

---

## Non-local\_pytorch

- Implementation of **Non-local Neural Block**.

### Statement

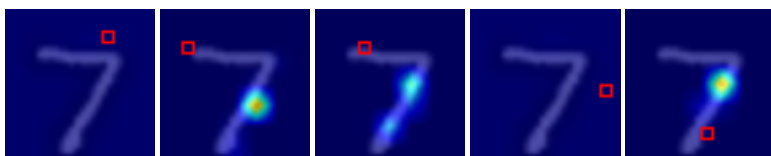
- You can find different kinds of non-local block in **lib/**.
- You can **visualize** the Non\_local Attention Map by following the **Running Steps** shown below.
- The code is tested on MNIST dataset. You can select the type of non-local block in **lib/network.py**.
- If there is something wrong in my code, please contact me, thanks!

### Environment

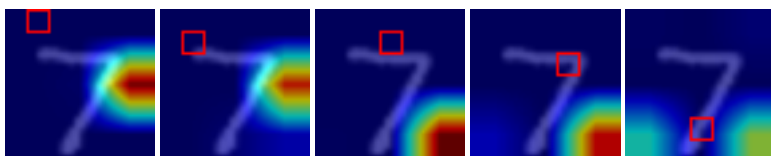
- python 3.7.7 (>=3.8)
- pytorch 1.4.0 (>=1.2.0; 2.0.0 works too)
- opencv 4.2.0.34 (others work too)

### Visualization

1. In the **first** Non-local Layer.



2. In the **second** Non-local Layer.



### Running Steps

1. Select the type of non-local block in **lib/network.py**.  
`from lib.non_local_concatenation  
import NONLocalBlock2D from lib.non_local_gaussian import`

---

```
NONLocalBlock2D from lib.non_local_embedded_gaussian import
NONLocalBlock2D from lib.non_local_dot_product import NONLocalBlock2D
```

2. Run **demo\_MNIST\_train.py** with one GPU or multi GPU to train the Network. Then the weights will be save in **weights/**. ““ `CUDA_VISIBLE_DEVICES=0,1 python demo_MNIST_train.py`  
`# Or train with Automatic Mixed Precision based on pytorch 1.6.0 CUDA_VISIBLE_DEVICES=0`  
`python demo_MNIST_AMP_train_with_single_gpu.py` ““
3. Run **nl\_map\_save.py** to save NL\_MAP of one test sample in **nl\_map\_vis**. `CUDA_VISIBLE_DEVICES=0,1 python nl_map_save.py`
4. Come into **nl\_map\_vis/** and run **nl\_map\_vis.py** to visualize the NL\_MAP. (tips: if the Non-local type you select is **non\_local\_concatenation** or **non\_local\_dot\_product** (without Softmax operation), you may need to normalize NL\_MAP in the visualize code) `python nl_map_save.py`

## Update Records

1. Figure out how to implement the **concatenation** type, and add the code to **lib/**.
2. Fix the bug in **lib/non\_local.py** (old version) when using multi-gpu. Someone shares the reason with me, and you can find it in here.
3. Fix the error of 3D pooling in **lib/non\_local.py** (old version). Appreciate **protein27** for pointing it out.
4. For convenience, I split the **lib/non\_local.py** into four python files, and move the old versions (**lib/non\_loca.py** and **lib/non\_local\_simple\_version.py**) into **lib/backup/**.
5. Modify the code to support pytorch 0.4.1, and move the code supporting pytorch 0.3.1 to **Non-Local\_pytorch\_0.3.1/**.
6. Test the code with pytorch 1.1.0 and it works.
7. Move the code supporting pytorch 0.4.1 and 1.1.0 to **Non-Local\_pytorch\_0.4.1\_to\_1.1.0/** (In fact, I think it can also support pytorch 1.2.0).
8. In order to visualize NL\_MAP, some code have been slightly modified. The code **nl\_map\_save.py** is added to save NL\_MAP (two Non-local Layer) of one test sample. The code **Non-local\_pytorch/nl\_map\_vis.py** is added to visualize NL\_MAP. Besieds, the code is support pytorch 1.2.0.
9. The code also works well in **pytorch 1.4.0**.

- 
10. The code also works well in **pytorch 1.6.0**. Add **demo\_MNIST\_AMP\_train\_with\_single\_gpu.py** with Automatic Mixed Precision Training (FP16), supported by **pytorch 1.6.0**. It can reduce GPU memory during training. What's more, if you use GPU 2080Ti (tensor cores), training speed can be increased. More details (such as how to train with multiple GPUs) can be found in [here](#)
  11. Verify that the code works well in **pytorch 1.7.0**.
  12. Verify that the code works well in **pytorch 1.8.1**.
  13. Verify that the code works well in **pytorch 1.9.0**.
  14. Verify that the code works well in **pytorch 1.10.1**.
  15. Verify that the code works well in **pytorch 1.11.0**.
  16. Verify that the code works well in **pytorch 1.12.0**.
  17. Remove redundant the code `net.train()` in training code files.
  18. Verify that the code works well in **pytorch 1.13.0 and 2.0.0**.

## Todo

- Experiments on Charades dataset.
- Experiments on COCO dataset.

## Related Repositories

1. **Non-local ResNet-50 TSM (Paper)** on Kinetics dataset. They report that their model achieves a good performance of **75.6% on Kinetics**, which is even higher than Non-local ResNet-50 I3D ([Here](#)).