Google Summer of Code (GSoC 2021)

Learning Mass of Dark Matter Halo

Submitted By: Divyansha Ms

College: National Institute of Technology Silchar

25th March 2021

1 Task

Implement a regression algorithm to learn the mapping between lensing images and the lensing dark matter halo mass. The repository for all the code is available on Github

2 Dataset

The dataset consists of a grayscale image of size (150, 150) and the value of lensing dark matter halo mass.

3 Pre-processing of the data

The dataset has been split into training and testing with a ratio of 90:10. The training data consists of 22500 samples and testing data consists of 2500 samples. The range of images was normalized from 0 - 255 to 0 - 1. A custom normalization was added to the value of lensing dark matter halo mass. The value was divided by 400 in order to get its range from 0 - 1. This custom normalization was based on manual inspection of data.

4 Network Architecture Used

ResNet [1] is one of the most powerful deep neural networks which has achieved great performance results in the ILSVRC 2015 classification challenge [2]. ResNet has achieved excellent generalization performance on other recognition tasks and won the first place on ImageNet detection, ImageNet localization, COCO detection and COCO segmentation in ILSVRC and COCO 2015 competitions [3]. There are many variants of ResNet architecture i.e. same concept but with a different number of layers. ResNet 18 was used to perform regression. ResNet-18 is a convolutional neural network that is 18 layers deep. It is well known that ResNets are usually built for performing multi-class classification tasks. The last layer activation i.e. softmax was removed. Thus the network is now free to output any number. The network is built using PyTorch. Given below is the network summary of our model

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1         [-1, 64, 150, 150]             576
       BatchNorm2d-2         [-1, 64, 150, 150]             128
            Conv2d-3         [-1, 64, 150, 150]          36,864
       BatchNorm2d-4         [-1, 64, 150, 150]             128
            Conv2d-5         [-1, 64, 150, 150]          36,864
       BatchNorm2d-6         [-1, 64, 150, 150]             128
        BasicBlock-7         [-1, 64, 150, 150]               0
            Conv2d-8         [-1, 64, 150, 150]          36,864
       BatchNorm2d-9         [-1, 64, 150, 150]             128
           Conv2d-10         [-1, 64, 150, 150]          36,864
      BatchNorm2d-11         [-1, 64, 150, 150]             128
       BasicBlock-12         [-1, 64, 150, 150]               0
           Conv2d-13          [-1, 128, 75, 75]          73,728
      BatchNorm2d-14          [-1, 128, 75, 75]             256
           Conv2d-15          [-1, 128, 75, 75]         147,456
      BatchNorm2d-16          [-1, 128, 75, 75]             256
           Conv2d-17          [-1, 128, 75, 75]           8,192
      BatchNorm2d-18          [-1, 128, 75, 75]             256
       BasicBlock-19          [-1, 128, 75, 75]               0
           Conv2d-20          [-1, 128, 75, 75]         147,456
      BatchNorm2d-21          [-1, 128, 75, 75]             256
           Conv2d-22          [-1, 128, 75, 75]         147,456
      BatchNorm2d-23          [-1, 128, 75, 75]             256
       BasicBlock-24          [-1, 128, 75, 75]               0
           Conv2d-25          [-1, 256, 38, 38]         294,912
      BatchNorm2d-26          [-1, 256, 38, 38]             512
           Conv2d-27          [-1, 256, 38, 38]         589,824
      BatchNorm2d-28          [-1, 256, 38, 38]             512
           Conv2d-29          [-1, 256, 38, 38]          32,768
      BatchNorm2d-30          [-1, 256, 38, 38]             512
       BasicBlock-31          [-1, 256, 38, 38]               0
           Conv2d-32          [-1, 256, 38, 38]         589,824
      BatchNorm2d-33          [-1, 256, 38, 38]             512
           Conv2d-34          [-1, 256, 38, 38]         589,824
      BatchNorm2d-35          [-1, 256, 38, 38]             512
       BasicBlock-36          [-1, 256, 38, 38]               0
           Conv2d-37          [-1, 512, 19, 19]       1,179,648
      BatchNorm2d-38          [-1, 512, 19, 19]           1,024
```

```
            Conv2d-39          [-1, 512, 19, 19]       2,359,296
       BatchNorm2d-40          [-1, 512, 19, 19]           1,024
            Conv2d-41          [-1, 512, 19, 19]         131,072
       BatchNorm2d-42          [-1, 512, 19, 19]           1,024
        BasicBlock-43          [-1, 512, 19, 19]               0
            Conv2d-44          [-1, 512, 19, 19]       2,359,296
       BatchNorm2d-45          [-1, 512, 19, 19]           1,024
            Conv2d-46          [-1, 512, 19, 19]       2,359,296
       BatchNorm2d-47          [-1, 512, 19, 19]           1,024
        BasicBlock-48          [-1, 512, 19, 19]               0
            Linear-49                   [-1, 1]           8,193
================================================================
Total params: 11,175,873
Trainable params: 11,175,873
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.09
Forward/backward pass size (MB): 248.52
Params size (MB): 42.63
Estimated Total Size (MB): 291.24
----------------------------------------------------------------
```

Listing 1: ResNet-18 Network Summary

5 Network Training

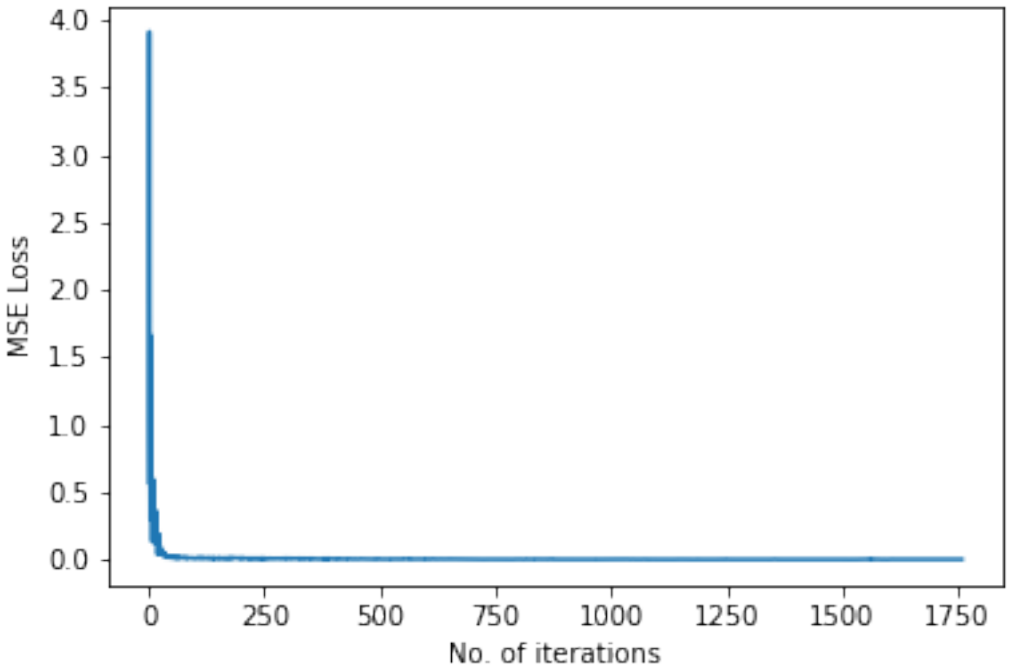

Figure 1: Loss

The network has been trained for 5 epochs with a learning rate of 0.00005. Adam [4] has been used to optimize the network with a weight decay of 0.00001. A batch

size of 64 has been used. As this is a regression task mean square loss has been used as a loss function. The network with minimum testing loss has been saved.

6 Results

We successfully achieved a testing loss of 0.0018 and a training loss of 0.0018. This shows that our model has successfully converged. Fig 1 shows the plot of training loss with the number of iterations.

References

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[3] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, pp. 740–755, Springer, 2014.

[4] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.