2D convolution operator and associated the number of MACs and parameters; at the bottom of the table put a sum for the whole network
- Section 4 – Extra (optional)
    - Clearly and concisely describe anything extra you did