一：数据集：直接使用lmdb格式Mnist数据集

二：自定义网络：先使用LeNet结构

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=5, num\_output=20 |
| Pool1 | kernel\_size=2, stride=2, MAX |
| Conv2 | kernel\_size=5, num\_output=50 |
| Pool2 | kernel\_size=2, stride=2, MAX |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| Fc2 | Num\_output=10 |
| loss |  |

三：配置

train\_net: "train.prototxt"

test\_net: "test.prototxt"

test\_iter: 100

test\_interval: 100

base\_lr: 0.00999999977648

display: 100

max\_iter: 1000

lr\_policy: "inv"

gamma: 9.99999974738e-05

power: 0.75

momentum: 0.899999976158

weight\_decay: 0.000500000023749

snapshot: 5000

snapshot\_prefix: "custom\_net"

solver\_mode: CPU

random\_seed: 831486

type: "SGD"

实验一： 默认配置作为基准

Test Accuracy

[ 0.1379 0.9121 0.9198 0.9656 0.9712 0.9707 0.9659 0.9786 0.9692 0.9794]

实验二： 增大base\_lr为0.05

[ 0.2009 0.9228 0.9357 0.9678 0.9706 0.9746 0.9785 0.9786 0.9813 0.9815]

发现效果变好了

实验三： 将lr\_policy改为fixed base\_lr为0.05

[ 0.2009 0.8626 0.9275 0.968 0.965 0.9686 0.9705 0.9699 0.982

0.9816]

实验四： 减小base\_lr为0.005

[ 0.1301 0.8975 0.9077 0.9521 0.9568 0.963 0.9559 0.9723 0.9622 0.9734]

实验五： 增大base\_lr为0.01

[ 0.1379 0.912 0.9192 0.9657 0.9714 0.9713 0.9652 0.9786 0.9672 0.9796]

实验六： 增大base\_lr为0.1

[ 0.2346 0.098 0.098 0.098 0.098 0.098 0.098 0.098 0.098

0.098 ]

发现出现无法收敛的情况。

实验7： 将lr\_policy改为inv， base\_lr 改为0.05, type改为AdaDelta

[ 0.1265 0.6567 0.7798 0.8166 0.8534 0.8666 0.8784 0.8864 0.8925 0.8979]

实验八：base\_lr改为0.1

[ 0.1308 0.7785 0.8487 0.8752 0.8934 0.9032 0.9089 0.9181 0.9196 0.9208]

实验九：base\_lr改为0.5

[ 0.1646 0.8767 0.9161 0.9201 0.9403 0.9499 0.955 0.9568 0.9643 0.9656]

实验10： type 改为 Adam , base\_lr改为0.05

[ 0.1304 0.1135 0.1135 0.1135 0.1135 0.1135 0.1032 0.1028 0.1028 0.1028]

实验十一： base\_lr改为0.005

[ 0.2355 0.9105 0.9236 0.9672 0.9629 0.9551 0.9698 0.9713 0.9706 0.972 ]

四： 更改网络结构 使用默认配置

实验一： Pooling 改为AVE

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=5, num\_output=20 |
| Pool1 | kernel\_size=2, stride=2, AVE |
| Conv2 | kernel\_size=5, num\_output=50 |
| Pool2 | kernel\_size=2, stride=2, AVE |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.1505 0.8928 0.8883 0.9417 0.9431 0.9516 0.9461 0.9538 0.9514 0.9601]

实验二： 用两个3×3的卷积代替一个5×5

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=3, num\_output=20 |
| Conv2 | kernel\_size=3, num\_output=20 |
| Pool1 | kernel\_size=2, stride=2, AVE |
| Conv3 | kernel\_size=3, num\_output=50 |
| Conv4 | kernel\_size=3, num\_output=50 |
| Pool2 | kernel\_size=2, stride=2, AVE |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.1465 0.9087 0.915 0.9499 0.9465 0.9541 0.9502 0.9631 0.9547 0.9654]

实验三：增加标准化层

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=3, num\_output=20 |
| BN |  |
| Conv2 | kernel\_size=3, num\_output=20 |
| BN |  |
| Pool1 | kernel\_size=2, stride=2, AVE |
| Conv3 | kernel\_size=3, num\_output=50 |
| BN |  |
| Conv4 | kernel\_size=3, num\_output=50 |
| BN |  |
| Pool2 | kernel\_size=2, stride=2, AVE |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.1986 0.908 0.9161 0.9422 0.9464 0.9499 0.9536 0.9581 0.9583 0.9627]

实验四：增加scale

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Conv2 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Pool1 | kernel\_size=2, stride=2, MAX |
| Conv3 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Conv4 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Pool2 | kernel\_size=2, stride=2, MAX |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.1838 0.9416 0.9318 0.9746 0.97 0.968 0.967 0.985 0.976

0.9825]

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Conv2 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Pool1 | kernel\_size=2, stride=2, AVE |
| Conv3 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Conv4 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Pool2 | kernel\_size=2, stride=2, AVE |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.1988 0.9082 0.9169 0.9418 0.9462 0.9505 0.9529 0.9582 0.9579 0.9622]

实验四：增加fc层

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Conv2 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Pool1 | kernel\_size=2, stride=2, MAX |
| Conv3 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Conv4 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Pool2 | kernel\_size=2, stride=2, MAX |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| FC | num\_output=100 |
| ReLU2 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.0941 0.9241 0.9221 0.9726 0.9633 0.9648 0.9718 0.9746 0.9784 0.9822]

实验五：无pooling层

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Conv2 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Conv3 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Conv4 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| FC | num\_output=100 |
| ReLU2 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.2351 0.9058 0.9096 0.9494 0.95 0.9606 0.9603 0.9632 0.9566 0.9617]

实验六：卷积代替Pooling

|  |  |
| --- | --- |
| name | detail |
| Conv1 | kernel\_size=3, num\_output=20 |
| BN+Scale |  |
| Conv2 | kernel\_size=3,stride=2 num\_output=20 |
| BN+Scale |  |
| Conv3 | kernel\_size=3, num\_output=50 |
| BN+Scale |  |
| Conv4 | kernel\_size=3,stride=2 num\_output=50 |
| BN+Scale |  |
| Fc1 | num\_output=500 |
| ReLU1 |  |
| FC | num\_output=100 |
| ReLU2 |  |
| Fc2 | Num\_output=10 |
| loss |  |

[ 0.0994 0.9089 0.915 0.9474 0.9529 0.9567 0.9568 0.9558 0.9579 0.9653]