25_26_March_Update

1. Gave a successful sample run over C++ implementation of custom convolution. The error metrics and run time for inference over 1 image is given below:

```
Time taken for iteration:1098 in batch:0 is:10
Time taken for iteration:1099 in batch:0 is:11
Time taken for iteration:1100 in batch:0 is:10
Time taken for iteration:1101 in batch:0 is:11
Time taken for iteration:1102 in batch:0 is:10
Time taken for iteration:1103 in batch:0 is:10
100%|
                                                    100%|
                    | 1/1 [3:19:57<00:00, 11997.94s/it]
       a1,         a2,         a3,        rel,        rms,     log_10
   0.5158,     0.8451,     1.0000,     0.2964,     0.3561,     0.1087
```

   The error metrics are quite comparable to the state of the art, and the metrics - **a3**, **rms** are better than state of the art in this case. The code for c++ extension implementation has been updatedin the github repo under ‘cudatry’ subdirectory.

2. Currently I have given an inference run for **decoder.conv2** layer over 5 images. The results for the run are shown below:
   (a) Approx model

```
Time taken for iteration:1098 in batch:4 is:10
Time taken for iteration:1099 in batch:4 is:10
Time taken for iteration:1100 in batch:4 is:10
Time taken for iteration:1101 in batch:4 is:10
Time taken for iteration:1102 in batch:4 is:10
Time taken for iteration:1103 in batch:4 is:10
100%|                                                  | 1/1 [15:52:52<00:00, 57172.21s/it]
       a1,         a2,         a3,        rel,        rms,     log_10
   0.4931,     0.8112,     0.9050,     0.2562,     0.6850,     0.1267

Test time 57172.62291789055 s
```

   (b)Accurate model

```
D/customConv_MBM/for_denseNet_161/Pytorch_for_161$ python evaluate_py.py
Loading test data...Test data loaded.

Testing...
100%|                                                  | 1/1 [00:11<00:00, 11.81s/it]
       a1,         a2,         a3,        rel,        rms,     log_10
   0.6026,     0.8197,     0.8884,     0.2543,     0.7175,     0.1322

Test time 11.873706102371216 s
```

   As it can be observed, the metrics **a3**, **rms** & **log_10** are better than the accurate model and the **a1**, **a2**, **rel** metrics differ from the accurate model by -27%, -1.03%

and +0.75% respectively.

3. I have been looking into alternate convolution methods to speed up the inference process.

   One of the reasons why the pytorch conv2d implementation is faster is because it switches and selects between various algorithms such as Strassen's matrix multiplication, Winograd convolution, switching input and kernel to Fourier domain via FFT; perform dot product; and IFFT of result... etc based on the input and kernel size in highly optimized libraries.

   I looked into the prospect of using FFT convolution operation, but various sources suggest to use it only in case of large image and kernel sizes. And furthermore, it might also have an impact on the final error metrics along with the custom MBM multiplication.

   Any suggestions from your side regarding speed up would be really helpful based on your experience with other CNN models.

4. The long inference time for the approx model is also because of the large number of feature maps (channels) that are propagated through the model.

   For instance when we consider the decoder.conv2 layer of the model ,we can see the number of operations = (**out_channel**) * (**input_channel*****kh*****kw**) * (**in_h** * **in_w**) = (1104) * (2208*1*1) * (15*20) ~ 731M operations (lookups in our case)

   When compared to the other network architectures used for image classification, such as MobileNetv2, the number of operations per layer for DenseNet-161's decoder.conv2 is quite large. (roughly x2 - x50 times when compared with different conv layers of MobileNetv2).

   The MobileNetv2 architecture is updated here:
   https://docs.google.com/spreadsheets/d/1tmXCuR8P1yGrYK8_bC07wBkrz_x6DhPl-a2OtFICZnw/edit?usp=sharing

   Hence, it is expected to take more time for inference when compared with other image classification CNNs.

5. On this note, considering inference over 1 image takes ~3.5 hours => for 10 images the inference will take ~35 hours. Since there are 10 different conv layers

in the decoder, the time taken to run inference over all layers will take ~350 hrs. If run parallely on my local machine, TU Dresden's server and Google Colab, we might be able to bring down the inference time by some factor. (provided the Google Colab runs without any disconnections). So there will be a minimum requirement of 1 week to be able to run inference on all layers for 10 images.

My other suggestion would be to run inference over 5 images on all the layers, so as to possibly reduce the number of days for inference. In that case, the inference over 1 layer is already done, (as specified above) and I can move forward with the other layers in the decoder.

Kindly let me know about your thoughts on this.