计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2020,29(4):187−194 [doi: 10.15888/j.cnki.csa.007347]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

一种深度学习批规范化改进算法①

罗国强[1], 李家华[1], 左文涛[2]

[1](广州科技职业技术大学 信息工程学院, 广州 510550)
[2](广州工商学院 计算机科学与工程系, 广州 510850)
通讯作者: 罗国强, E-mail: rogerjob@126.com

摘　要: 现实中采集的数据由于需要适应实际工程需求以及数据细粒度信息的分类形式多样, 样本数据间很难保持完全的独立同分布. 而非独立同分布数据会严重降低深度神经网络模型训练的鲁棒性以及特定任务上的泛化性能. 为了降低非独立同分布数据在模型训练和推断过程中的不良影响, 提出一种批规范化的改进算法. 该算法在神经网络模型训练开始前从数据集中取出一小批量数据做批规范化, 求解出的均值与方差作为参考值用来更新训练时的其他批量数据. 实验结果表明, 该改进算法一定程度上能够加快神经网络模型训练收敛, 相对于 BN 算法, 分类错误率降低了 0.3%, 提高了神经网络模型训练的鲁棒性. 在目标检测和实例分割任务上, 应用该改进算法的预训练模型能够有效提高某些检测算法的泛化性能.

关键词: 深度学习; 批规范化; 独立同分布; 鲁棒性; 泛化性

引用格式: 罗国强,李家华,左文涛.一种深度学习批规范化改进算法.计算机系统应用,2020,29(4):187–194. http://www.c-s-a.org.cn/1003-3254/7347.html

Improved Batch Normalization Algorithm for Deep Learning

LUO Guo-Qiang[1], LI Jia-Hua[1], ZUO Wen-Tao[2]

[1](College of Information Engineering, Guangzhou Vocational and Technical University of Science and Technology, Guangzhou 510550, China)
[2](College of Computer Science and Engineering, Guangzhou College of Technology and Business, Guangzhou 510850, China)

Abstract: It is needed to be adapted to the actual engineering requirements and the classification of the fine-grained data when we collect and annotate data. However, It is difficult to maintain complete independent and identical distribution between the samples. The non-i.i.d data seriously reduce the training's robustness of deep neural network model and the generalization performance of specific tasks. In order to overcome the shortcomings, this study proposes an improved algorithm of batch normalization, which normalizes a fix reference batch to calculate its mean and variance when the model training started, and then, the statistics of the reference batch is used to update other batches. Experimental results show that the proposed algorithm can accelerate the training convergence speed of the neural network model, meanwhile, the classification error is reduced by 0.3% compared with the BN algorithm. On the other hand, the robustness of neural network model and the generalization performance of some detection frameworks like object detection or instance segmentation are also improved effectively.

Key words: deep learning; batch normalization; non-i.i.d; robustness; generalization

深度学习是机器学习的一个子集, 通过组合低层特征形成更加抽象的高层语义以发现数据的特征分布表示. Hinton 等人[1]提出, 深度置信网络 (Deep Belief Network, DBN) 的训练可以由非监督逐层训练以及后

① 基金项目: 广东省教育厅重点科研平台项目 (2017GWTSCX064)
Foundation item: Major Scientific Research Platform Project of Education Bureau, Guangdong Province (2017GWTSCX064)
收稿时间: 2019-09-05; 修改时间: 2019-10-08; 采用时间: 2019-10-21; csa 在线出版时间: 2020-04-05

期微调的方式完成. 这为解决深层神经网络结构相关的训练优化难题带来希望. 但是在 2012 年之前, 深度学习仍然处于理论研究阶段, 还没有真正进入应用阶段. 这受制于两个原因, 第一是深度模型的训练需要大批量数据, 当时在模型预训练阶段一般采用 ImageNet[2] 数据集或其子集; 第二是计算力特别是 GPU 等硬件设备还没能够提供强大计算支持. 2012 年, 在 ImageNet 图像识别竞赛中, Hinton 和他的学生 Alex Krizhevsky 设计出 AlexNet[3]神经网络结构, 并以此在这次比赛中获得冠军. 这成为深度学习应用领域的标志性事件. 之后各种神经网络结构应运而生. 深度学习应用领域比较成功的有计算机视觉、自然语言处理、语音识别、目标检测等. 其中计算机视觉领域常用的基础网络结构有 AlexNet、VGGNet[4]、GoogleNet[5]、ResNet[6]、denseNet[7]、mobileNet[8]、shuffleNet[9,10]等. 这些基础网络[3−10]在实际应用中表现各有优劣. Goodfellow[11,12] 提出, 一个测试效果良好的分类器并不是学习到了所分类样本的真正底层意义, 只不过刚好构建了一个在训练数据上运行相当良好的模型, 而这个模型遇到一些空间中不太可能出现的点时, 模型能力的有限性就会随之暴露出来. 这个可能也就是许多模型对于外来样本泛化能力不足的原因.

深度学习会自动学习数据集上的特征分布. 一般而言, 数据需要尽可能地保持独立同分布. 但是实际情况由于各种条件的限制, 独立同分布往往是不可能做到的. 当非独立同分布的数据量比较大, 会严重影响模型对数据特征的有效学习, 模型的鲁棒性得不到保证. 实际应用中针对特定任务需要采集和标注适合自己任务的数据集, 这些数据的独立同分布特性往往得不到保证. 当然, 有关联的数据往往被用作上下文信息, 但就数据的角度看, 这些有关联的数据是不符合机器学习的独立同分布假设的.

本文研究分别采用了 LeNet-5[13], VGGNet16, ResNet50 作为基础网络, 实验对比发现, 批规范化算法[14](Batch Normalization, BN) 有非常大的可调空间, 其对最终的分类和检测识别结果也有一定程度的影响. 实验观察出该算法有下面三点不足:

(1) 批规范化算法在模型训练时要求有足够大的批量才能工作. 如果每批的数据量太少则会导致对统计数据的估计不准确. 每批的数据量减少则会显著增加模型误差. 现在的模型训练, 如果硬件设备条件允许, 则是采用大批量数据来训练的. 而这大批量数据中非独立同分布数据也可能比较多.

(2) 批规范化算法做规范化运算时, 每批的输出与这批量数据的每一个样本都有关联. 这从批规范化的计算公式可以看出.

(3) 目标检测、分割、视频识别和其他基于此的高级系统对批量大小的限制要求更高. 如 Faster[15]和 Mask R-CNN[16]系列检测框架使用批量为 1 或 2 的图像, 为了能够使用更高分辨率的图像, 批规范化算法通过变换而被线性层所固定.

基于以上 3 个观察, 本文提出批规范化算法的改进算法. 该改进算法在模型训练前, 在数据集中固定一个批量数据, 对该批量数据做规范化计算后, 其结果作为参考值对训练中的其他数据进行更新运算. 提高该改进策略, 实验效果比较明显.

1 相关工作

在神经网络训练过程中, 网络隐层的输入分布经常变化, 如果要使训练数据获得的模型能够有泛化能力, 就须使训练数据与测试数据满足独立同分布假设. 批规范化算法的初衷是使神经网络训练过程中输入分布保持一致, 即把神经网络每一层神经元输入值的分布规范化为标准正态分布. 这就很大程度防止了训练过程中的梯度消失问题. 在反向传播时以批为单位进行梯度更新, 极大加快了网络训练速度.

当批量数据样本很少的时候, 非独立同分布问题越来越显现, 模型的泛化能力大幅度降低, 模型也难以训练. 为解决这个问题, 批再规范化算法[17](Batch ReNormalization, B-RN) 在批规范化算法基础上引入两个参数, 通过对权重的尺度 (scale) 和偏移 (shift) 去适应一个小批量数据, 然后移动平均值消除归一化后的激活值对当前批量数据的依赖性. 其本质是网络参数前传中仿射变换修正小批量数据和数据集普适样本的差异, 使得该层的激活值在推断阶段能得到更有泛化性的修正.

批规范化算法主要适用于 CNN 或者 DNN 这种有固定深度的神经网络, 而在 RNN 中, 序列 (sequence) 的长度不一致, 不同的时间步 (time-step) 需要保存不同的统计特征, 可能存在一个特殊序列比其他序列长很多. 即 RNN 的深度不固定. 因此, 批规范化算法在

RNN 上效果不理想. 层规范化算法[18](Layer Normalization, LN) 不依赖于批量数据的大小和输入序列的深度, 同一层的输入样本拥有相同的均值和方差, 而不同的输入样本有不同的均值和方差. 如图 1 所示, 层规范化算法不在批量维度上做规范化, 而是在层的维度上做规范化. 层规范化算法在 RNN 中效果明显.

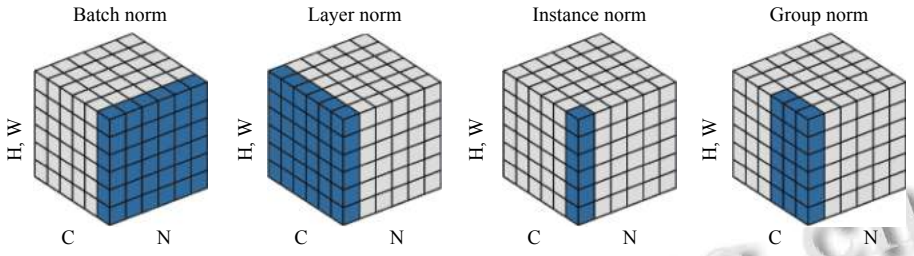

图 1 各种规范化算法示例

图像风格化应用中, 生成结果依赖于某个图像实例, 因此也不能在批量数据的维度进行归范化操作. 实例规范化算法[19](Instance Normalization, IN) 在 HW 的维度做规范化操作, 可以加速模型收敛, 并且保持每个图像实例之间的独立.

自适配规范化算法[20](Switchable Normalization, SN) 使用可微分学习, 为一个深度网络中的每一个需要做规范化操作的层确定合适的规范化操作. 区别于 BN 需要手工为每一个规范化层设计操作, 自适配规范化算法的这个特性可以减少手工设计量, 省去大量的实验.

群组规范化算法[21](Group Normalization, GN) 在通道的维度分组, 并在每组内进行规范化操作, 计算出均值和方差. 群组规范化算法的计算与批量数据的大小无关, 并且在神经网络模型训练时, 其准确度在大范围的批量下运行都非常稳定.

2 批规范化算法

令 $X=\{x_1, x_2, \cdots, x_N\}$为训练数据集, 在神经网络训练时, 可令损失函数:

$$l = F(x, \Theta) \tag{1}$$

其中, F 为每一层的非线性转换函数. 因此, 对于每一个样本, 使用随机梯度下降法[22,23](Stochastic Gradient Descent, SGD) 计算最优化参数Θ, 即,

$$\Theta = \arg\min_{\Theta} \frac{1}{N}\sum_{i=1}^{N} l(x_i, \Theta) \tag{2}$$

在该数据集中取一个批量数据 $B=\{x_1, x_2, \cdots, x_m\}$, 用该批量数据的平均梯度去拟合该批量数据每一个样本的梯度, 即:

$$\frac{1}{m}\frac{\partial l(x_i, \Theta)}{\partial \Theta} \tag{3}$$

由式 (1) 可知, 对于每一层收入 x, 有:

$$l = F_2(F_1(x, \Theta_1), \Theta_2) \tag{4}$$

因此可知, 每一个梯度优化步骤为:

$$\Theta_2 \leftarrow \Theta_2 - \frac{\alpha}{m}\sum_{i=1}^{m}\frac{\partial F_2(x_i, \Theta_2)}{\partial \Theta_2} \tag{5}$$

其中, α为学习率.

上述过程是典型的随机梯度优化过程. 大量的实验经验表明, 如果在神经网络训练过程中, 简单地使用随机梯度下降法会使训练的计算量和时间大大增加. 而且, 经过非线性转换函数 $F(*)$ 的作用, 每一层的输出不能保持同分布状态, 模型难以训练, 并且表达能力有限, 致使泛化性能降低. 因此, 批规范化算法引入两个可学习参数γ, β, 该参数有保持神经网络模型特征表达能力的作用. 那么, 神经网络每一层输出即:

$$y = \gamma\hat{x} + \beta \tag{6}$$

对于每一层输入 x, 有:

$$\hat{x} = \frac{x-\mu}{\delta} \tag{7}$$

其中, μ和δ即为每一层输入 x 的均值和标准差. 当:

$$\gamma = \delta \tag{8}$$

$$\beta = \mu \tag{9}$$

时, 神经网络训练时的每一层即可保持标准正态分布. 因此, 由以上分析, 令批规范化转换表示为:

$$BN_{\gamma,\beta}: x_i \rightarrow y_i \tag{10}$$

本文给出批规范化转换算法伪代码描述如算法 1. 其中, ε是一个常量, 其作用是为了保证方差数值上的稳定.

算法 1. 批规范化变换算法

输入: 批量数据 $B=\{x_1, x_2, \cdots, x_m\}$;
需学习的参数λ, β;
输出: $\{y_i=BN_{\gamma,\beta}(x_x)\}$

开始:

1. $\mu_B \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
2. $\delta_B^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_B)^2$
3. $\widehat{x}_i \leftarrow \frac{x_i-\mu_B}{\sqrt{\delta_B^2+\varepsilon}}$
4. $y_i \leftarrow \gamma\widehat{x}_i+\beta \equiv BN_{\gamma,\beta}(x_i)$

结束

3 批规范化改进算法

本文在第一章相关工作中介绍了批规范化算法的几个改进算法[18–21]. 该系列算法针对批规范化算法的许多不足之处做出改进. 但是各有其使用场景. 本文提出一种批规范化改进算法, 一定程度上解决神经网络训练过程中数据不能保持独立同分布问题. 即: 每一层批量数据的输出与这批量数据的每一个样本都有关联. 该改进算法即是降低这种数据的关联性.

该改进算法分为 3 步:

(1) 神经网络训练时, 每一层的输入数据中, 取出一个批量数据, 按照算法 1 取出该批量数据的均值和方差, 这两个统计数据在之后训练时固定下来保持不变. 令 $B_{\text{fix}}=\{x_1, x_2, \cdots, x_m\}$, 则:

$$\mu_{\text{fix}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \tag{11}$$

$$\delta_{\text{fix}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_{\text{fix}})^2 \tag{12}$$

(2) 求出其他批量数据的均值和方差, 并根据批量数据的样本量求出其他批量数据与固定批量数据的比例系数, 根据该比例系数更新其他批量数据的均值和方差. 令:

$$\lambda_1 = \frac{1}{m+1} \tag{13}$$

$$\lambda_2 = 1-\lambda_1 \tag{14}$$

则其他批量数据的均值和方差更新方式为:

$$\mu_{\text{others}} \leftarrow \lambda_2 \cdot \mu_{\text{others}} + \lambda_1 \cdot \mu_{\text{fix}} \tag{15}$$

$$\delta_{\text{others}}^2 \leftarrow \lambda_2 \cdot \delta_{\text{others}}^2 + \lambda_1 \cdot \delta_{\text{fix}}^2 \tag{16}$$

(3) 根据 (2) 得出的μ_{others}和δ_{others}^2在批量数据上作规范化运算.

根据以上分析, 可以得出批规范化改进算法的伪代码描述如算法 2.

算法 2. 批规范化变换改进算法

输入: 批量数据 $B_{\text{fix}}=\{z_1, z_2, \cdots, z_m\}$;
批量数据 $B=\{x_1, x_2, \cdots, x_m\}$;
需学习的参数λ, β;
输出: $\{y_i=BN_{\gamma,\beta}(x_x)\}$

开始:

1. $\mu_{\text{fix}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} z_i$
2. $\delta_{\text{fix}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(z_i-\mu_{\text{fix}})^2$
3. $\mu_{\text{others}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$
4. $\delta_{\text{others}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i-\mu_{\text{others}})^2$
5. $\mu_{\text{others}} \leftarrow \lambda_2 \cdot \mu_{\text{others}} + \lambda_1 \cdot \mu_{\text{fix}}$
6. $\delta_{\text{others}}^2 \leftarrow \lambda_2 \cdot \delta_{\text{others}}^2 + \lambda_1 \cdot \delta_{\text{fix}}^2$
7. $\widehat{x}_i \leftarrow \frac{x_i-\mu_{\text{others}}}{\sqrt{\delta_{\text{others}}^2+\varepsilon}}$
8. $y_i \leftarrow \gamma\widehat{x}_i+\beta \equiv BN_{\gamma,\beta}(x_i)$

结束

给定一个神经网络, 加入批规范化改进算法后的训练和推断方式与批规范化算法无异. 下面给出具有批规范化改进算法网络层的神经网络训练和推断算法描述如算法 3.

算法 3. 神经网络训练与推断

输入: 神经网络 Net; 可训练参数Θ;
每一层的输入$\{x^{(k)}\}_{k=1}^{K}$
输出: 推断网络$\text{Net}_{\text{BN}}^{\text{inf}}$

开始:

1. 初始化训练网络$\text{Net}_{\text{BN}}^{\text{train}} \leftarrow$Net
2. **for** k=1 to K **do**
3. 根据算法 2 在该训练$y^{(k)}=\text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$
4. 修改$\text{Net}_{\text{BN}}^{\text{train}}$每层输入, 用$y^{(k)}$替换$x^{(k)}$
5. **end for**
6. 优化$\text{Net}_{\text{BN}}^{\text{train}}$的参数$\Theta \cup \{\gamma^{(k)},\beta^{(k)}\}_{k=1}^{K}$
7. 训练好网络后$\text{Net}_{\text{BN}}^{\text{inf}} \leftarrow \text{Net}_{\text{BN}}^{\text{train}}$
8. **for** k=1 to K **do**
9. 对多个大小为 m 的批量数据 B, 计算:
$E[x] \leftarrow E_B[\mu_B]$
$\text{Var}[x] \leftarrow \frac{m}{m-1}E_B[\delta_B^2]$
10. 在推断网络$\text{Net}_{\text{BN}}^{\text{inf}}$中, 推断方式为:
$y=\frac{\gamma \cdot x}{\sqrt{\text{Var}[x]+\varepsilon}}+(\beta-\frac{\gamma \cdot E[x]}{\sqrt{\text{Var}[x]+\varepsilon}})$
11. **end for**

结束

4 实验

4.1 实验环境

本文算法的实验环境配置为 Intel I7 8700 处理器, 8 块 NVIDIA GTX 1080ti 显卡, 64 GB RAM 的深度学习服务器. 软件环境配置为 Ubuntu 16 系统, GCC 5.4, CUDA 9, OpenCV 3, TensorFlow, Caffe/Caffe2, Detectron 框架等.

4.2 实验结果及分析

为了验证算法各方面性能, 本文进行了 3 组对比实验, 分别为训练速度对比, ImageNet 分类实验, 目标检测实验.

4.2.1 模型训练速度对比

为了快速得到训练结果, 简化调参步骤, 使实验对照组更具说服力, 该组对比实验模型选用 LeNet-5 和 VGG-16, 数据集选用 Mnist, 对比算法选用批规范化算法以及本文改进算法. 图 2 和图 3 分别说明了批规范化算法和改进算法在 Mnist 数据集上两个神经网络模型的训练情况.

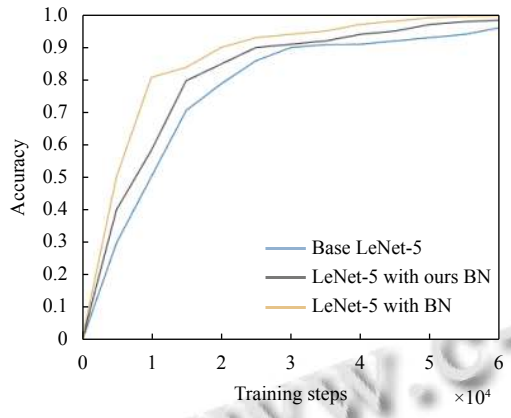

图 2 LeNet-5 训练曲线

由图 2 和图 3 训练曲线可以看出, 在 Mnist 数据集上, LeNet-5 和 VGG-16 的 3 条曲线的趋势大致是一样的, 随着训练步的增加, 训练精度分别呈上升趋势.

在模型训练过程中, 相对于没有规范化操作的 LeNet-5 或者 VGG-16 网络, 加入本文改进算法后, 网络模型训练速度有所提高, 特别是在训练的前期, 改进后的模型更快的进入饱和点. 图 2 中在训练步为 26 000 时可以看出, 3 条曲线的斜率变缓, 模型训练进入饱和阶段. 其中, 批规范化算法最先进入饱和点, 其次本文改进算法进入饱和点, 没有进行规范化操作的 LeNet-5 最后进入饱和点. 图 3 中 VGG 曲线也遵循这个趋势.

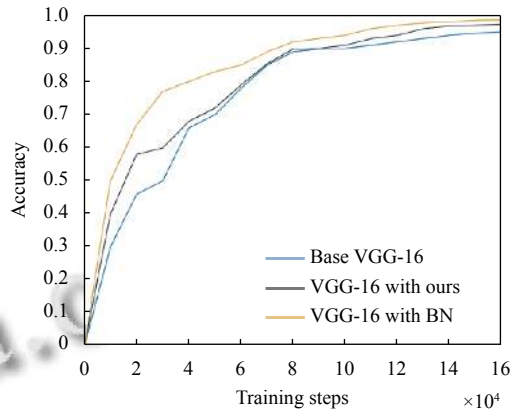

图 3 VGG-16 训练曲线

由图 2 和图 3 也可以对比得出, 在训练速度上, 改进算法可以加快网络模型的训练, 但是相对于批规范化算法, 改进算法没有明显优势. 可能的原因在于, 在设计改进算法时引入了一个固定批量数据, 对其做了规范化操作. 因此, 改进算法相对于原批规范化算法, 相当于做了两次规范化操作. 计算量有所增加, 体现在训练曲线上即为饱和点右移.

4.2.2 ImageNet 数据集分类实验

该组实验采用 ResNet50 作为基准网络, 在 ImageNet 数据集上进行分类训练. 实验采用了 8 块 GPU, 批规范化运算时, 批量数据的均值和方差将在不同的 GPU 上进行计算. 卷积层初始化采用 He 等 [24]的方法, γ初始化为 1, 而每一个残差模块的最后一个规范化层的γ初始化为 0. 训练时权重衰减为 0.0001, 学习率初始化为 0.001. 当模型训练分别到 100 epochs、120 epochs 和 160 epochs 时, 学习率依次降低 10 倍. 该训练参数设置分别参考了文献[17–21].

为了作对比, 本文在 ResNet50 基准网络上分别采用 BN、LN、IN、GN 以及本文改进算法 Ours 进行分类训练.

图 4 和图 5 分别为 ImageNet 训练集和验证集上的训练曲线. 因为几个算法都是在 BN 算法的基础上改进而来, 本实验以 BN 算法作为基准对比组. 由图 4 和图 5 可以看出, ResNet50 在模型训练时, 引入 5 种算法后都能够收敛. 此时规范化批量数据大小为 128, 群

组规范化算法的分组数为 32. 表 1 和表 2 列出了 ResNet50 基准网络引入 5 种算法后在 ImageNet 训练集和验证集上训练错误率.

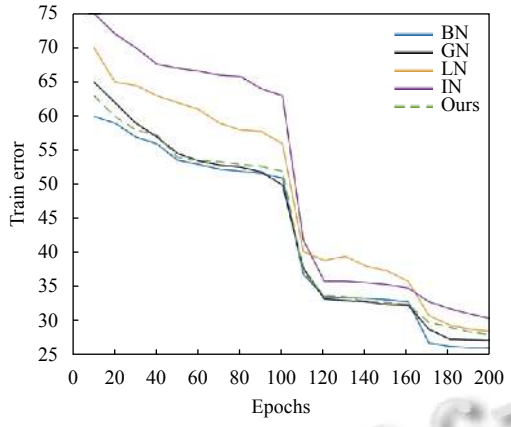

图 4 ImageNet 训练 error 曲线

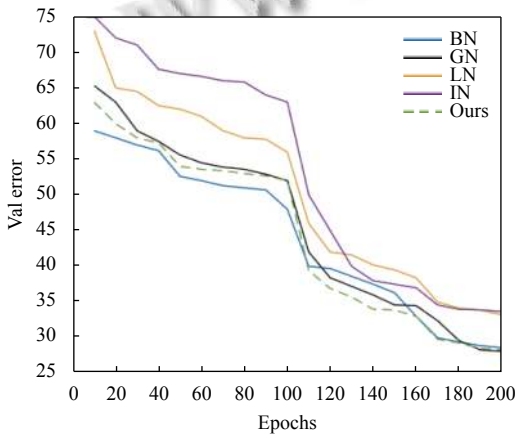

图 5 ImageNet 验证集训练 error 曲线

表 1 ImageNet 训练集结果对比 (%)

算法	BN	LN	IN	GN	**Ours**
Error	**26.3**	28.7	30.5	27.4	**28.2**
vs. BN	~	2.4	4.2	1.1	**1.9**

表 2 ImageNet 验证集结果对比 (%)

算法	BN	LN	IN	GN	**Ours**
Error	28.6	33.2	33.7	28.1	**28.3**
vs. BN	~	4.6	5.1	–0.5	**–0.3**

分析表 1, 可以看出 BN 算法在批量数据大小为 128 时, 错误率比其它 5 个算法低. 相对于 BN 算法训练集上 26.3% 的错误率, GN 算法高出 1.1 个百分点, 本文的改进算法 Ours 高出 1.9 个百分点. 而 LN 和 IN 算法则分别高出 2.4 和 4.2 个百分点. 因此, 本文改进算法 Ours 在 ImageNet 训练集上相对于 BN 算法和 GN 算法训练错误率并没有优势, 但是优于 LN 算法和 IN 算法.

图 4 和表 1 在训练集上的训练结果并不能说明一个模型的泛化能力, 一个模型即使训练拟合得很好, 其对于外来样本可能没有泛化性. 分析表 2 可知, 5 个算法在 ImageNet 验证集错误率均高于在 ImageNet 训练集上的错误率. 对照组 BN 算法验证集错误率为 28.6%, 相对于 GN 算法 28.1% 的错误率高出 0.5 个百分点, 而相对于本文改进算法 Ours 的 28.3% 错误率则高出 0.3 个百分点. 可以得出, 本文改进算法验证集上错误率在 GN 算法和 BN 算法之间, 略差于 GN 算法, 而优于 BN 算法. LN 算法和 IN 算法的错误率相对于 BN 算法则分别高出 4.6 和 5.1 个百分点.

综上分析, 在本实验环境和条件下, 本文改进算法 Ours 相对于 BN 算法和 GN 算法有一定的竞争优势. Ours 算法在训练集上表现不如 BN 算法和 GN 算法, 但是在验证集上则优于 BN 算法, 略差与 GN 算法. 即, 本文改进算法对外来样本处理得较好, 训练的模型泛化性能相对于 BN 算法有一定程度的提高. 由以上实验也可以得出, 采用 LN 算法和 IN 算法训练的 ResNet 网络虽然训练能够收敛, 但是训练集和验证集分类错误率都比较高. 这是因为 LN 算法主要用于 RNN, 而 IN 算法主要用于图像风格化等, 都有其特定的适用场景, 其适用范围可能没有 BN 算法、GN 算法以及本文改进算法 Ours 那么广.

4.2.3 目标检测和分割实验对比

该组实验中, 本文验证改进算法在 COCO2017 数据集上目标检测和实例的效果. 实验采用深度学习框架 caffe2, 以及通用目标检测框架 Detectron, 目标检测算法使用 Mask R-CNN.

表 3 列出了采用 Mask R-CNN 作目标检测和实例分割时, 以 ResNet50 的 conv4 作为目标的特征提取层, conv5 层接一个 ROI 层用于目标分类和回归. 在该实验中, 改进算法比批规范化算法目标边框的 AP^{bbox} 值提高了 0.5, 而掩膜的 AP^{mask} 值提高 0.1. 该实验说明在目标检测和实例分割任务上, 一定程度上本文改进算法的目标检测泛化能力优于 BN 算法. 图 6 为采用本文改进算法后 Mask R-CNN 效果图.

表 3 Mask R-CNN 目标检测与分割

backbone	AP^{bbox}	AP^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
BN	36.5	56.8	40.6	32.4	53.3	34.3
Ours	**37.0**	56.8	**41.2**	**32.5**	53.0	**34.7**

图 6 Mask R-CNN 效果图

4.2.4 鲁棒性说明

本文算法在神经网络模型训练过程中能够一定程度上提高基准网络模型的鲁棒性. 其中, 在 VGG 网络模型训练过程中尤为明显, 梯度弥散的现象出现的几率很小, 网络训练易于收敛. 基于大量对比实验以及分析得出:

(1) 本文算法在大批量数据训练过程中可以降低样本之间的关联性. 这种特性使模型易于训练.

(2) 鲁棒性的强弱很大程度依赖于基准网络. Su 等[25]通过大量实验对 18 个常用的分类基准网络进行对抗样本研究分析发现: 1) 准确度越高的模型的普遍鲁棒性越差, 且分类错误率的对数和模型鲁棒性存在线性关系; 2) 相比于模型的大小, 模型的结构对于鲁棒性的影响更大; 3) 黑盒迁移攻击是一直以来都比较困难的任务, 但在 VGG 系列模型上生成的对抗样本可以比较容易地直接攻击其它的模型. 这在一定程度佐证了本文实验.

5 结束语

深度神经网络模型的训练到目前为止还是非常具有挑战性的研究点. 本文在批规范化算法方面对其进行了探索. 在神经网络训练中, 改进算法有比较好的实验表现, 能够一定程度上提高分类精度, 在特定检测任务上提高检测精度. 但是, 该改进算法在模型训练前传过程中做了两次批规范化运算, 计算量增加不少, 因此, 深度神经网络模型的训练速度减慢了. 这是对时间与空间, 性能与速度的取舍平衡的考虑. 本课题将继续这两方面的研究.

参考文献

1 Hinton GE, Osindero S, Teh YW. A fast learning algorithm for deep belief nets. Neural Computation, 2006, 18(7): 1527–1554. [doi: 10.1162/neco.2006.18.7.1527]

2 Deng J, Dong W, Socher R, *et al*. ImageNet: A large-scale hierarchical image database. Proceedings of 2009 IEEE Conference on Computer Vision and Pattern Recognition. Miami, FL, USA. 2009. 248–255.

3 Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information Processing Systems. Lake Tahoe, NV, USA. 2012. 1097–1105.

4 Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv: 1409.1556, 2014.

5 Szegedy C, Liu W, Jia YQ, *et al*. Going deeper with convolutions. Proceedings of 2015 IEEE Conference on Computer Vision and Pattern Recognition. Boston, MA, USA. 2015. 1–9.

6 He KM, Zhang XY, Ren SQ, *et al*. Deep residual learning for image recognition. Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition. Las Vegas, NV, USA. 2016. 770–778.

7 Huang G, Liu Z, van der Maaten L, *et al*. Densely connected convolutional networks. Proceedings of 2017 IEEE Conference on Computer Vision and Pattern Recognition. Honolulu, HI, USA. 2017. 2261–2269.

8 Howard AG, Zhu ML, Chen B, *et al*. MobileNets: Efficient convolutional neural networks for mobile vision applications. arXiv: 1704.04861, 2017.

9 Zhang XY, Zhou XY, Lin MX, *et al*. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. arXiv: 1707.01083, 2017.

10 Ma NN, Zhang XY, Zheng HT, *et al*. ShuffleNet V2: Practical guidelines for efficient CNN architecture design. Proceedings of the 15th European Conference on Computer Vision. Munich, Germany. 2018. 122–138.

11 Goodfellow IJ, Shlens J, Szegedy C. Explaining and harnessing adversarial examples. International Conference on Learning Representations. arXiv: 1412.6572, 2015.

12 Akhtar N, Mian A. Threat of adversarial attacks on deep learning in computer vision: A survey. IEEE Access, 2018, 6: 14410–14430. [doi: 10.1109/ACCESS.2018.2807385]

13 LeCun YL, Bottou L, Bengio Y, *et al*. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 1998, 86(11): 2278–2324. [doi: 10.1109/5.726791]

14 Ioffe S, Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. Proceedings of the 32nd International Conference on Machine Learning. Lille, France. 2015. 448–456.

15 Ren SQ, He KM, Girshick RB, *et al*. Faster R-CNN: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(6): 1137–1149. [doi: 10.1109/TPAMI.2016.2577031]

16 He KM, Gkioxari G, Dollár P, *et al*. Mask R-CNN. Proceedings of 2017 IEEE International Conference on Computer Vision. Venice, Italy. 2017. 2980–2988.

17 Ioffe S. Batch renormalization: Towards reducing minibatch dependence in batch-normalized models. Proceedings of the 31st Conference on Neural Information Processing Systems. Long Beach, CA, USA. 2017. 1945–1953.

18 Ba JL, Kiros JR, Hinton GE. Layer normalization. arXiv: 1607.06450, 2016.

19 Ulyanov D, Vedaldi A, Lempitsky V. Instance normalization: The missing ingredient for fast stylization. arXiv: 1607.08022, 2017.

20 Luo P, Ren JM, Peng ZL, *et al*. Differentiable learning-to-normalize via switchable normalization. arXiv: 1806.10779, 2018.

21 Wu YX, He KM. Group normalization. arXiv: 1803.08494, 2018.

22 Sutskever I, Martens J, Dahl G, *et al*. On the importance of initialization and momentum in deep learning. Proceedings of the 30th International Conference on International Conference on Machine Learning. Atlanta, GA, USA. 2013. III-1139–III-1147.

23 Duchi J, Hazan E, Singer Y. Adaptive subgradient methods for online learning and stochastic optimization. The Journal of Machine Learning Research, 2011, 12: 2121–2159.

24 He KM, Zhang XY, Ren SQ, *et al*. Delving deep into rectifiers: surpassing human-level performance on ImageNet classification. Proceedings of 2015 IEEE International Conference on Computer Vision (ICCV). IEEE Computer Society. Santiago, Chile. 2015. 1026–1034.

25 Su D, Zhang H, Chen HG, *et al*. Is robustness the cost of accuracy? -- A comprehensive study on the robustness of 18 deep image classification models. arXiv: 1808.01688, 2018.