UPDATE - 2: CUSTOM CONV LAYER FOR MBM

1. **Tensorflow/ Keras Disadvantages:**
    - Have to modify underlying C++ code/ kernel of python wrapper to implement own functionality
    - The right class to sub-class from :- **tf.nn.convolution**; to make the custom Conv2D layer
    - Tried following possibilities: 1) Sub-classing, 2) Lambda layer(inline function)
    - Problems with backpropagation also have to be handled separately
    - Implementation seemed a bit complex

2. **Switch over to PyTorch!**
    - It is simpler to implement custom convolution layer
    - Has somewhat better community blogs/ posts over the custom layer development topic

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

3. **Work Done & To be implemented:**
    - Adding custom operations to *autograd* (automatic differentiation package) by creating a class; subclassing from *torch.autograd.Function* - contains forward() and backward() functions as class methods **(Implemented)**

    - Adding a module and making use of the custom function by creating a class; subclassing from *nn.Module* - contains __init()__ and forward() functions as class methods **(Implemented)**

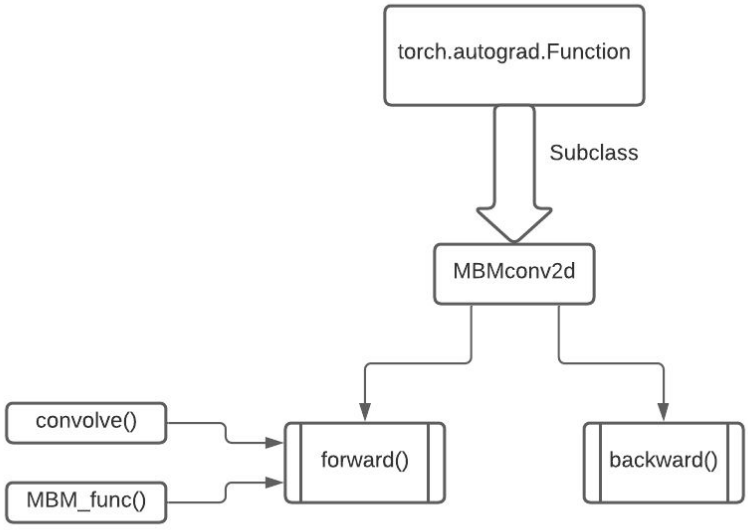
    - Write the MBM implementation in python. **(Not implemented)**

    - Integration of the code with the Single Image Depth Estimation (SIDE) project source codes. **(Not implemented)**

Some useful links that I referred to have been attached below.[2]
***custom_conv2d.py** - my implementation has been added to the dropbox*

4. **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

    - Custom function implementation - from *torch.autograd.Function*



forward(): This function is used to perform the necessary operations.
backward(): Function used to derive gradient formula for parameterized variables - weight/kernel, bias, etc

Code is parallelized using the *torch.nn.functional.unfold()* function.

- Custom Model implementation - from *nn.Model - containing forward() and __init__()*

Kernels/ weights are initialized in this class. They are also parametrized i.e gradients applied.

Some useful links for the implementation details have been attached.[3]

5. **Further Work:**
    - The process of training all the weights on the Encoder-Decoder structure as done in SIDE took a lot of time. Roughly ~2.5 hrs for training 1 epoch with batch size of 6 when trained on Google Colab.
      In order to test accuracy even faster, I'm thinking of implementing the custom-conv layer first on MNIST dataset followed by SIDE implementation. Kindly let me know if this is fine!

6. **References:**

[1] Custom conv2d implementation leads using PyTorch:

1) https://stackoverflow.com/questions/59149785/custom-conv2d-operation-pytorch
2) https://discuss.pytorch.org/t/custom-nn-conv2d/62068
3) https://discuss.pytorch.org/t/custom-a-new-convolution-layer-in-cnn/43682

[2] Custom module and function Pytorch documentation & other stuff:

1) https://pytorch.org/docs/stable/notes/extending.html
2) https://pytorch.org/docs/stable/generated/torch.nn.Unfold.html -Unfold

[3] Miscellaneous articles for functionality:

1) https://medium.com/apache-mxnet/multi-channel-convolutions-explained-with-ms-excel-9bbf8eb77108
2) https://blog.paperspace.com/pytorch-101-advanced/
3) https://discuss.pytorch.org/t/fold-unfold-to-get-patches/53419

REFERENCES USED IN SINGLE IMAGE DEPTH ESTIMATION

1) Koch, Tobias, et al. "Evaluation of CNN-based single-image depth estimation methods." Proceedings of the European Conference on Computer Vision (ECCV) . 2018.
2) Li, Qiaohong, et al. "Gradient-weighted structural similarity for image quality assessments." 2015 IEEE International Symposium on Circuits and Systems (ISCAS) . IEEE, 2015.
3) Hore, Alain, and Djemel Ziou. "Image quality metrics: PSNR vs. SSIM." 2010 20th International Conference on Pattern Recognition . IEEE, 2010.
4) Google Colab : https://colab.research.google.com/
5) Li, Zhengqi, et al. "Learning the Depths of Moving People by Watching Frozen People." Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition .2019.
6) Alhashim, Ibraheem, and Peter Wonka. "High Quality Monocular Depth Estimation via Transfer Learning." arXiv preprint arXiv:1812.11941 (2018).
7) Xue, Wufeng, et al. "Gradient magnitude similarity deviation: A highly efficient perceptual image quality index." IEEE Transactions on Image Processing 23.2 (2013): 684-695
8) Eigen, David, Christian Puhrsch, and Rob Fergus. "Depth map prediction from a single image using a multi-scale deep network." Advances in neural information processing systems . 2014.
9) Cordts, Marius, et al. "The cityscapes dataset for semantic urban scene understanding." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.