# Baseline复现（CIFAR10）

## 无噪音

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| clean data | baseline\_model.py代码采用softmax+nll的计算loss方式 | baseline\_model.py代码直接采用crossentropy的loss计算方式 | simple model（没有那些mcsample和不确定性的部件） | 网上的resnet trainer函数 | meta原文中写的 |
| wrn28-10(40 epoch) | 94.48 | 94.75 | 94.59 | 94.53 | 95.6 |
| rn32(60 epoch) | 91.15 | 92.22 | 92.16 | 92.22 | 92.89 |

## Flip noise

20% of one class is fliped half half to two classes, 0.8 are left for each class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0.2 flip2 | b nll | b cross | s | rt | meta原文 |
| rn32(60 epoch) | 89.17 | 89.55 | 90.16 | 90.17 | 76.83 |

## Uniform noise

40% of one class is divided into 10 classes, 0.64 are left for each class

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 0.4 uniform | b nll | b cross | s | rt | meta原文 |
| wrn28-10(40 epoch) | 89.42 | 89.41 | 89.14 | 85.07 | 68.07 |

# 伪代码

1. data loader:
   1. 用corruption prob与corruption type计算corruption matrix C
   2. 对每一个dataset中的图片：
      1. 判断是否为参与训练的类，若是则按照corruption matrix更改label，不是则加到mask里面最后从数据中删除
   3. 包装为data loader
2. test loader直接包装为data loader
3. 网络结构的py文件：直接在他人的resnet最后一层（输出层）做更改，10类变为输出20个logit（前十个是logit后十个为aleatoric uncertainty prediction）
4. 训练函数：
   1. 一个phase的训练函数：形参为第0或1phase，epoch数量
      1. 每个epoch：
         1. 先对训练数据集执行relabel函数，相当于一个test函数但会把根据网络输出算出的不确定度值转换为权重值，如果小于某阈值（即不确定性超过某阈值）的数据直接relabel到：预测值类/新噪声类
         2. batch循环，以下为每个epoch中一个batch的训练：
            1. 如果不reweight：

如果为cross entropy loss或者heteroscedastic single loss（hs）

MC sample一次

如果为heteroscedastic multi loss（hm）

MC sample 设定好的次数

MC sample的结果为(T,N,(n\_class+n\_var))

* + - * 1. 如果reweight：

MC sample 设定好的次数

* + - * 1. 计算loss

如果为cross entropy loss，直接计算得到batch中每个图的loss，(N,)

如果为其他，则以MC sample的结果前n\_class位为均值后如果n\_var为10则分别为10类的方差，为1则10个共用一个方差，这样产生gaussian distribution，一共有T\*N\*n\_class个distribution（若hs为1\*N\*n\_class），在其中采样100次此时形状为(100,T,N,n\_class)，注意为重采样，用torch的rsample，送进softmax，之后在第一，第二维度取平均，故只剩下(N,n\_class)，这个就是预测的概率，再做nll即得到loss，(N,)

* + - * 1. 计算不确定性

用MC sample来按照a\_pred，entropy，var\_ratio，variance四种的定义计算，注意entropy和var\_ratio使用的是softmax后的平均值，与gal2015年的一致

* + - * 1. 计算weight

计算：

如果是inverse，直接变倒数

如果是exp，变指数倒数

归一化：

如果归一化在batch内

如果归一化到01之间，用sigmoid再乘以2减1，之后除以batchsize

* + - * 1. 如果是第0phase或1phase noreweight，按照noreweight回传，如果第1phase且指定reweight方式，loss对位乘以weights，加权之后回传
        2. 按照learning rate schedular调整学习率
  1. 进行两个phase的训练，第0个不reweight，第1个按照args指定reweight

# 前期探索性质实验

### 选择不确定性统计量

1. 实验配置：WideResNet40-2, MC\_n\_samples=10, batch\_size=128, dropout\_prob=0.2, epochs=75, learning\_rate=0.1, loss\_type='heteroscedastic\_single'
2. 实验结果：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| reference | [Kendall & Gal 2017](https://www.semanticscholar.org/paper/What-Uncertainties-Do-We-Need-in-Bayesian-Deep-for-Kendall-Gal/ff7bcaa4556cb13fc7bf03e477172493546172cd)(What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision) Learning for Computer Vision?) | | [Ribeiro et al. 2019](https://www.semanticscholar.org/paper/Deep-Bayesian-Self-Training-Ribeiro-Caliv%C3%A0/b6dea49867d4f5a0110d82085cb9df2b26e3eea2)(Deep Bayesian Self Training) | | other statistics |
| uncertainty related statistics/metrics 0: does not know or does not change 1: should ascend -1: should descend | data uncertainty: entropy of averaged softmax prob output(from T MC sampling of the network parameter) | model uncertainty: the average of variance of each logit(variance of MC samples) | data uncertainty: average of the predicted sigma of logits(or just predict one sigma for all logits) | model uncertainty: same as Gal's data uncertainty | variational ratio |
| increase trainset size(0.25,0.5,1), measure on original testset prediction: 0, -1 | same as column 4 | 0.2124, 0.1594, 0.0690 | 0.2220, 0.2441, 0.2778 | 1.4570, 1.5223, 1.6653 | 0.4761, 0.4804, 0.5034 |
| increase corruption prob on trainset(0.4, 0.6, 0.8), measure on original testset prediction: 1, 0 |  | 0.1013, 0.0690, 0.0169 | 0.2854, 0.2778, 0.3643 | 1.1832, 1.6653, 2.0912 | 0.3235, 0.5034, 0.7043 |
| train classes: range(0,5), test classes: range(0,5) train classes: range(0,5), test classes: range(5,10) prediction: 0, 1 |  | 0.0616, 0.0753 | 0.3366, 0.3334 | 1.6667, 1.7804 | 0.4984, 0.5550 |
| throughout training process, measure on original testset(we know that nn learns pattern quickly before noise) prediction: 0, -1 |  | -1 | 1, -1 | 1 | -1 |
| only some classes are corrupted, measure on original testset |  |  |  |  |  |

### 不同dropout rate

1. 实验配置：wrn28-10
2. 实验结果：
   1. dropout rate need to be hign enough, from now we choose 0.5

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **dropout\_prob** | **corruption\_prob=0.4** | | | | **corruption\_prob=0.6** | | | |
| **heteroscedastic single loss** | | **cross entropy** | | **heteroscedastic single loss** | | **cross entropy** | |
| best | last | best | last | best | last | best | last |
| **0** | 84.68 | 67.32 | 84.79 | 67.55 | - | - | 79.34 | 47.13 |
| **0.1** | 85.04 | 70.84 | 84.57 | 69.63 | 79.45 | 51.46 | 80.83 | 50.59 |
| **0.2** | 85.15 | 71.56 | 85.55 | 72.0 | 80.33 | 54.66 | 80.89 | 55.09 |
| **0.3** | 85.63 | 74.3 | 86.03 | 74.63 | 80.56 | 59.21 | 81.33 | 58.19 |
| **0.4** | 87.05 | 77.92 | 86.55 | 77.66 | 81.25 | 67.06 | 80.67 | 67.53 |
| **0.5** | 87.25 | 81.68 | 87.27 | 82.43 | 82.72 | 74.06 | 82.39 | 74.49 |
| **0.6** | 87.77 | 86.19 | 87.67 | 85.5 | 83.1 | 80.25 | 83.02 | 79.62 |
| **0.7** | 88.54 | 87.84 | 88.25 | 87.6 | 83.92 | 83.7 | 83.83 | 83.18 |
| **0.8** | 88.63 | 88.43 | 88.73 | 88.31 | 83.72 | 83.72 | 83.75 | 83.65 |
| **0.9** | 83.03 | 82.97 | 83.88 | 83.76 | 75.9 | 75.41 | 76.53 | 76.53 |

### Reweighting方法的上限（只用数据集中无噪声的样本训练）

1. 实验配置：wrn28-10，reweighting统计量=var\_ratio，reweight norm=sig\_m
2. 实验结果：

|  |  |  |  |
| --- | --- | --- | --- |
| **loss** | **reweighting** | **reweight norm** | **corruption prob=0.6(only clean data so 0.46 of clean data)** |
| s | var\_ratio | sig\_m | 92.82/92.39 |

### 各种reweighting方法的对比

1. 实验配置：MC\_n\_samples=10, batch\_size=128,  corruption\_type='unif', dropout\_prob=0.5, epochs=75, learning\_rate=0.1, lr\_scheduler='cosine\_m'(cosine monotone), momentum=0.9, nesterov=True, weight\_decay=0.0005, backnone: WRN28-10
2. 实验结果：choose inverse exp function over just inverse, like the one in DBST

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **loss** | **reweighting** | **reweight norm** | **corruption prob=0** | **corruption prob=0.6** |
| s | no reweight |  | 96.07/95.96 | 81.79/74.64 |
| var\_ratio | sig\_m(sigmoid modification) | 95.91/95.81 | 82.23/60.81 |
| batch norm | 96.18/95.8 | 80.61/55.61 |
| none reweight(from DBST) | 96/95.56 | 71.7/71.38 |
| entropy | sig\_m | 95.78/95.34 | 80.53/71.60 |
| batch norm | 95.99/95.41 | 80.42/60.02 |
| none reweight(from DBST) | 95.95/95,54 | 73.9/69.74 |
| hs | no reweight |  | 95.65/95.63 | 82.72/74.29 |
| var\_ratio | sig\_m | 95.92/95.64 | 82.5/76.9 |
| batch norm | 95.51/95.35 | 82.17/73.26 |
| none reweight(from DBST) |  | 9.42/5.74 |
| entropy | sig\_m |  | 81.38/79.47 |
| batch norm | 9.92/9.92 | 81.72/74.4 |
| none reweight(from DBST) |  | 9.65/3.22 |

### Reweighting方法预测低权重样本中噪音样本占比

1. 实验配置：MC\_n\_samples=10, batch\_size=128, clean\_only=False, corrupted\_classes=[0, 1, 2, 3, 4], corruption\_prob=0.6, corruption\_type='unif', cuda=True, depth=28, dropout\_prob=0.5, learning\_rate=0.1, log\_interval=100, loss\_debug='n', loss\_type='s', lr\_scheduler='cosine\_m', milestones=[40, 50], model='wrn', momentum=0.9, n\_var=10, nesterov=True, net\_arg=10, no\_cuda=False, num\_classes=10, p0epochs=0, p1epochs=60, relabel\_f='1', relabel\_thre=0, reweight='var\_ratio', reweight\_function='exp', reweight\_norm='sig\_m', reweight\_precision\_thre=0.1, seed=1, size\_ratio=1, test\_class=range(0, 10), train\_class=range(0, 10), weight\_decay=0.0005
2. 注意这里weight function是默认的e^(-uncer)
3. 实验结果：

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **corruption prob=0.6**  **wrn28-10**  **loss type=s**  **60 epochs with no relabel, just reweight** | **select the 10% sample with the least weight** | **30%** | **50%** | **70%** |
| none reweight | 84.14 | 78.69 | 71.69 | 65.13 |
| entropy reweight, exp function, sig\_m norm | 78.82 | 74.44 | 68.78 | 63.20 |
| var\_ratio reweight, exp function, sig\_m norm | 88.68 | 84.12 | 78.08 | 68.30 |

# 提出方法主题实验

### 第一步与第二步：尽可能找到更好的甄别noise的不确定性表征&各种与不确定性表征相匹配的weight function

#### 提出的

\alpha为weight function中超参数，weight=exp(-\alpha\*uncer)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Weight  Uncer function  statistics | \alpha=0.5 | 1 | 2 | 4 |
| A\_pred | 甄别率&正确率 |  |  |  |
| variance |  |  |  |  |
| entropy |  |  |  |  |
| Var\_ratio |  |  |  |  |

#### 前人的

|  |  |
| --- | --- |
|  | 甄别率&正确率 |
| Meta-weight-net |  |
| Noise modeling |  |

### 第三步：尝试分开noised sample与hardcase

#### Reweighting with two stage training

#### Relabeling

#### Relabeling + noise class