人工智能基础
第二次大作业

班级: 自72
姓名: 程晔安
学号: 2017011627
Kaggle 账号: 502870976@qq.com
Kaggle 名称: LA12138

实验时间 2019年 12月 13日

目录

1 任务描述

本次作业的数据包含30000张图片组成的训练集和5000张图片的测试集,图片可以被分为如下所示的十类.

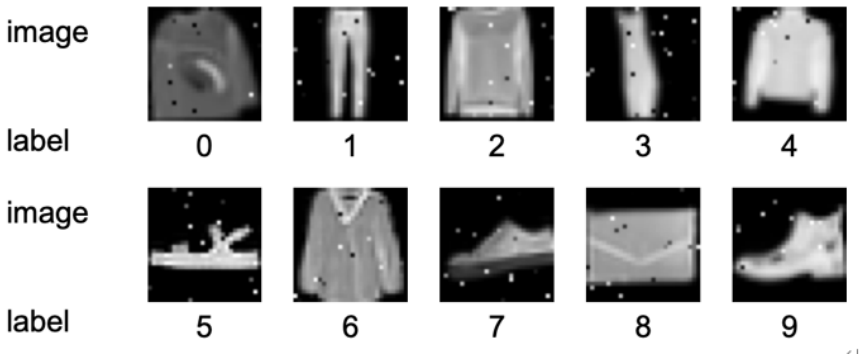

图 1: 数据集图片

可以看到,本次作业处理的数据是28×28的灰度图片,图片中存在较为明显的噪点,总体任务是十分类任务.

2 训练过程

本次实验的所有模型使用python编写,利用的深度学习框架为pytorch1.2.0,所有的实验都是在NVIDIA GTX 1080Ti GPU上完成的.

下面分为模型选择,模型融合,数据增强,超参数选择几个板块描述模型训练过程中我的思考和选择.

2.1 模型选择

模型的选择是整个任务成功与否最重要的因素.有一个表达能力强的模型,就可以更好地提取出数据图片中的语义信息,成功地进行分类.本次实验我一共选用了两类分类网络,分别为Resnet [1] 和Densenet [2].在这两种网络中,我又分别选择了ResNet18,ResNet50,DenseNetBC和DenseNet121进行了具体的实现.在训练完成之后,我再利用"模型融合"技术将几个网络的输出加以汇总,得到了我的最终模型结果.

下面分别介绍这两种网络和我选择这两种网络的原因.

2.1.1 ResNet

ResNet(Residual Neural Network)是由微软研究院的Kaiming He等四人提出深度神经网络,其在ILSVR2015 比赛中在top5的错误率仅为3.57%,同时在ImageNet detection等大赛上也都夺得了冠军,可以说效果极为突出,其引用量也达到了3.5w以上,可以说是现在应用最广的分类网络.因此我选择了使用这种网络.

其思想的核心是引进了如图所示的残差结构.

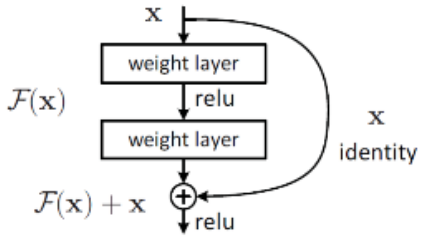

图 2: ResNet构建快

如图所示的残差结构让网络在更深层次中仍然能够保留比较浅层的信息,在参数回传的时候也不会容易丢失信息, 这样的特点让网络可以构建得更深,同时收敛速度更快,优化更容易,同时还可以保持模型的参数较少.

在本次训练过程中,我选择使用ResNet18和Resnet50两种网络结构,这样选择的原因是:首先,本次任务是十分类任务,图片本身只有784个像素点,问题相对比较简单,因此不需要太过深层的网络即可完成任务.但是如果选择单种网络,又有可能出现单个网络在某个类别之间的判别能力不强,导致表达能力有缺点的情况.最终我就选择了相对较浅的两种ResNet进行了训练.ResNet18和Resnet50的网络结构图如下所示.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

图 3: ResNet18和ResNet50

最后从结果上来看,我将两个网络训练了50个epoch,batch size为32,最终选出的最优模型在valid集上的正确率在89%左右.

2.1.2 DenseNet

DenseNet(Densely Connected Convolutional Networks)是CVPR2017的best paper,同时其作者为自动化系的黄高老师,我感受到这篇论文是离我很近的一篇论文, 因此我抱着学习的心态选择了这种模型.DenseNet借鉴了ResNet中的思想,使用捷径连接的方式成功减轻

了 ”gradient-vanishing”问题,强化了特征的传输,强调参数重用,并可观地减少了参数量.如下图所示为一个 DenseBlock的模型示意图,可以看到,很多skip connection使特征的传递更加方便,不用经过很深的网络.

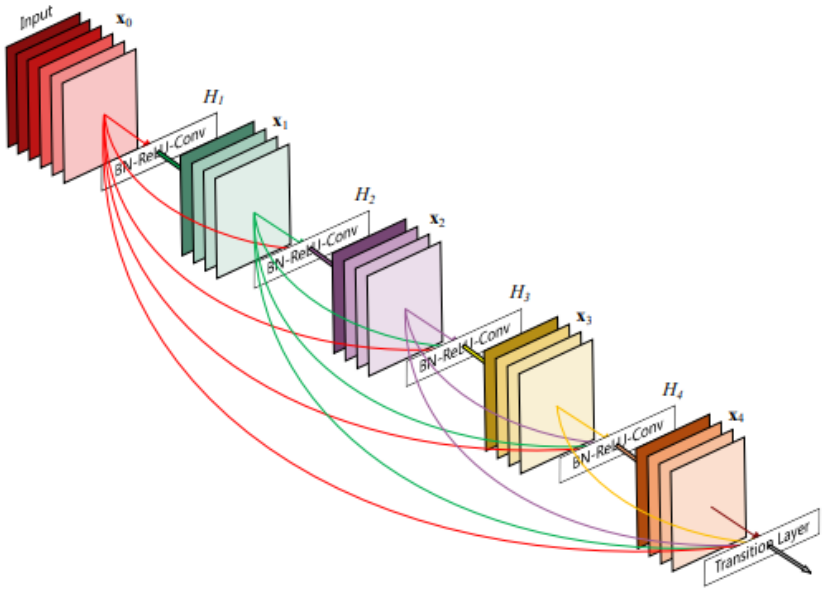

图 4: DenseNet

同样,我选择了最浅的两种DenseNet,即DenseNetBC和DenseNet121进行训练. 由于网络规模较ResNet较大,在实际应用中,我只训练了10个epoch,batch size为32 在valid集上,我的准确率在92%左右.

2.2 交叉检验和模型融合

在本次实验的训练过程中,我还使用了k折交叉检验(k-fold cross validation)和模型融合(Ensemble learning) [5]来提高我训练过程的效果. 在使用了这两种技巧之后,我整个模型的训练结果在Kaggle网站上测试集的正确率提高了4%左右.这是极大的提升. 下面分别介绍这两种技术.

2.2.1 交叉检验

在本次实验中,我采用了k-折交叉检验.k-折交叉检验的做法是首先将训练集平均分为如图所示的k个数据子集.

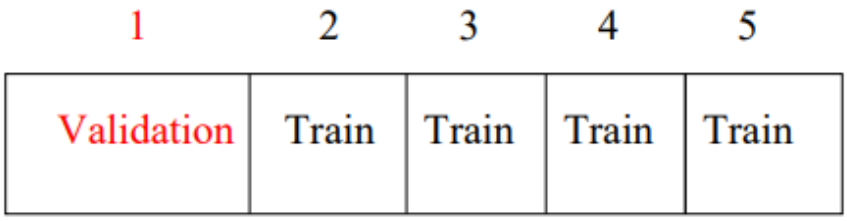

图 5: k-折交叉检验

对每个$k = 1, 2, ..., K$,用其他k-1个数据部分训练模型.假设我们的模型的超参数为λ,参数为$\hat{\beta}$,则可记为模型预测出的结果为$\hat{\beta}^{-k}(k)$,可计算其预测第k折的预测误差为:

$$E_k(\lambda) = \Sigma_{i \in kthpart}(y_i - x_i\hat{\beta}^{-k}(\lambda))^2$$

则可以得到其相应的交叉检验总误差为:

$$CV(\lambda) = 1/K \sum_{k=1}^{K} E_k(\lambda)$$

在本次实验中,我取$k = 5$.实际上,我并没有进行直接的k-折交叉检验,而是把它作为模型融合操作的中的Blending进行使用.具体使用方法请见下一版块.

2.2.2 模型融合

模型融合的方法指的是训练多个模型,再将模型的结果以某种方式融合起来的方法. 我在这里采用了模型融合中的Blending的方法,即综合k-fold Cross Validation 的思想,采用不同模型$model_A, model_B$来分别利用不同的k-1个数据部分进行训练.假设$model_A, model_B$的参数为$\hat{\beta}_A$ $\hat{\beta}_B$ 则上述的模型计算其预测第k折的预测误差为:

$$E_k(\lambda) = \Sigma_{i \in kthpart}(y_i - x_i\hat{\beta}_A^{-k}(\lambda))^2$$

$$E_k(\lambda) = \Sigma_{i \in kthpart}(y_i - x_i\hat{\beta}_B^{-k}(\lambda))^2$$

交叉检验总误差仍可以写为:

$$CV(\lambda) = 1/K \sum_{k=1}^{K} E_k(\lambda)$$

利用模型融合的方法有如下所示的好处:

- 过拟合的问题.一般来说,训练集上的高正确率并不意味着模型的优秀.有可能在训练集准确率过高的情况下,模型的泛化能力就会变差,因此产生过拟合的问题
- 模型的随机性和表达能力的不同.一般来说,由于初始化的不同和浮点误差, 神经网络的训练具有随机性.而这种随机性往往带来结果上的变差.同时,不同模型的能力不同,可能对某一种标签的分类性能较好或较不好,这样的问题都可以通过模型融合的方式进行解决.

如图所示为进一步的说明.为了学到图中的复杂的Decision Boundary,我们可以使用不同的模型在不同区域进行拟合,最后再把结果加以汇总,就会得到很好的结果.

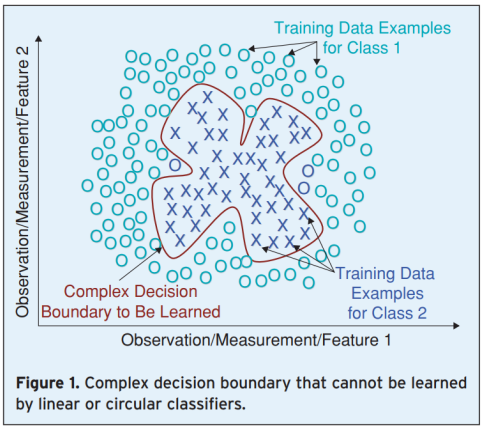

图 6: 复杂目标

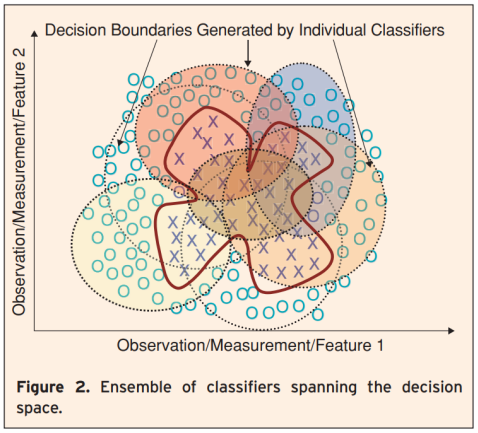

图 7: 模型融合

2.3 数据增强

由于本次实验所给的数据量不够大,有可能会出现训练集过少,网络无法成功学到有用的信息的问题.因此,我对原有数据进行了数据增强.

我分别进行了如下所示的数据增强操作:

- 随机纵向反转
- 加椒盐噪点.这样的处理是考虑到模型对于图片中可能存在的椒盐噪声的鲁棒性.
- 随机裁剪
- 均值滤波去噪.这个处理是为了让模型能够学到正常的图片信息中的特征.

加上原图片,我一共就有150000张图片.这样的数据量让训练过程更加充分,模型能够见到更多种类的相同的图片.此外,我发现不同的网络模型在数据是否正规化(Normalize)的情况下表现有好有坏, 因此我分别进行了是否正规化的实验.并利用模型融合将结果综合起来;我在最后将输入的图片变换到224×224 大小,这样可以利用图片边缘的信息,也让网络中的参数更多,更易学到特征.

2.4 超参数选择

在训练过程中,其他超参数的选择也显得十分重要.我在batch size,epoch,学习率和优化器的选择上都进行了多种尝试.

2.4.1 batch size

在训练初期,由于我有很好的GPU计算资源,为了加快网络训练速度和收敛速度,我将batch size设置为512,但是发现这样的训练过程得到的结果并不好.在经过调研和实践之后,我发现

虽然大的batch size可以让模型更快的收敛,但是收敛的过程太快也会导致模型的泛化性能变差,收敛成"sharp minimizers". [3]. 因此,我最后将模型的batch size设置成32,在横向对比过程中,这样训练一个模型的时间大约在一整天左右,显存大概在4.5G左右.学出的模型的性能最好.

2.4.2 epoch和择优策略

在我的训练过程中,我将整个30000张数据集图片分成了27000张测试图片和3000张valid图片.为了找出性能最好的模型,我在每个epoch结束之后都在validation集上测试目前模型,在整个训练过程中保存在validation集上表现最好的模型.由于validation集在整个训练过程中没有被模型见过,这样的测试是检测模型泛化性能很好的方法.

在实际训练中,我设置模型训练50个epoch左右,最后选出的最后模型一般在10-20个epoch中.

2.4.3 学习率和优化器

在训练模型的过程中,我发现学习率的设置也是影响训练效果的很重要的指标.具体来说,我一开始尝试使用的优化器是 Adam优化器,学习率是$5 \times 10^{-}4$.但是在之后的应用过程中,我发现Adam优化器虽然收敛速度比较快,但是比较不稳定,有可能会产生在最优点附近徘徊的情况,因此我最后采用了SGD with momentum [6] 和Adabound [4]两种优化器进行训练.最后再用模型融合的方式将两者训练出来的模型进行组合.

SGD是最基础的利用梯度进行更新优化的优化器,其更新过程可以用如下过程进行描述:

$$x_{t+1} = x_t - \alpha \nabla f(x_t)$$

但是其有更行速度过慢,没有办法逃出鞍点的缺点.因此SGD with Momentum用如下所示的方法改进了这一缺点:

$$v_{t+1} = \rho v_t + \nabla f(x_t)$$

$$x_{t+1} = x_t - \alpha v_{t+1}$$

让参数在更新过程中使用了上一步的梯度信息,这样的操作可以让函数有能力脱离鞍点,继续向全局最优点前进.

AdaBound是一种最新才提出的优化器,他改进了Adam优化器在接近最优点之后变得不稳定的缺点,采用先Adam,在某个阈值以后再变成SGD的操作,既保证了收敛速度,又保证了在最优点之中的稳定性.在实际运用中效果较好.

3 结果分析

3.1 最终模型

综合上面的训练方法,我最终的模型由八个模型组成,各个模型选择的网络结构,超参数均不相同,在目前, 在Kaggle上达到了92.133%的正确率(rank 5 in 197).下面分别列出这8个模型的训练方法:

模型编号	网络	优化器	学习率	正则化	Valid集正确率
1	ResNet18	AdaBound	5.00E-04	是	88.78%
2	ResNet50	AdaBound	5.00E-04	是	89.43%
3	DenseNetBC	AdaBound	5.00E-04	是	91.37%
4	DenseNet121	AdaBound	5.00E-04	是	90.97%
5	ResNet50	SGD with Momentum	1.00E-03	是	92.12%
6	ResNet50	SGD with Momentum	1.00E-03	否	92.33%
7	DenseNetBC	SGD with Momentum	1.00E-03	是	91.78%
8	DenseNetBC	SGD with Momentum	1.00E-03	否	90.54%

图 8: 数据集图片

一般来说,在本地validation集上得到的正确率和Kaggle上的正确率相差不大.

3.2 正确率分析

我将我使用的四个网络的正确率和loss截取前30个epoch的范围,得到了如下所示的图形,如图所示.

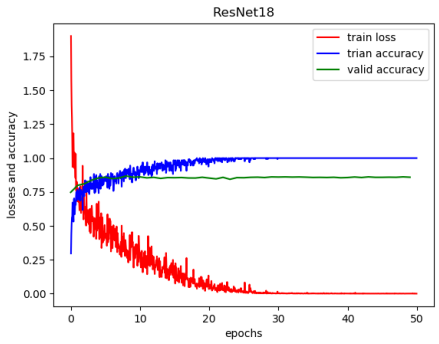

图 9: ResNet18

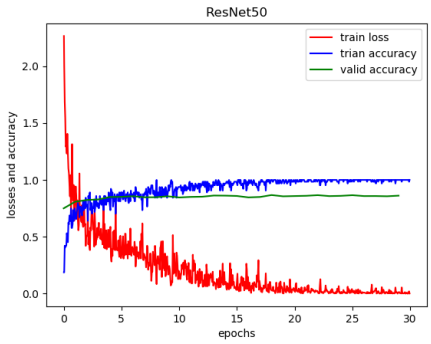

图 10: ResNet50

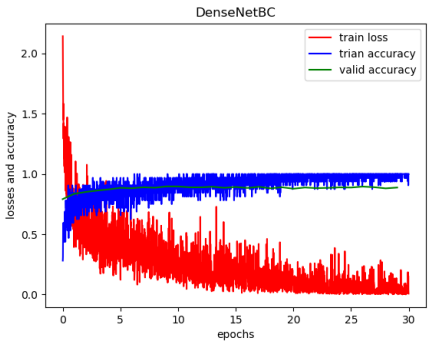

图 11: DenseNetBC

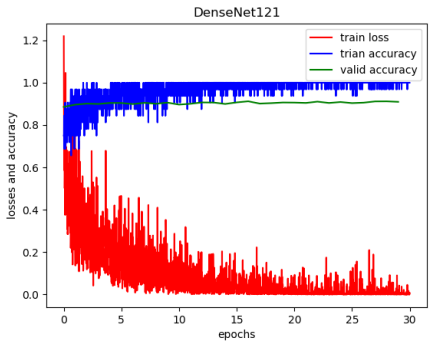

图 12: DenseNet121

可以看到,各个网络的收敛过程不同,在整个过程中模型的波动程度也不相同,但是各个模型最后在 validation集上都达到了将近90%的正确率,这说明数据集的特征被充分地学习到了.

3.3 ROC分析

我对主要的四个模型在validation集上进行了ROC(Receiver operating characteristics)分析, 这种基于假设检验的分析可以衡量模型对于各个类型的分辨能力,我最后得到了如下所示的曲线.

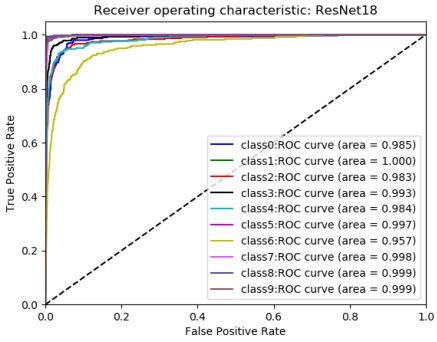

图 13: ResNet18_roc结果

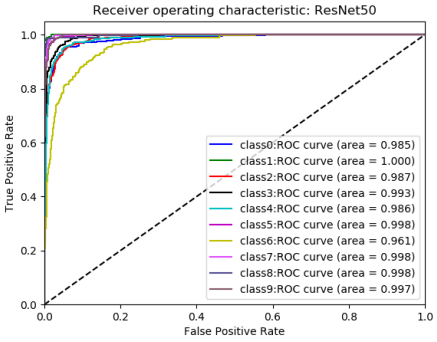

图 14: ResNet50_roc结果

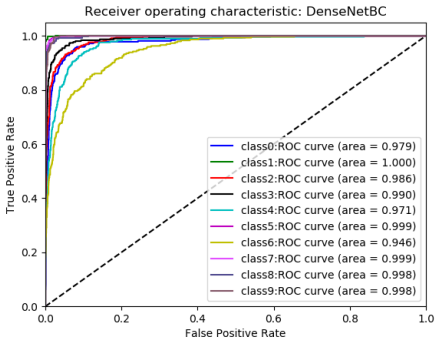

图 15: DenseNetBC_roc结果

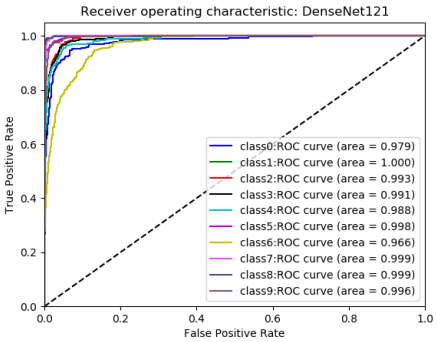

图 16: DenseNet121_roc结果

可以看到,模型对每个label都有比较好的分类性能,但是对模型6的分类性能有可能比较差,因为确实裙子和衬衫的标签的图像比较相近,很难区分.

3.4 测试结果分析

4 实验总结

在本次实验中,我收获了很多,首先是自己动手学习搭建神经网络,并利用pytorch的各种框架进行训练的能力; 其次是文献阅读和代码复现能力.在本次作业中,我大量查阅了论文资料,并在理解的基础上参照pytorch官方的实现代码对论文中的模型进行复现;最后就是模型调参的能力,在训练过程中,其实有很多不起眼的小点都可以深刻地决定模型的训练结果的好坏,尤其是在模型的能力变得比较好以后,进一步的优化就变得难上加难.在模型效果

不好的时候,我学到了要积极交流,寻找原因并勇于改正,这样的态度才可以成功地做出比较好的实验结果.

感谢助教和老师在整个大作业过程中耐心的讲解和答疑,我收获了很多!

参考文献

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[2] G. Huang, Z. Liu, G. Pleiss, L. Van Der Maaten, and K. Weinberger. Convolutional networks with dense connectivity. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[3] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima, 2016.

[4] L. Luo, Y. Xiong, Y. Liu, and X. Sun. Adaptive gradient methods with dynamic bound of learning rate. In *Proceedings of the 7th International Conference on Learning Representations*, New Orleans, Louisiana, May 2019.

[5] R. Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[6] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural networks : the official journal of the International Neural Network Society*, 12 1:145–151, 1999.