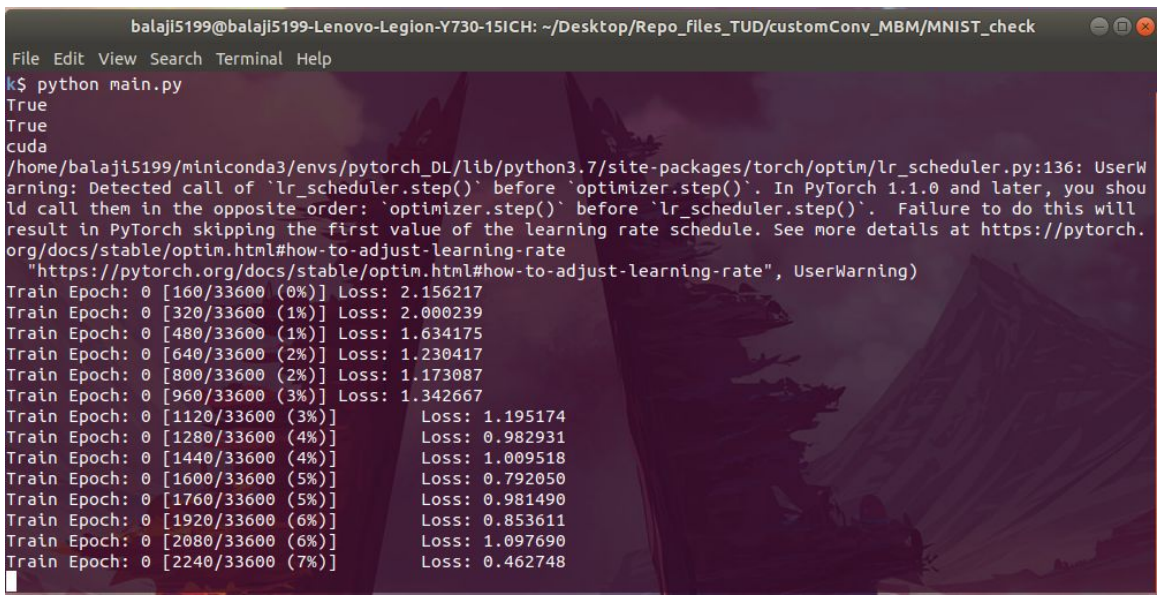
UPDATE - 3: CUSTOM CONV LAYER FOR MBM

1. **Custom conv layer:**
   - The backward() function required some changes, to work properly (was encountering dimension mismatch errors). Solved by making use of the convolution functions defined under torch.nn.grad ----> conv2d_input, conv2d_weight rather than doing matrix mul manually.

   - Tried training the model for 1 epoch over MNIST dataset, to check functionality. Image of epoch progress below:



```
balaji5199@balaji5199-Lenovo-Legion-Y730-15ICH: ~/Desktop/Repo_files_TUD/customConv_MBM/MNIST_check
File  Edit  View  Search  Terminal  Help
k$ python main.py
True
True
cuda
/home/balaji5199/miniconda3/envs/pytorch_DL/lib/python3.7/site-packages/torch/optim/lr_scheduler.py:136: UserW
arning: Detected call of `lr_scheduler.step()` before `optimizer.step()`. In PyTorch 1.1.0 and later, you shou
ld call them in the opposite order: `optimizer.step()` before `lr_scheduler.step()`.  Failure to do this will
result in PyTorch skipping the first value of the learning rate schedule. See more details at https://pytorch.
org/docs/stable/optim.html#how-to-adjust-learning-rate
  "https://pytorch.org/docs/stable/optim.html#how-to-adjust-learning-rate", UserWarning)
Train Epoch: 0 [160/33600 (0%)] Loss: 2.156217
Train Epoch: 0 [320/33600 (1%)] Loss: 2.000239
Train Epoch: 0 [480/33600 (1%)] Loss: 1.634175
Train Epoch: 0 [640/33600 (2%)] Loss: 1.230417
Train Epoch: 0 [800/33600 (2%)] Loss: 1.173087
Train Epoch: 0 [960/33600 (3%)] Loss: 1.342667
Train Epoch: 0 [1120/33600 (3%)]        Loss: 1.195174
Train Epoch: 0 [1280/33600 (4%)]        Loss: 0.982931
Train Epoch: 0 [1440/33600 (4%)]        Loss: 1.009518
Train Epoch: 0 [1600/33600 (5%)]        Loss: 0.792050
Train Epoch: 0 [1760/33600 (5%)]        Loss: 0.981490
Train Epoch: 0 [1920/33600 (6%)]        Loss: 0.853611
Train Epoch: 0 [2080/33600 (6%)]        Loss: 1.097690
Train Epoch: 0 [2240/33600 (7%)]        Loss: 0.462748
```

As observed, the loss is decreasing as expected, but to even complete **7%** of **1 epoch** took about **~4 hours** (while it should only take less than a minute to complete 1 epoch)… The model performance is too slow. (done with MBM.py just returning the product of the inputs)

*The **custom_conv2d.py** code has been updated into github repo.*

- Possible reasons are:
    1. *Torch.nn.grad.conv2d_input* and *Torch.nn.grad.conv2d_weight* functions might be the reason for bad performance… an alternative is to directly call from the cudnn counterparts directly.
    2. The code in forward() pass might not be parallelized properly… (check if further unfoldings are possible in conv operation)
    3. Problem with local hardware. Should try it out in Google Colab as well and cross verify.

Some useful links that I have used have been given below.[1, 2]

2. **Brief Working Detail:**
    - **Software/ packages details :-**
      python 3.7.9; pytorch 1.7.0; torchvision 0.8.1; torchsummary 1.5.1; cudatoolkit 10.2.89; And other common packages in their latest versions

    - **MBM implementation in python:-**
      The python implementation of MBM algo might potentially slow down the training process. (have to check it)... One possibility would be to implement it in C/C++ and make a python wrapper using **PyCLIF**.

    - **Github Repo:-** Constant update to the codes done here.
      https://github.com/bALAJi-aDItHYa/MBM_implementation.git

3. **Further Work:**
    - To implement the code on Google Colab to check for time taken on MNIST dataset.

    - To replace the convolutions in backward() with function calls to cudnn_conv2d_input and cudnn_conv2d_weight.

- To check if forward() has been parallelized properly - in the convolution part of the code.

- Implementation of MBM on python or C/C++ and make a python wrapper to boost performance. (depends on how MBM performs on MNIST dataset)

4. **References:**

[1] Understanding convolution in backpropagation:

1) https://medium.com/@pavisj/convolutions-and-backpropagations-46026a8f5d2c
2) https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199
3) https://www.youtube.com/watch?v=BvrWiL2fd0M

[2] Helpful for updating the backward() function & for cudnn calling:

1) https://github.com/pytorch/pytorch/blob/master/torch/nn/grad.py - for conv2d_input and conv2d_weight
2) https://discuss.pytorch.org/t/make-custom-conv2d-layer-efficient-wrt-speed-and-memory/70175 - efficiency of the functions
3) https://discuss.pytorch.org/t/implementing-a-custom-convolution-using-conv2d-input-and-conv2d-weight/18556
4) https://github.com/jordan-g/PyTorch-cuDNN-Convolution/blob/master/cudnn_convolution.cpp - pytorch cudnn convolution