

深度学习归一化算法的分析、比较以及改进——以基于深度学习的猫狗分类任务为例

腾讯二班 许恹芃

3019244081@tju.edu.cn

摘要

本次研究目的旨在特定任务下实现 Batch Normalization、Layer Normalization、Instance Normalization 和 Group Normalization 等归一化算法,对不同的归一化算法进行参数调优、分析比较各种归一化算法的性能,并在最后尝试对现有的归一化算法进行总结与改进,提出一个对特定任务效果更好的算法。受个人的算力与时间条件约束,在第一版提交的 proposal 的基础上,将“基于深度学习的图像识别分类任务”细化落实为“基于深度学习的猫狗识别二分类任务”,并将“ImageNet 图像数据集”略作加工简化为较小数据集,并发布到 GitHub 上(以上微调在此声明,不再另附新版 proposal)。在该任务背景下,根据经验构建多层卷积神经网络,分别添加不同的归一化算法,针对性的调整超

参数以发挥算法最大的性能,实验结果发现,在该实际问题中 BN 效果最佳,GN 效果稍逊一筹,IN 效果次之,LN 效果最差。并且可以考虑在 batch size 更小的时候换用 GN 以获得最佳效果。在验证已有的四种归一化算法后,基于已有算法,作者提出了一种新的归一化改进算法——全归一化(FN),并通过不同方式进行了实现,形成了 FNBB 和 FNBL 两种算法,分别适于精准识别和快速识别两个不同任务领域,与已有算法进行效果比较,取得了不错的成果。

关键词: 卷积神经网络, 归一化算法, 二分类, 全归一化(FN), FNBB, FNBL

一、引言

1.1 归一化

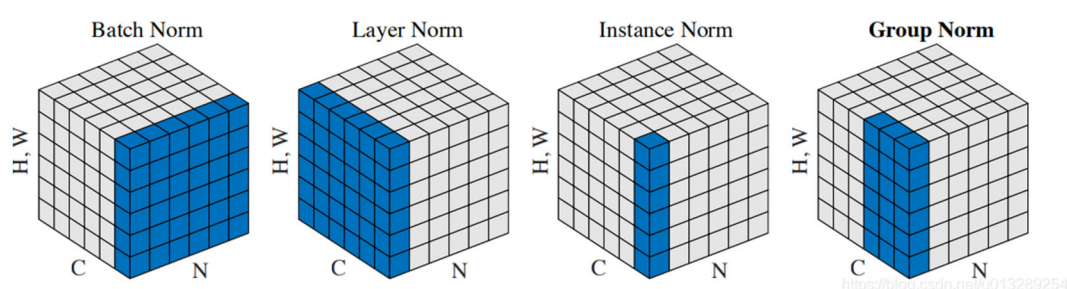
归一化是要把需要的数据经过处理后,通过某种算法限制在一定的范

围内，如在特征图分布不变的条件下设定其均值为 0 且标准差为 1。在统计学中，归一化的具体作用是归纳统一样本的统计分布性。在神经网络中，可以便于后续数据处理，加快收敛速度。

1.2 课题意义

由于每层网络的输入数据在不断

的变化，且每一批次训练数据的分布都很有可能不一样，这会导致不容易找到合适的平衡点，最终使得构建的神经网络模型梯度消失或爆炸。而对数据进行归一化处理可以在多个分布中找到一个合适的平衡点，使模型更快收敛，加快训练速度，节约计算资源。



而由于各种归一化算法实现原理与方式的不同，在处理相同问题时拥有不同的表现（例如，Instance Normalization 孤立了各通道之间的相关性，因此常用于风格化迁移而不适用于其他任务）。且对于不同的参数变化，也有着截然不同的表现（例如，batch size 越大 Batch Normalization 的表现越好，而其他算法则不受 batch size 大小影响）。

因此，对于不同的问题，需要研究适合它的归一化算法，对于不同的归一化算法，需要研究适合它的超参数。

1.3 研究进展与存在的问题

第一个归一化算法 LRN 最早出现

在 2012 年的 AlexNet 中，其主要思想是：借鉴“侧抑制”的思想实现局部神经元抑制，即使得局部的神经元产生竞争机制，使其中相应值较大的将变得更大，响应值较小的将变得更小。2015 年 2 月的 Inception V2 提出了 Batch Normalization，后续在不同的模型上有各种改进。深度学习尤其是在 CV 上都需要对数据归一化，因为深度神经网络主要是为了学习训练数据的分布，并在测试集上达到很好的泛化效果，但是如果每个 batch 输入的数据都具有不同的分布，显然会给网络训练带来困难。另一方面，数据经过一层网络计算后，其数据分布也会发生变化，这会给下一层网络学

习带来困难，google 将此现象成为 Internal Covariate Shift，并提出了 Batch Normalization (2015 年)，解决这个分布变化的问题。而在这之前的解决方案就是使用较小的学习率，和小心的初始化参数，对数据做白化处理，但是显然治标不治本。

由于 BN 实际使用时需要计算并且保存某一层神经网络 batch 的均值和方差等统计信息，对于对一个固定深度的前向神经网络很方便，但不适用于 RNN，且 BN 高度依赖于 mini-batch 的大小，于是便有了 2016 年 7 月的 Layer Normalization 用来填补 RNN 的归一化空白，减少了对 Batch Size 和内存的依赖以及减少归一化所需时间，但在 CNN 时效果普遍不如 BN。2016 年 7 月的 Instance Normalization 操作与 LN 类似。在论文中，作者指出在图像风格迁移任务中，生成式模型计算得到的 Feature Map 的各个 Channel 的均值与方差将影响到所生成图像的风格，IN 应运而生。2018 年 3 月提出的 Group Normalization 与 BN 相比，不再依赖 Batch Size，计算成本可由超参数进行调节，并且可以看作是 LN 和 IN 的一般化。

1.4 技术报告主要贡献总结

(1) 基于猫狗识别二分类任务分

别完成 Batch Normalization、Layer Normalization、Instance Normalization 和 Group Normalization 算法的实现并调整超参数。

(2) 分析比较各归一化算法性能并给出合理分析解释。

(3) 提出一种对现有归一化算法改进的方案并实现。

二、 研究方法

2.1 思路

(1) 实现不带有归一化算法的一般图像识别分类功能，进行参数调优并测试记录模型性能。

(2) 将 Batch Normalization (BN)、Layer Normalization (LN)、Instance Normalization (IN) 和 Group Normalization (GN) 这四种现有的常用归一化算法应用到已实现的图像识别分类网络，分别进行参数调优并测试记录模型性能。

(3) 将上述五种测试性能结果进行分析比较，给出合理解释。根据分析结论总结各算法优劣，改进现有算法并提出一种新的归一化算法 FN，分析该新算法的优越性和泛化性。

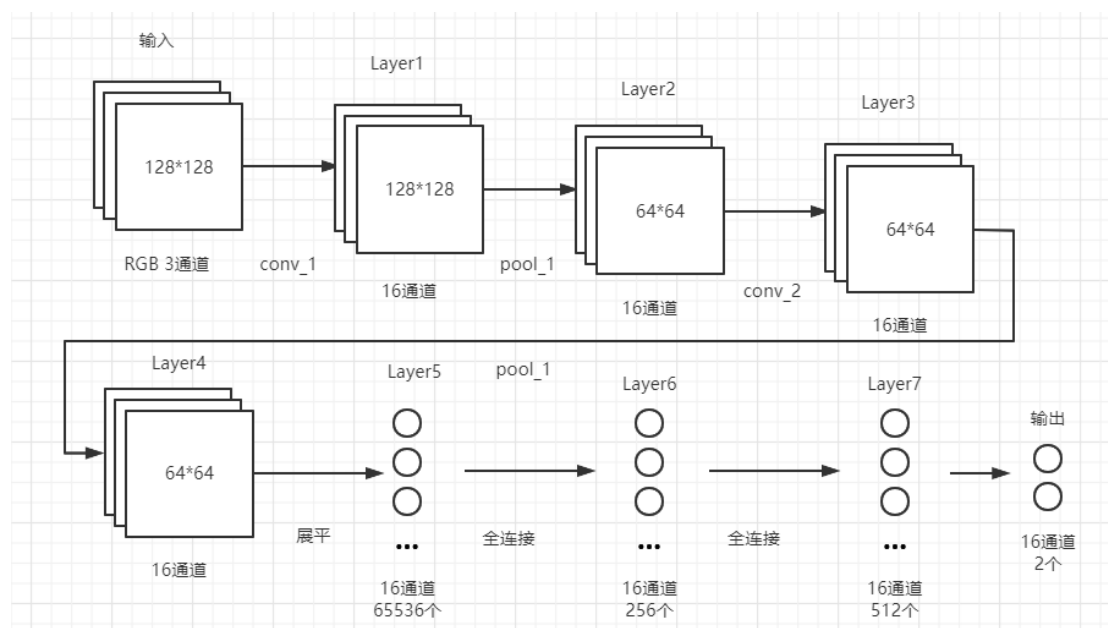
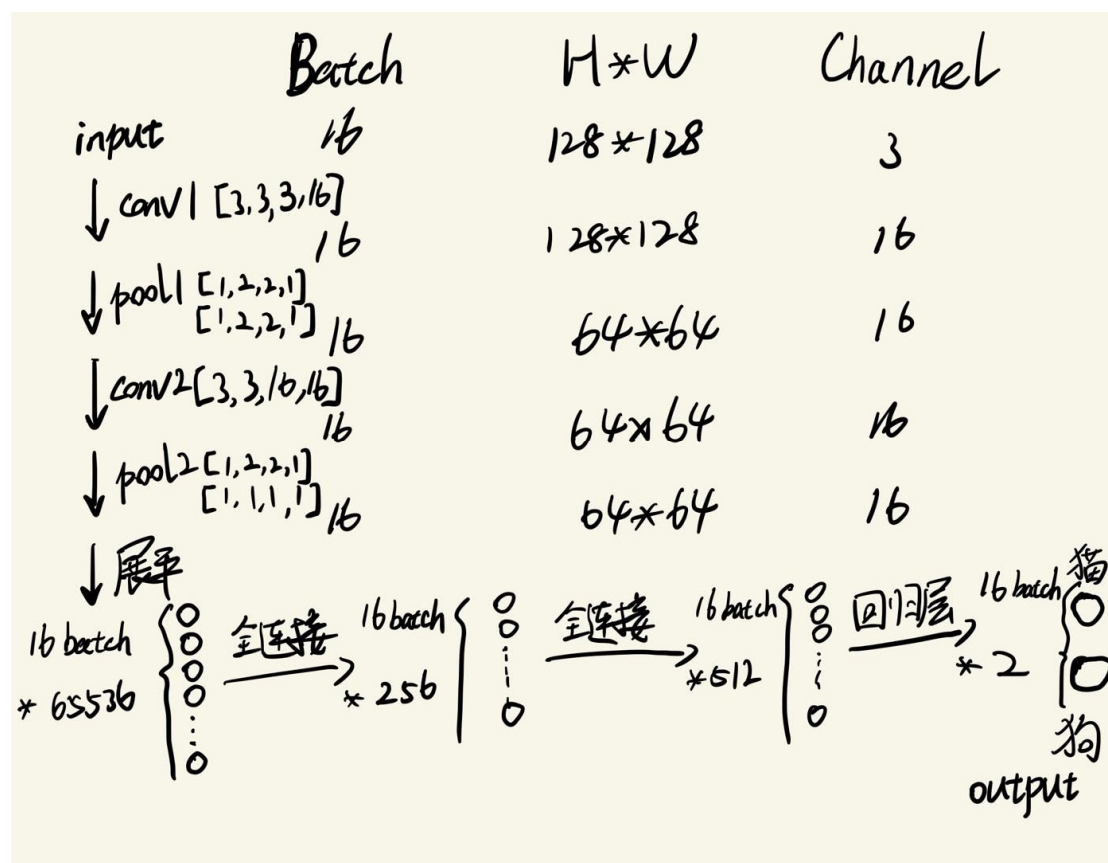
(4) 实现该新算法 FN，并应用到已实现的图像识别分类网络，进行参

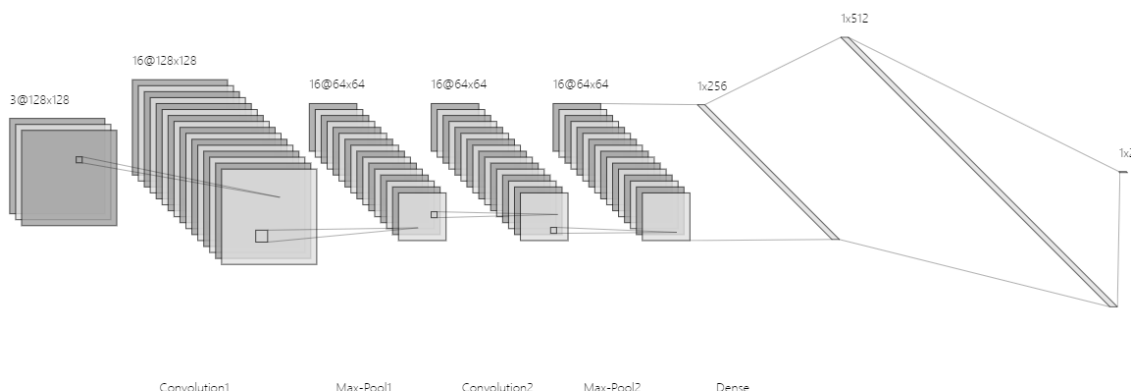
数调优并测试记录模型性能。

新算法的优越性和泛化性。

(5) 与上述五种测试性能结果进行分析比较, 给出合理解释。验证该

2.2 网络架构





2.3 优化原则

将归一化优化添加到每一个卷积层与线性整流层中间，除去卷积量纲的干扰并且重新调整数据分布，使其适应 ReLU 函数，加速梯度下降的求解速度。

三、 实验与结果分析

3.1 实验数据集

受个人的算力与时间条件约束，根据本研究需要，将来源于网络的猫狗图进行标注，加工简化成为本次研究的数据集，并发布到 GitHub 上。
(<https://github.com/NaCl-fish/cat-or-dog--dataset>)。

3.2 实验平台

由于课程并没有提供算力支持，综合各种考量，决定使用本机进行实验。

设备规格

设备名称	DESKTOP-
处理器	AMD Ryzen 5 1600 Six-Core Processor 3.20 GHz
机带 RAM	16.0 GB
设备 ID	7668440D-901C-4D74-A703-
产品 ID	00326-40000-00000-
系统类型	64 位操作系统, 基于 x64 的处理器
笔和触控	没有可用于此显示器的笔或触控输入

Windows 规格

版本	Windows 10 家庭中文版
版本号	21H2
安装日期	2021/3/13
操作系统内部版本	19044.1466
体验	Windows Feature Experience Pack 120.2212.3920.0

期间软件部分代码使用的是 Pycharm 和 python3.8，运行环境在 Anaconda，框架使用 tensorflow2.0，此外还引用了 GitHub 社区实现的 keras-contrib 包
(<https://github.com/keras-team/keras-contrib>)。

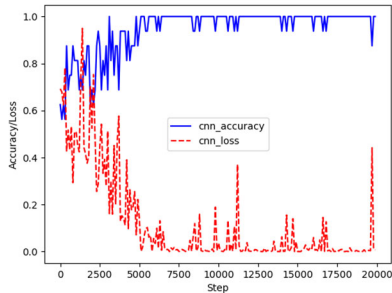
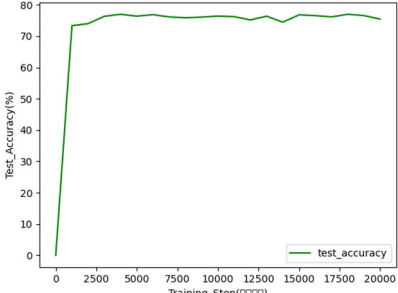
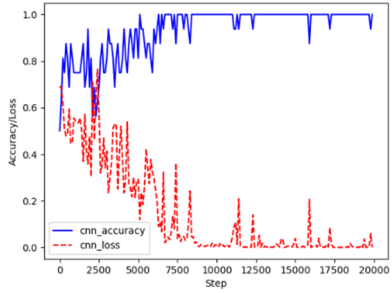
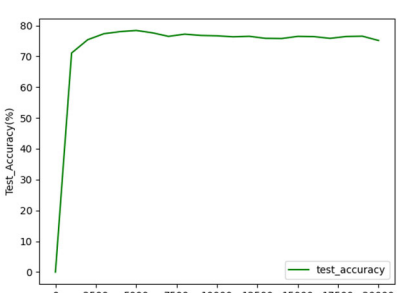
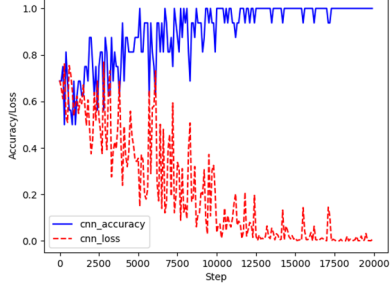
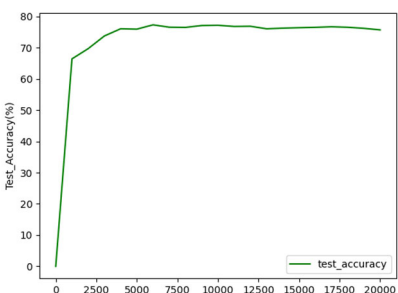
3.3 实验内容

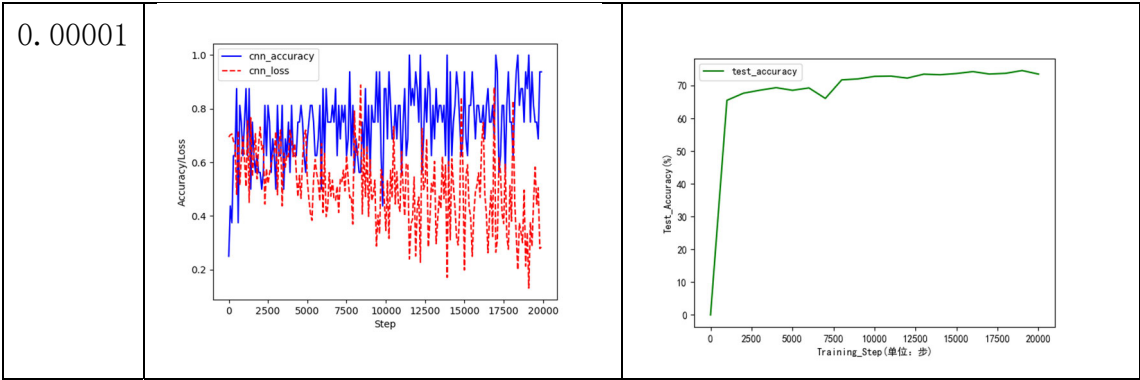
3.3.1 无归一化算法实现猫狗二

分类识别

不加归一化算法直接实现该任务，对学习率和学习步数等参数进行超参数调参（本研究的学习步数不同于训练数，而是训练 batch 数，训练数是学习步数 * batch size）。为方便统计作图，研究将以学习步数为横坐标，

每一千步为一个数据点，共 20000 步。对 0.001、0.0001、0.00005 和 0.00001 这四个不同学习率取值进行比较。测试结果如下。（此处的训练 Accuracy&Loss 是当前训练 batch 的平均值，与之前 batch 的值不做平均）。

学习率	训练 Accuracy&Loss	测试 Accuracy
0.001		
0.0001		
0.00005		



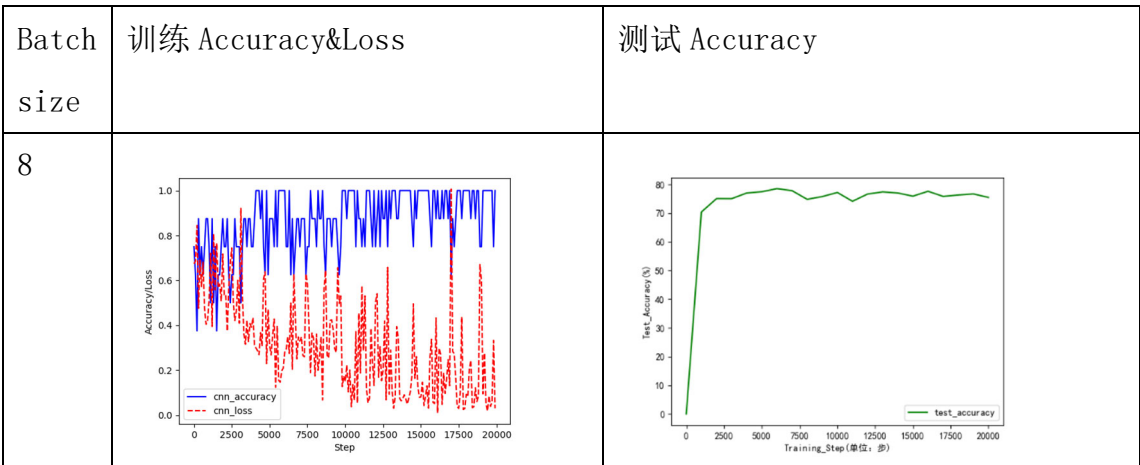
当学习率设置的过小时，收敛过程将变得十分缓慢。而当学习率设置的过大时，梯度可能会在最小值附近来回震荡，甚至可能无法收敛。

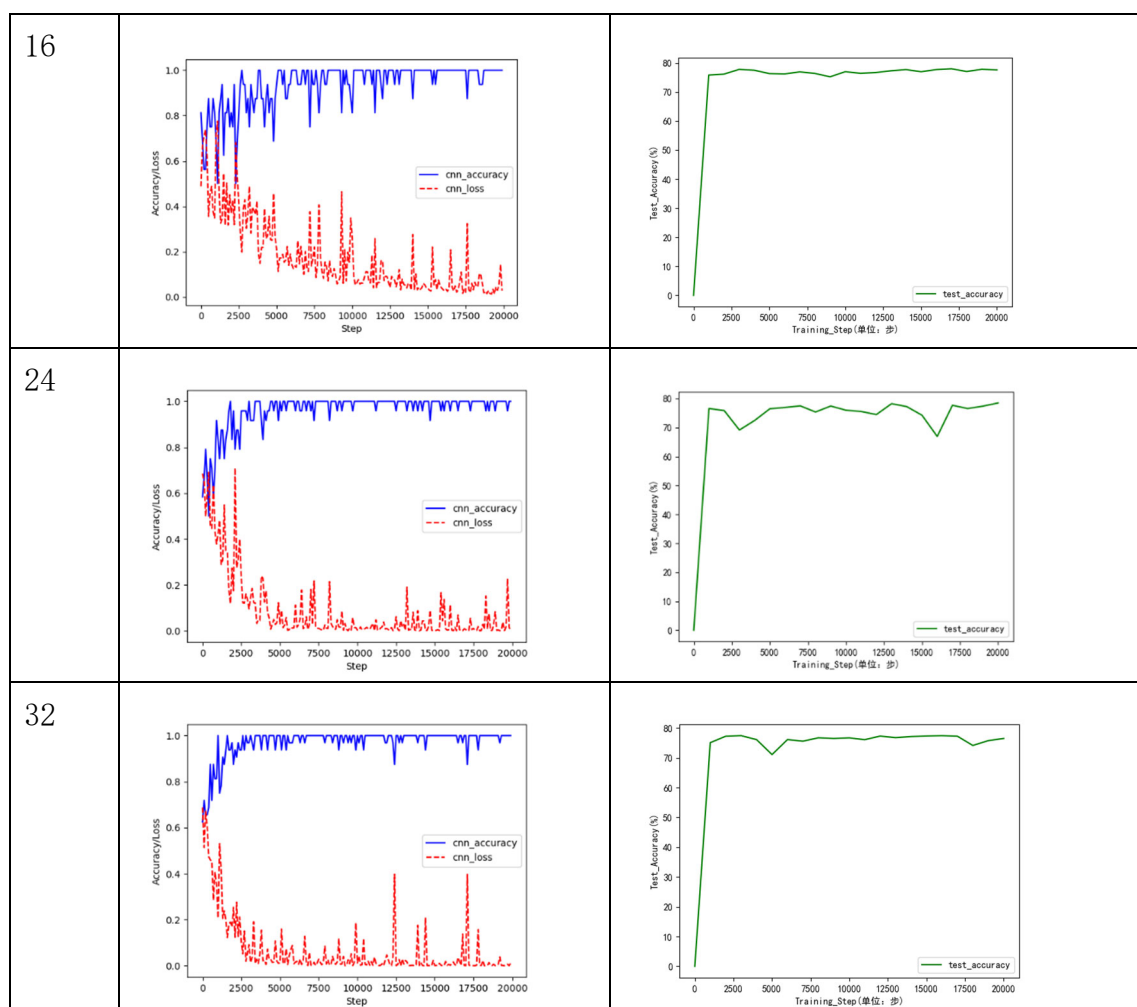
通过结果我们发现当学习率为偏大的 0.001 时，训练准确率和损失率在 5000-10000 与 15000-20000 之间有着近乎相同的震荡，而当学习率为偏小的 0.00005 时，可以看出在相同训练步数下较 0.0001 进度放缓，当学习率为更小的 0.00001 时，由于收敛极其缓慢，20000 的超大训练步数也已很难发现其趋势。以上结果均符合理论预期，在该任务中 0.0001 的学习率更加合适，收敛速度快且测试准确率较

高。后续的模型将继续使用 0.0001 作为统一的学习率，便于比较。

3.3.2 Batch Normalization

即在 batch 上，对 NHW 做归一化。在上面的代码中加入 Batch Normalization 归一化算法实现该任务，固定学习率为 0.0001，对 **batch size** 和**学习步数**等参数进行超参数调参。为方便统计作图，研究将以学习步数为横坐标，每一千步为一个数据点，共 20000 步。对 8、16、24 和 32 四个不同 batch size 取值进行比较。测试结果如下。（此处的训练 Accuracy&Loss 是当前训练 batch 的平均值，与之前 batch 的值不做平均）。





当 batch size 设置的过小时，需要消耗的训练步数将更多。而当 batch size 设置的更大时，Batch Normalization 中一次操作的均值方差更接近于整个网络的均值方差，可以加快网络收敛速度。

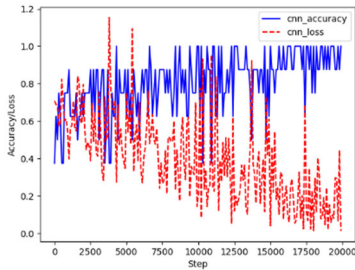
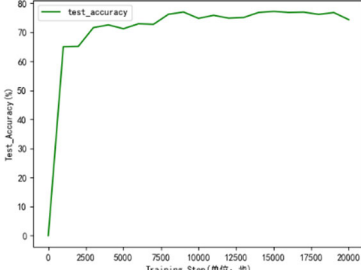
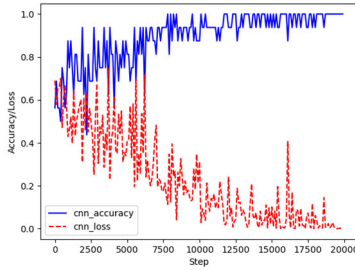
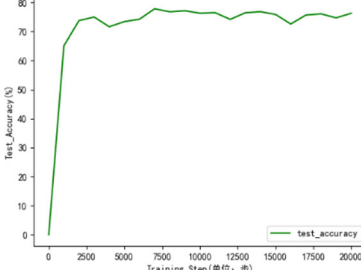
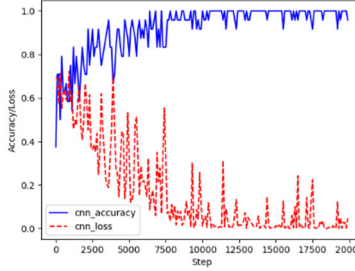
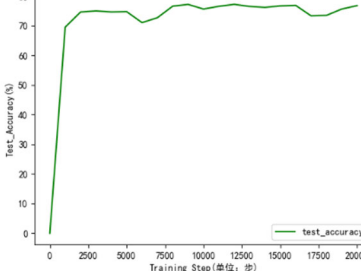
通过结果我们发现当 batch size 为偏小的 8 时，即便是 20000 的超大训练步数也已很难使其收敛。而当 batch size 逐渐增大时，训练准确率和损失率收敛速度逐渐增大，近似成正比关系，但不可忽视的问题是，测

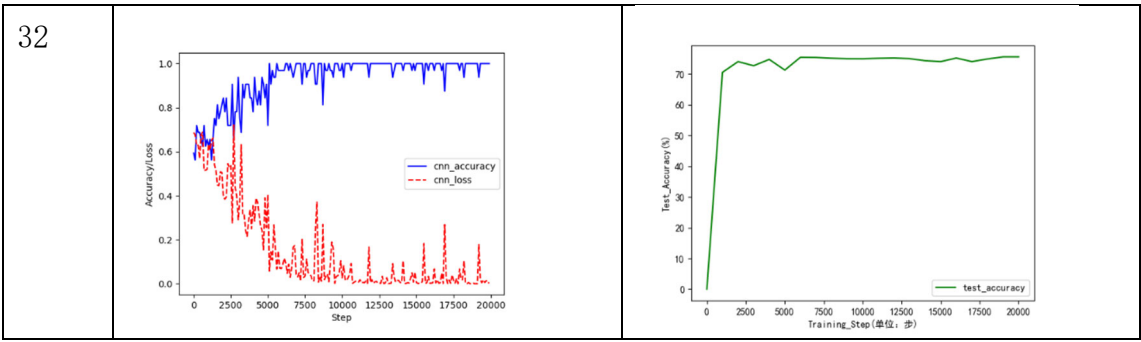
试准确率方面，当 batch size 为 24 和 32 时，分别在 16000 步和 5000 步处出现较大抖动，根据理论推测，可能是 $\text{batch size} \times \text{学习步数}$ 过大，导致的和训练集过拟合，所以在测试集上表现不佳。以上结果均符合理论预期，在该任务中 batch size 设置为 16 更加适合该网络的规模和数据集清晰度，收敛速度较快且过拟合不会出现得过早，可以更准确地找到合适的学习步数。后续模型将继续使用 16 作为统一的 batch size，便于比较。

3.3.3 Layer Normalization

即在通道方向上,对 CHW 归一化。
在上面的代码中换用 Layer Normalization 归一化算法实现该任务,根据前两个实验的经验固定学习率为0.0001,对 **batch size** 和**学习步数**等参数进行超参数调参。为方便统

计作图,研究将以学习步数为横坐标,每一千步为一个数据点,共 20000 步。
对 8、16、24 和 32 这四个不同 batch size 取值进行比较。测试结果如下。
(此处的训练 Accuracy&Loss 是当前训练 batch 的平均值,与之前 batch 的值不做平均)。

Batch size	训练 Accuracy&Loss	测试 Accuracy
8		
16		
24		



batch size 的大小与 Layer Normalization 性能没有直接关系,且 Layer Normalization 相较 Batch Normalization 更适合 RNN 的归一化而不是本次研究任务中的 CNN。需要注意的是本研究的学习步数不同于训练数,而是训练 batch 数,训练数是学习步数 * batch size。

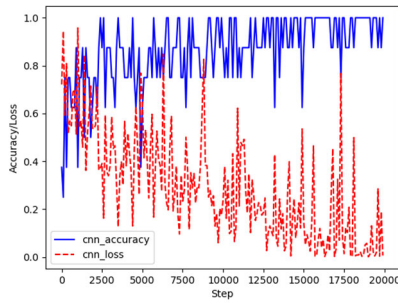
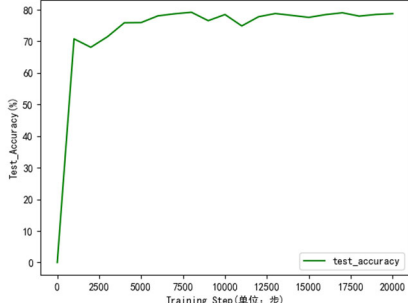
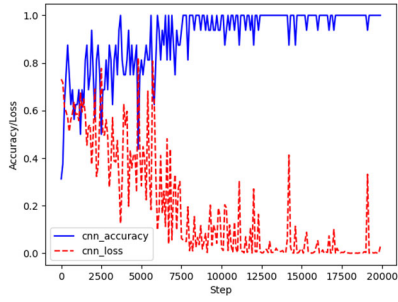
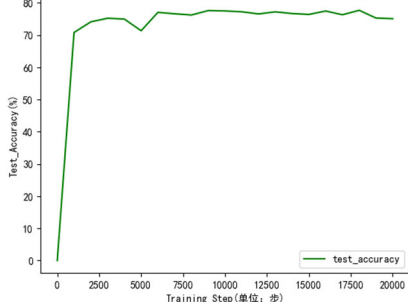
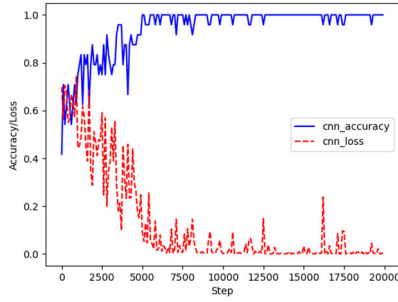
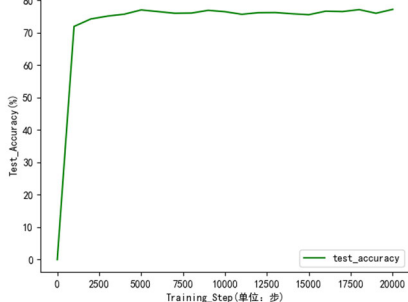
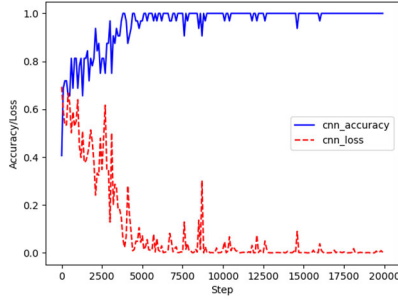
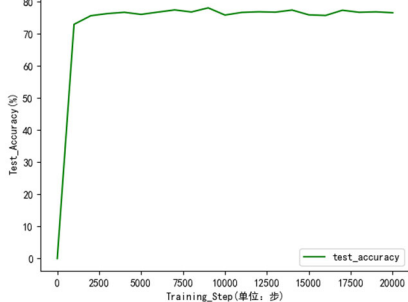
通过结果我们发现当训练数相同(即学习步数 * batch size 相同)时,模型的收敛进度大致相同。而且我们不难看到使用 Layer Normalization 后甚至在测试集上的效果不如不用归一化算法准确率高或抖动更剧烈。以上结果符合实验预期。

	无 norm	LN	比较
16 batch			LN 抖动剧烈

3.3.4 Instance Normalization

即在图像像素上,对 HW 归一化。在上面的代码中换用 Instance Normalization 归一化算法实现该任务,根据前两个实验的经验固定学习率为 0.0001,对 **batch size** 和**学习步数**等参数进行超参数调参。为方便统

计作图,研究将以学习步数为横坐标,每一千步为一个数据点,共 20000 步。对 8、16、24 和 32 这四个不同 batch size 取值进行比较。测试结果如下。(此处的训练 Accuracy&Loss 是当前训练 batch 的平均值,与之前 batch 的值不做平均)。

Batch size	训练 Accuracy&Loss	测试 Accuracy
8		
16		
24		
32		

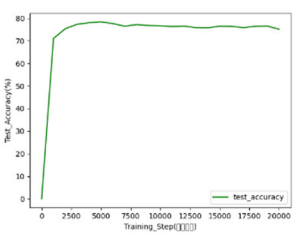
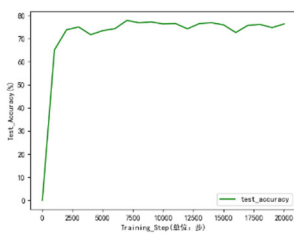
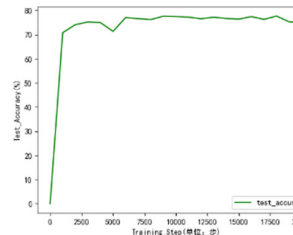
batch size 的大小与 Instance Normalization 性能没有直接关系, 且

Instance Normalization 相较于 Batch Normalization 主要用于风格迁移的

归一化而不是本次研究任务中的 CNN。需要注意的是本研究的学习步数不同于训练数，而是训练 batch 数，训练数是学习步数 * batch size。

通过结果我们发现当训练数相同（即学习步数 * batch size 相同）时，

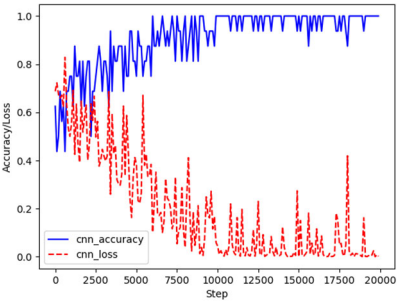
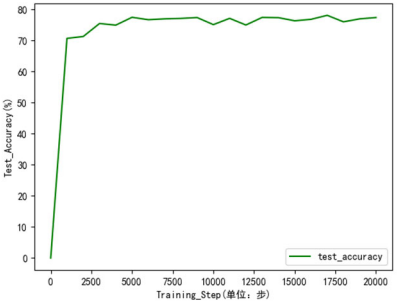
模型的收敛进度大致相同。而且我们不难看到使用 Instance Normalization 后虽然在测试集上的效果不如不用归一化算法准确率高或抖动更剧烈，但比较接近且相较 LN 更好一些。以上结果符合实验预期。

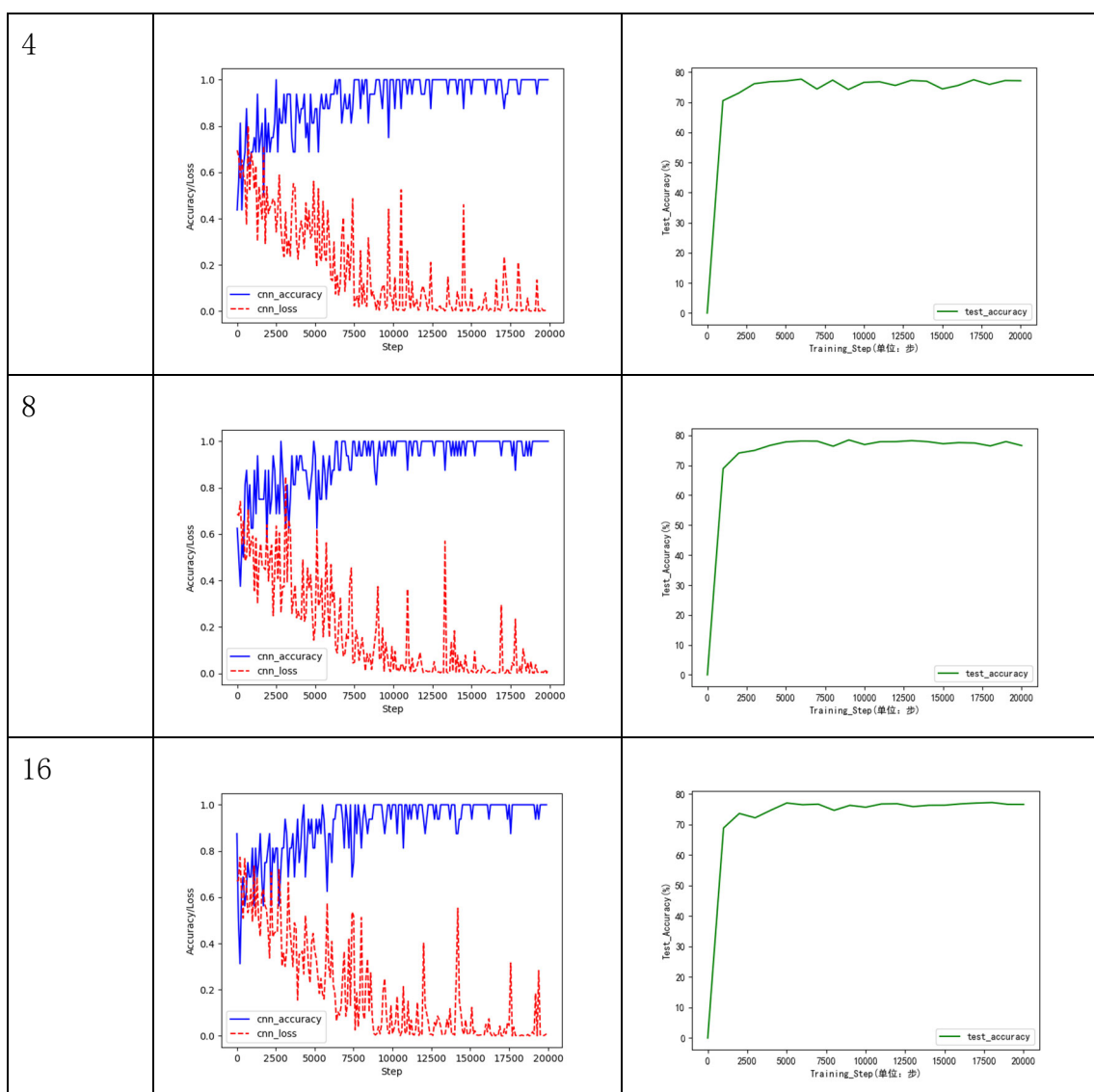
	无 norm	LN	IN
16 batch			

3.3.5 Group Normalization

即将通道分组，然后在 CHW 上再做归一化。在上面的代码中换用 Group Normalization 归一化算法实现该任务，根据前两个实验的经验固定学习率为 0.0001 以及 batch size 为 16，对分组数和学习步数等参数进行超参

数调参。为方便统计作图，研究将以学习步数为横坐标，每一千步为一个数据点，共 20000 步。对 2、4、8 和 16 这四个不同分组数取值进行比较。测试结果如下。（此处的训练 Accuracy&Loss是当前训练batch的平均值，与之前 batch 的值不做平均）。

分组数	训练 Accuracy&Loss	测试 Accuracy
2		

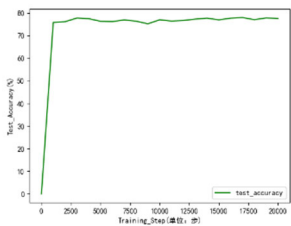
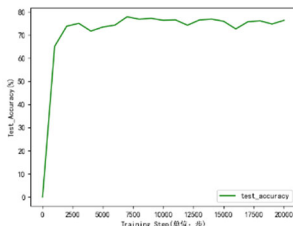
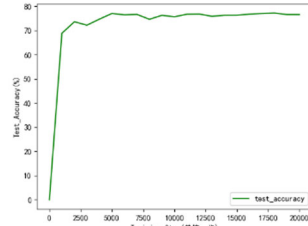


Group Normalization 主要是针对 Batch Normalization 对小 batch size 效果差,GN 将 channel 方向分 group, 然后每个 group 内做归一化, 算 $(C//G)HW$ 的均值, 这样与 batch size 无关, 不受其约束。

通过结果我们发现当分组数越小即每个分组越大时, 训练准确率和损失率收敛趋势越明显, 当分组数为 2、4 和 16 时, 测试准确率抖动幅度较大

且抖动频发。而当分组数为 8 时, 训练准确率和损失率收敛较好, 且基本没有抖动, 比较稳定。

与 batch size 为 16 的 BN 和 LN (理论上等价于 1 分组的 GN) 相比较, 效果优于 LN, 接近但还稍逊于 BN 的效果。以上结果符合实验预期。并且可以考虑在 mini-batch 小于 16 的时候将 BN 换成 GN 获得最佳效果。

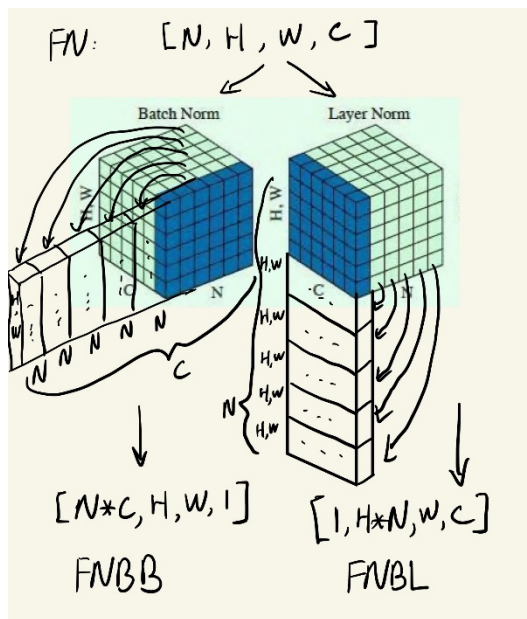
	BN	LN (理论上等价于 1 分组的 GN)	GN
16 batch 或 8 分组			

3.3.5 改进算法 —— Full Normalization

即全归一化。受启发于 BN 和 LN，我们发现，两种算法在特征图基础上，要么取全 batch 而分离各通道，要么取全通道而分离 batch。根据之前的实验结果，在二分类识别任务中，BN 的效果最好，但是，将各通道分别归一化将导致色彩搭配信息的丢失，而 LN 虽保持了色彩分布，但因为单个 batch 做归一化，所以不能很好的反映整体数据集特征图的分布特征。如果将特征图、各 batch 和各通道综合在一起，对于有色图的二分类识别任务将会有不错的效果。在上面的代码中换用 Full Normalization 归一化算法实现该任务，根据前两个实验的经验固定学习率为 0.0001，对 **batch size** 和 **学习步数** 等参数进行超参数调参。为方便统计作图，研究将以学习步数为横坐标，每一千步为一个数据点，共 20000 步。对 8、16、24 和 32 这四个

不同 batch size 取值进行比较。

Full Normalization 的具体实现方法如下。由于是由已有算法的改进与延伸，且受限于研究时间限制因素与算力因素，FN 将借助 BN 和 LN 的实现来实现，也由此引申出了两种实现方法。相同点是，对于神经元展平之后的归一化，皆继续使用 BN 实现。不同点是，对于神经元展平之前的部分，由于要归一化的神经元相较 BN 和 LN 多出一维，所以要先进行一次降维，再经过 BN 或 LN，最后再升维回原本的形状。借助 BN 实现和借助 LN 实现分别对应不同的降维方式，含义也不一样，将两种方式分别命名为 Full Normalization based on BN (FNBB) 和 Full Normalization based on LN (FNBL)。



```
dim = 128
reshape = tf.reshape(pre_activation, shape=[batch_size * 16, dim, dim, 1])

bn1 = tf.layers.batch_normalization(reshape, training=is_training)

bn1 = tf.reshape(bn1, shape=[batch_size, dim, dim, 16])
```

```
dim = 128
reshape = tf.reshape(pre_activation, shape=[1, batch_size * dim, dim, 16])

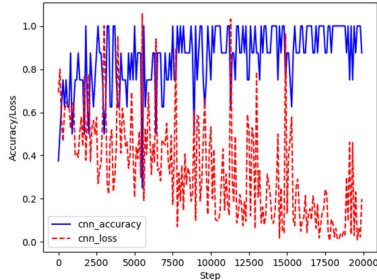
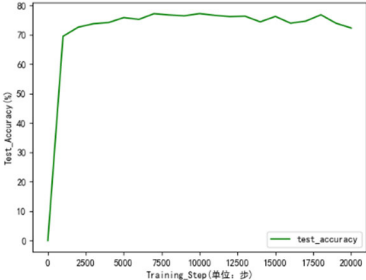
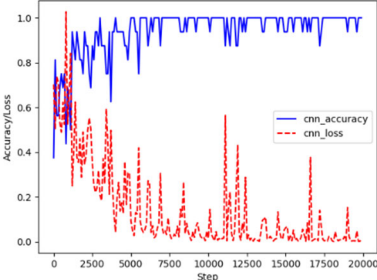
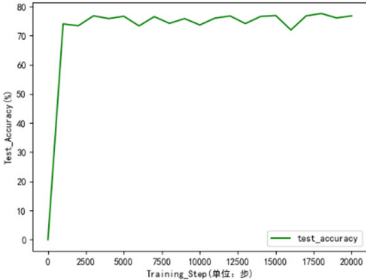
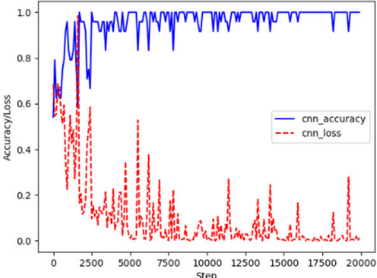
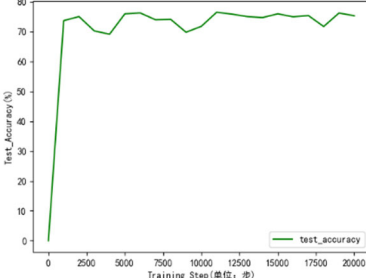
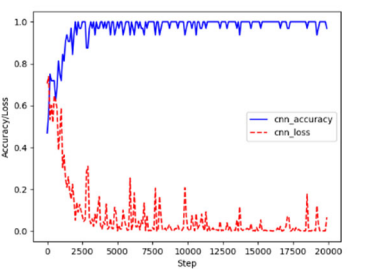
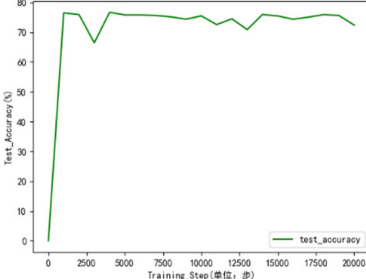
ln1 = tf2.keras.layers.LayerNormalization(trainable=is_training)(reshape)

ln1 = tf.reshape(ln1, shape=[batch_size, dim, dim, 16])
```

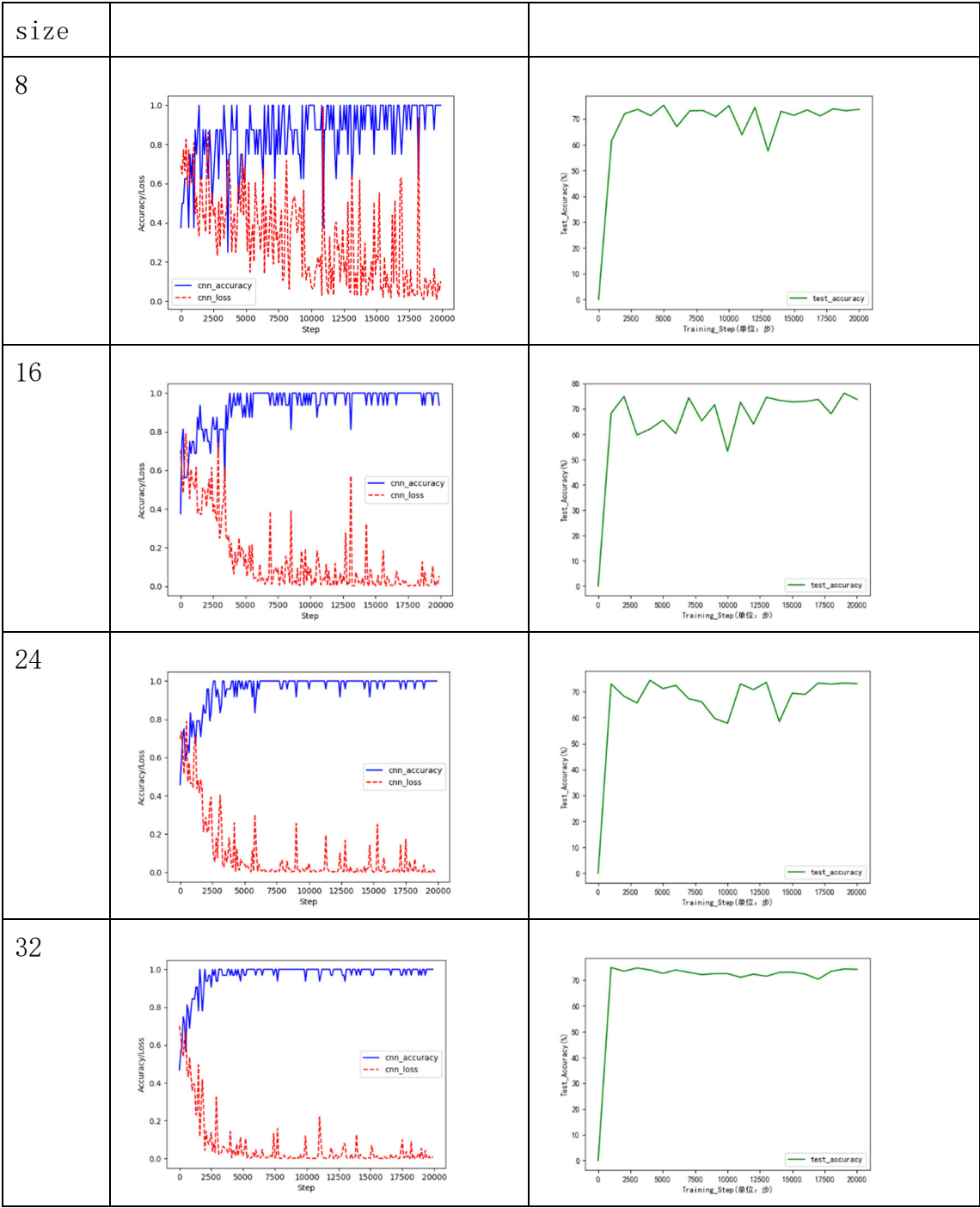
FNBB 最先被提出，因为它最直观也最符合人的理解。特征图维度不变，将通道叠放到 batch 维上归一化，在理论上也丝毫不影响效果，堪称是完美的转化！但在实际实验过程中由于向量变换时维度调整过大，时间消耗过长，于是提出了第二种方法 FNBL。将变动最大的两个维度，放在第一维

和第二维，可以保证后两维基本形状不变，理论上大大降低了向量变换损耗的时间，在实际训练过程中也得到了验证，FNBL 耗时仅为 FNBB 的三分之一不到，但在特征图含以上存在缺失，将统一 batch 的其他图在 H 维度上拼接归一化，似乎含义不是非常理想，但也不无道理。

测试结果如下。（此处的训练均值，与之前 batch 的值不做平均）。
Accuracy&Loss 是当前训练 batch 的平

(FNBB) Batch size	训练 Accuracy&Loss	测试 Accuracy
8		
16		
24		
32		

(FNBL) Batch	训练 Accuracy&Loss	测试 Accuracy
-----------------	------------------	-------------



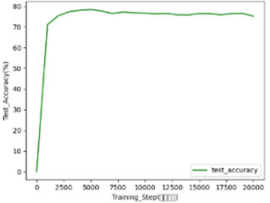
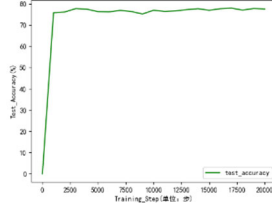
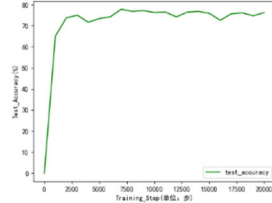
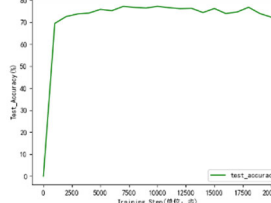
通过结果我们发现，在相同训练步数和 batch size 下，虽然 FNBL 算法训练时间远远小于 FNBB，但是，在测试准确率上，FNBL 算法远不如 FNBB 算法的效果，平均准确率低且抖动剧烈。因此，从算法整体而言，FNBB 更适合大算力的精准识别任务，FNBL 更适合

小组织或个人的快速识别任务。

FNBB 算法平均测试准确率较高，随着 batch size 的升高训练收敛速度升高，但测试准确率抖动也越剧烈，最佳取值为 8 batch、学习步数约 10000，这样的取值更适合该研究的模型。

FNBL 算法训练时间更短，与 FNBB 相似，随着 batch size 的升高训练收敛速度升高，当 batch size 取 8、16 和 24 时，随着 batch size 的升高，测试准确率抖动也越剧烈，最佳取值为 32 batch、学习步数约 3000，这样的取值更适合该研究的模型。

用于大算力精准预测的 FNBB 算法与无归一化算法和应用 BN 算法以及 LN 算法的模型相比，在学习约 10000 步时，使用了更少的训练数（即学习步数 * batch size），但测试准确率更高更稳定，效果更好。

	无 norm (16 batch)	BN (16 batch)	LN (16 batch)	FNBB (8 batch)
测试准确率				

四、 结论

此次研究基于猫狗识别二分类任务，应用消融实验和对照实验的方法，系统全面地对 BN、LN、IN 和 GN 四种已有归一化算法进行了比较验证和总结分析，实验结果符合理论预期。此外，本技术报告开创性地提出了 FNBB 和 FNBL 两种新的归一化改进算法，同已有算法进行了详细的比较，证明了各自地优越性和适用范围，验证了两种新算法的应用价值。

此次研究受限于时间限制和算力不足的影响，采用的猫狗二分类问题背景比较简单，使用的神经网络规模比较小，参数调整范围比较小且与多

任务、多模型、多整流函数匹配试验较少。

归一化算法的不断演进迭代推动着神经网络与深度学习的发展，对不同任务、不同场景研究并使用不同的适合的归一化算法可以节约算力、获得更好的学习效果。日后的研究应围绕特定热门任务场景，与不同模型和功能联系的特殊归一化算法研究。

最后，感谢天津大学王旗龙老师在课堂上悉心的讲授以及胡琪瑶、庄旭两位学长学姐耐心的答疑，这次研究的完成与完善离不开他们的帮助。

五、 参考文献

[1] Sergey Ioffe, Christian Szegedy.
Batch Normalization: Accelerating Deep

Network Training by Reducing Internal Covariate Shift, 2015.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton. Layer Normalization, 2016.

[3] Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky. Instance Normalization: The Missing Ingredient for Fast Stylization, 2017.

[4] Yuxin Wu, Kaiming He. Group Normalization, 2018.

[5] Squadrick and gabrieldemarmiesse. Keras-contrib.
<https://github.com/keras-team/keras-contrib>, 2019.

[6] 时光碎了天. 深度学习中的五种归一化 (BN、LN、IN、GN 和 SN) 方法简介.
<https://blog.csdn.net/u013289254/article/details/99690730>, 2019.

[7] Solomon1588. 归一化函数 normalize 详解.
<https://blog.csdn.net/solomon1558/article/details/44689611>, 2015.

[8] 公众号机器学习与生成对抗网络. 归一化 Normalization 的发展历程.
<https://blog.csdn.net/lgzlgz3102/article/details/111306612>, 2020.

[9] UpCoderXH. LRN, BN, LN, IN, GN.
<https://blog.csdn.net/liangdong2014/article/details/88038360>, 2019.

[10] -牧野-. tensorflow 中协调器 tf.train.Coordinator 和入队线程启动器 tf.train.start_queue_runners.

<https://blog.csdn.net/dcrmg/article/details/79780331>, 2018.

[11] CrazyVertigo. 【深度学习技术】LRN 局部响应归一化.
<https://blog.csdn.net/hduxiejun/article/details/70570086>, 2017.

[12] andrewsher. 神经网络中的几种归一化方式.
<https://zhuanlan.zhihu.com/p/137995496>, 2020.

[13] zjrn. 【TensorFlow】关于 tf.nn.sparse_softmax_cross_entropy_with_logits().
<https://blog.csdn.net/ZJRN1027/article/details/80199248>, 2018.

[14] 惜曦. keras 的 InstanceNormalization 的使用.
https://blog.csdn.net/qg_41033241/article/details/104076010, 2020.

[15] Vico_Men. 深度学习第二课 改善深层神经网络: 超参数调试、正则化以及优化 第三周超参数调试+Batch normalization 笔记和作业.
https://blog.csdn.net/qg_28031525/article/details/78983447, 2018.

[16] cat-or-dog-dataset. NaCl-fish.
<https://github.com/NaCl-fish/cat-or-dog-dataset>, 2022.