Batch Normalization论文翻译——中英文对照

| 2049

Batch Normalization论文翻译——中英文对照

文章作者：Tyan
博客：noahsnail.com | CSDN | 简书

声明：作者翻译论文仅为学习，如有侵权请联系作者删除博文，谢谢！

翻译论文汇总：https://github.com/SnailTyan/deep-learning-papers-translation

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Abstract

Training Deep Neural Networks is complicated by the fact that the distribution of each layer's inputs changes during training, as the parameters of the previous layers change. This slows down the training by requiring lower learning rates and careful parameter initialization, and makes it notoriously hard to train models with saturating nonlinearities. We refer to this phenomenon as *internal covariate shift*, and address the problem by normalizing layer inputs. Our method draws its strength from making normalization a part of the model architecture and performing the normalization *for each training mini-batch*. Batch Normalization allows us to use much higher learning rates and be less careful about initialization. It also acts as a regularizer, in some cases eliminating the need for Dropout. Applied to a state-of-the-art image classification model, Batch Normalization achieves the same accuracy with 14 times fewer training steps, and beats the original model by a significant margin. Using an ensemble of batch-normalized networks, we improve upon the best published result on ImageNet classification: reaching `4.9% top-5` validation error (and `4.8%` test error), exceeding the accuracy of human raters.

摘要

训练深度神经网络的复杂性在于，每层输入的分布在训练过程中会发生变化，因为前面的层的参数会发生变化。通过要求较低的学习率和仔细的参数初始化减慢了训练，并且使具有饱和非线性的模型训练起来非常困难。我们将这种现象称为*内部协变量转移*，并通过标准化层输入来解决这个问题。我们的方法力图使标准化成为模型架构的一部分，并为*每个训练小批量数据*执行标准化。批标准化使我们能够使用更高的学习率，并且不用太注意初始化。它也作为一个正则化项，在某些情况下不需要Dropout。将批量标准化应用到最先进的图像分类模型上，批标准化在取得相同的精度的情况下，减少了14倍的训练步骤，并以显著的差距击败了原始模型。使用批标准化网络的组合，我们改进了在ImageNet分类上公布的最佳结果：达到了`4.9% top-5`的验证误差（和`4.8%`测试误差），超过了人类评估者的准确性。

1. Introduction

Deep learning has dramatically advanced the state of the art in vision, speech, and many other areas. Stochastic gradient descent (SGD) has proved to be an effective way of training deep networks, and SGD variants such as momentum (Sutskever et al., 2013) and Adagrad (Duchi et al., 2011) have been used to achieve state of the art performance. SGD optimizes the parameters Θ of the network, so as to minimize the loss

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

where $x_{1\ldots N}$ is the training data set. With SGD, the training proceeds in steps, and at each step we consider a *mini-batch* $x_{1\ldots m}$ of size m. The mini-batch is used to approximate the gradient of the loss function with respect to the parameters, by computing $\frac{1}{m}\sum_{i=1}^{m}\frac{\partial \ell(x_i,\Theta)}{\partial \Theta}$. Using mini-batches of examples, as opposed to one example at a time, is helpful in several ways. First, the gradient of the loss over a mini-batch is an estimate of the gradient over the training set, whose quality improves as the batch size increases. Second, computation over a batch can be much more efficient than m computations for individual examples, due to the parallelism afforded by the modern computing platforms.

1. 引言

深度学习在视觉、语音等诸多方面显著提高了现有技术的水平。随机梯度下降（SGD）已经被证明是训练深度网络的有效方式，并且已经使用诸如动量（Sutskever等，2013）和Adagrad（Duchi等人，2011）等SGD变种取得了最先进的性能。SGD优化网络参数Θ，以最小化损失

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

$x_{1\ldots N}$是训练数据集。使用SGD，训练将逐步进行，在每一步中，我们考虑一个大小为m的*小批量数据* $x_{1\ldots m}$。通过计算$\frac{1}{m}\sum_{i=1}^{m}\frac{\partial \ell(x_i,\Theta)}{\partial \Theta}$，使用小批量数据来近似损失函数关于参数的梯度。使用小批量样本，而不是一次一个样本，在一些方面是有帮助的。首先，小批量数据的梯度损失是训练集上的梯度估计，其质量随

着批量增加而改善。第二，由于现代计算平台提供的并行性，对一个批次的计算比单个样本计算m次效率更高。

While stochastic gradient is simple and effective, it requires careful tuning of the model hyper-parameters, specifically the learning rate used in optimization, as well as the initial values for the model parameters. The training is complicated by the fact that the inputs to each layer are affected by the parameters of all preceding layers —— so that small changes to the network parameters amplify as the network becomes deeper.

虽然随机梯度是简单有效的，但它需要仔细调整模型的超参数，特别是优化中使用的学习速率以及模型参数的初始值。训练的复杂性在于每层的输入受到前面所有层的参数的影响——因此当网络变得更深时，网络参数的微小变化就会被放大。

The change in the distributions of layers' inputs presents a problem because the layers need to continuously adapt to the new distribution. When the input distribution to a learning system changes, it is said to experience *covariate shift* (Shimodaira, 2000). This is typically handled via domain adaptation (Jiang, 2008). However, the notion of covariate shift can be extended beyond the learning system as a whole, to apply to its parts, such as a sub-network or a layer. Consider a network computing

$$\ell = F_2(F_1(u, \Theta_1), \Theta_2)$$

where F_1 and F_2 are arbitrary transformations, and the parameters Θ_1, Θ_2 are to be learned so as to minimize the loss ℓ. Learning Θ_2 can be viewed as if the inputs $x = F_1(u, \Theta_1)$ are fed into the sub-network

$$\ell = F_2(x, \Theta_2).$$

层输入的分布变化是一个问题，因为这些层需要不断适应新的分布。当学习系统的输入分布发生变化时，据说会经历*协变量转移*（Shimodaira，2000）。这通常是通过域适应（Jiang，2008）来处理的。然而，协变量漂移的概念可以扩展到整个学习系统之外，应用到学习系统的一部分，例如子网络或一层。考虑网络计算

$$\ell = F_2(F_1(u, \Theta_1), \Theta_2)$$

F_1和F_2是任意变换，学习参数Θ_1，Θ_2以便最小化损失ℓ。学习Θ_2可以看作输入$x = F_1(u, \Theta_1)$送入到子网络

$$\ell = F_2(x, \Theta_2)。$$

For example, a gradient descent step

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

(for batch size m and learning rate α) is exactly equivalent to that for a stand-alone network F_2 with input x. Therefore, the input distribution properties that make training more efficient —— such as having the same distribution between the training and test data —— apply to training the sub-network as well. As such it is advantageous for the distribution of x to remain fixed over time. Then, Θ_2 does not have to readjust to compensate for the change in the distribution of x.

例如，梯度下降步骤

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

（对于批大小m和学习率α）与输入为x的单独网络F_2完全等价。因此，输入分布特性使训练更有效——例如训练数据和测试数据之间有相同的分布——也适用于训练子网络。因此x的分布在时间上保持固定是有利的。然后，Θ_2不必重新调整来补偿x分布的变化。

Fixed distribution of inputs to a sub-network would have positive consequences for the layers *outside* the sub-network, as well. Consider a layer with a sigmoid activation function $z = g(Wu + b)$ where u is the layer input, the weight matrix W and bias vector b are the layer parameters to be learned, and $g(x) = \frac{1}{1+\exp(-x)}$. As $|x|$ increases, $g'(x)$ tends to zero. This means that for all dimensions of $x = Wu + b$ except those with small absolute values, the gradient flowing down to u will vanish and the model will train slowly. However, since x is affected by W, b and the parameters of all the layers below, changes to those parameters during training will likely move many dimensions of x into the saturated regime of the nonlinearity and slow down the convergence. This effect is amplified as the network depth increases. In practice, the saturation problem and the resulting vanishing gradients are usually addressed by using Rectified Linear Units (Nair & Hinton, 2010) $ReLU(x) = \max(x, 0)$, careful initialization (Bengio & Glorot, 2010; Saxe et al., 2013), and small learning rates. If, however, we could ensure that the distribution of nonlinearity inputs remains more stable as the network trains, then the optimizer would be less likely to get stuck in the saturated regime, and the training would accelerate.

子网络输入的固定分布对于子网络外的层也有积极的影响。考虑一个激活函数为$g(x) = \frac{1}{1+\exp(-x)}$的层，$u$是层输入，权重矩阵$W$和偏置向量$b$是要学习的层参数，$g(x) = \frac{1}{1+\exp(-x)}$。随着$|x|$的增加，$g'(x)$趋向于0。这意味着对于$x = Wu + b$的所有维度，除了那些具有小的绝对值之外，流向$u$的梯度将会消失，模型将缓慢的进行训练。然而，由于$x$受$W, b$和下面所有层的参数的影响，训练期间那些参数的改变可能会将x的许多维度移动到非线性的饱和状态并减慢收敛。这个影响随着网络深度的增加而放大。在实践中，饱和问题和由此产生的梯度消失通常通过使用修正线性单元(Nair & Hinton, 2010) $ReLU(x) = \max(x, 0)$，仔细的初始化(Bengio & Glorot, 2010; Saxe et al., 2013)和小的学习率来解决。然而，如果我们能保证非线性输入的分布在网络训练时保持更稳定，那么优化器将不太可能陷入饱和状态，训练将加速。

We refer to the change in the distributions of internal nodes of a deep network, in the course of training, as *Internal Covariate Shift*. Eliminating it offers a promise of faster training. We propose a new mechanism, which we call *Batch Normalization*, that takes a step towards reducing internal covariate shift, and in doing so dramatically accelerates the training of deep neural nets. It accomplishes this via a normalization step that fixes the means and variances of layer inputs. Batch Normalization also has a beneficial effect on the gradient flow through the network, by reducing the dependence of gradients on the scale of the parameters or of their initial values. This allows us to use much higher learning rates without the risk of divergence. Furthermore, batch normalization regularizes the model and reduces the need for Dropout (Srivastava et al., 2014). Finally, Batch Normalization makes it possible to use saturating nonlinearities by preventing the network from getting stuck in the saturated modes.

我们把训练过程中深度网络内部结点的分布变化称为*内部协变量转移*。消除它可以保证更快的训练。我们提出了一种新的机制，我们称为为*批标准化*，它是减少内部协变量转移的一个步骤，这样做可以显著加速深度神经网络的训练。它通过标准化步骤来实现，标准化步骤修正了层输入的均值和方差。批标准化减少了梯度对参数或它们的初始值尺度上的依赖，对通过网络的梯度流动有有益的影响。这允许我们使用更高的学习率而没有发散的风险。此外，批标准化使模型正则化并减少了对Dropout(Srivastava et al., 2014)的需求。最后，批标准化通过阻止网络陷入饱和模式让使用饱和非线性成为可能。

In Sec. 4.2, we apply Batch Normalization to the best-performing ImageNet classification network, and show that we can match its performance using only 7% of the training steps, and can further exceed its accuracy by a substantial margin. Using an ensemble of such networks trained with Batch Normalization, we achieve the top-5 error rate that improves upon the best known results on ImageNet classification.

在4.2小节，我们将批标准化应用到性能最好的ImageNet分类网络上，并且表明我们可以使用仅7%的训练步骤来匹配其性能，并且可以进一步超过其准确性一大截。通过使用批标准化训练的网络的集合，我们取得了top-5错误率，其改进了ImageNet分类上已知的最佳结果。

2. Towards Reducing Internal Covariate Shift

We define *Internal Covariate Shift* as the change in the distribution of network activations due to the change in network parameters during training. To improve the training, we seek to reduce the internal covariate shift. By fixing the distribution of the layer inputs x as the training progresses, we expect to improve the training speed. It has been long known (LeCun et al., 1998b; Wiesler & Ney, 2011) that the network training converges faster if its inputs are whitened – i.e., linearly transformed to have zero means and unit variances, and decorrelated. As each layer observes the inputs produced by the layers below, it would be advantageous to achieve the same whitening of the inputs of each layer. By whitening the inputs to each layer, we would take a step towards achieving the fixed distributions of inputs that would remove the ill effects of the internal covariate shift.

2. 减少内部协变量转变

由于训练过程中网络参数的变化，我们将*内部协变量转移*定义为网络激活分布的变化。为了改善训练，我们寻求减少内部协变量转移。随着训练的进行，通过固定层输入x的分布，我们期望提高训练速度。众所周知(LeCun et al., 1998b; Wiesler & Ney, 2011)如果对网络的输入进行白化，网络训练将会收敛的更快——即输入线性变换为具有零均值和单位方差，并去相关。当每一层观察下面的层产生的输入时，实现每一层输入进行相同的白化将是有利的。通过白化每一层的输入，我们将采取措施实现输入的固定分布，消除内部协变量转移的不良影响。

We could consider whitening activations at every training step or at some interval, either by modifying the network directly or by changing the parameters of the optimization algorithm to depend on the network activation values (Wiesler et al., 2014; Raiko et al., 2012; Povey et al., 2014; Desjardins & Kavukcuoglu). However, if these modifications are interspersed with the optimization steps, then the gradient descent step may attempt to update the parameters in a way that requires the normalization to be updated, which reduces the effect of the gradient step. For example, consider a layer with the input u that adds the learned bias b, and normalizes the result by subtracting the mean of the activation computed over the training data: $\hat{x} = x - E[x]$ where $x = u + b, X = x_{1...N}$ is the set of values of x over the training set, and $E[x] = \frac{1}{N}\sum_{i=1}^{N} x_i$. If a gradient descent step ignores the dependence of $E[x]$ on b, then it will update $b \leftarrow b + \Delta b$, where $\Delta b \propto -\partial \ell / \partial \hat{x}$. Then $u + (b + \Delta b) - E[u + (b + \Delta b)] = u + b - E[u + b]$. Thus, the combination of the update to b and subsequent change in normalization led to no change in the output of the layer nor, consequently, the loss. As the training continues, b will grow indefinitely while the loss remains fixed. This problem can get worse if the normalization not only centers but also scales the activations. We have observed this empirically in initial experiments, where the model blows up when the normalization parameters are computed outside the gradient descent step.

我们考虑在每个训练步骤或在某些间隔来白化激活值，通过直接修改网络或根据网络激活值来更改优化方法的参数(Wiesler et al., 2014; Raiko et al., 2012; Povey et al., 2014; Desjardins & Kavukcuoglu)。然而，如果这些修改分散在优化步骤中，那么梯度下降步骤可能会试图以要求标准化进行更新的方式来更新参数，这会降低梯度下降步骤的影响。例如，考虑一个层，其输入u加上学习到的偏置b，通过减去在训练集上计算的激活值的均值对结果进行归一化：$\hat{x} = x - E[x]$ ，$x = u + b, X = x_{1...N}$是训练集上x值的集合，$E[x] = \frac{1}{N}\sum_{i=1}^{N} x_i$。如果梯度下降步骤忽略了$E[x]$对$b$的依赖，那它将更新$b \leftarrow b + \Delta b$ ，其中$\Delta b \propto -\partial \ell / \partial \hat{x}$。然后 $u + (b + \Delta b) - E[u + (b + \Delta b)] = u + b - E[u + b]$ 。因此，结合b的更新和接下来标准化中的改变会导致层的输出没有变化，从而导致损失没有变化。随着训练的继续，b将无限增长而损失保持不变。如果标准化不仅中心化而且缩放了激活值，问题会变得更糟糕。我们在最初的实验中已经观察到了这一点，当标准化参数在梯度下降步骤之外计算时，模型会爆炸。

The issue with the above approach is that the gradient descent optimization does not take into account the fact that the normalization takes place. To address this issue, we would like to ensure that, for any parameter values,

the network *always* produces activations with the desired distribution. Doing so would allow the gradient of the loss with respect to the model parameters to account for the normalization, and for its dependence on the model parameters Θ. Let again x be a layer input, treated as a vector, and $\mathcal{X}$ be the set of these inputs over the training data set. The normalization can then be written as a transformation

$$\hat{x} = Norm(x, \mathcal{X})$$

which depends not only on the given training example x but on all examples $\mathcal{X}$ – each of which depends on Θ if x is generated by another layer. For backpropagation, we would need to compute the Jacobians $\frac{\partial Norm(x,\mathcal{X})}{\partial x}$ and $\frac{\partial Norm(x,\mathcal{X})}{\partial \mathcal{X}}$; ignoring the latter term would lead to the explosion described above. Within this framework, whitening the layer inputs is expensive, as it requires computing the covariance matrix $Cov[x] = E_{x\in\mathcal{X}}[xx^T] - E[x]E[x]^T$ and its inverse square root, to produce the whitened activations $Cov[x]^{-1/2}(x - E[x])$, as well as the derivatives of these transforms for backpropagation. This motivates us to seek an alternative that performs input normalization in a way that is differentiable and does not require the analysis of the entire training set after every parameter update.

上述方法的问题是梯度下降优化没有考虑到标准化中发生的事实。为了解决这个问题，我们希望确保对于任何参数值，网络*总是*产生具有所需分布的激活值。这样做将允许关于模型参数损失的梯度来解释标准化，以及它对模型参数Θ的依赖。设x为层的输入，将其看作向量，$\mathcal{X}$是这些输入在训练集上的集合。标准化可以写为变换

$$\hat{x} = Norm(x, \mathcal{X})$$

它不仅依赖于给定的训练样本x而且依赖于所有样本$\mathcal{X}$——它们中的每一个都依赖于Θ，如果x是由另一层生成的。对于反向传播，我们将需要计算雅可比行列式$\frac{\partial Norm(x,\mathcal{X})}{\partial x}$和$\frac{\partial Norm(x,\mathcal{X})}{\partial \mathcal{X}}$；忽略后一项会导致上面描述的爆炸。在这个框架中，白化层输入是昂贵的，因为它要求计算协方差矩阵 $Cov[x] = E_{x\in\mathcal{X}}[xx^T] - E[x]E[x]^T$和它的平方根倒数，从而生成白化的激活$Cov[x]^{-1/2}(x - E[x])$和这些变换进行反向传播的偏导数。这促使我们寻求一种替代方案，以可微分的方式执行输入标准化，并且在每次参数更新后不需要对整个训练集进行分析。

Some of the previous approaches (e.g. (Lyu & Simoncelli, 2008)) use statistics computed over a single training example, or, in the case of image networks, over different feature maps at a given location. However, this changes the representation ability of a network by discarding the absolute scale of activations. We want to a preserve the information in the network, by normalizing the activations in a training example relative to the statistics of the entire training data.

以前的一些方法（例如（Lyu＆Simoncelli，2008））使用通过单个训练样本计算的统计信息，或者在图像网络的情况下，使用给定位置处不同特征图上的统计。然而，通过丢弃激活值绝对尺度改变了网络的表示能力。我们希望通过对相对于整个训练数据统计信息的单个训练样本的激活值进行归一化来保留网络中的信息。

3. Normalization via Mini-Batch Statistics

Since the full whitening of each layer's inputs is costly and not everywhere differentiable, we make two necessary simplifications. The first is that instead of whitening the features in layer inputs and outputs jointly, we will normalize each scalar feature independently, by making it have the mean of zero and unit variance. For a layer with d-dimensional input $x = (x^{(1)} \dots x^{(d)})$, we will normalize each dimension

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

where the expectation and variance are computed over the training data set. As shown in (LeCun et al., 1998b), such normalization speeds up convergence, even when the features are not decorrelated.

3. 通过Mini-Batch统计进行标准化

由于每一层输入的整个白化是代价昂贵的并且不是到处可微分的，因此我们做了两个必要的简化。首先是我们将单独标准化每个标量特征，从而代替在层输入输出对特征进行共同白化，使其具有零均值和单位方差。对于具有d维输入$x = (x^{(1)} \dots x^{(d)})$的层，我们将标准化每一维

$$\hat{x}^{(k)} = \frac{x^{(k)} - E[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

其中期望和方差在整个训练数据集上计算。如(LeCun et al., 1998b)中所示，这种标准化加速了收敛，即使特征没有去相关。

Note that simply normalizing each input of a layer may change what the layer can represent. For instance, normalizing the inputs of a sigmoid would constrain them to the linear regime of the nonlinearity. To address this, we make sure that *the transformation inserted in the network can represent the identity transform.* To accomplish this, we introduce, for each activation $x^{(k)}$, a pair of parameters $\gamma^{(k)}$, $\beta^{(k)}$, which scale and shift the normalized value:

$$y^{(k)} = \gamma^{(k)} \hat{x}^{(k)} + \beta^{(k)}.$$

These parameters are learned along with the original model parameters, and restore the representation power of the network. Indeed, by setting $\gamma^{(k)} = \sqrt{Var[x^{(k)}]}$ and $\beta^{(k)} = E[x^{(k)}]$, we could recover the original activations, if that were the optimal thing to do.

注意简单标准化层的每一个输入可能会改变层可以表示什么。例如，标准化sigmoid的输入会将它们约束到非线性的线性状态。为了解决这个问题，我们要确保*插入到网络中的变换可以表示恒等变换*。为了实现这个，对于每一个激活值$x^{(k)}$，我们引入成对的参数$\gamma^{(k)}$，$\beta^{(k)}$，它们会归一化和移动标准化值：

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}.$$

这些参数与原始的模型参数一起学习，并恢复网络的表示能力。实际上，通过设置$\gamma^{(k)} = \sqrt{Var[x^{(k)}]}$和$\beta^{(k)} = E[x^{(k)}]$，我们可以重新获得原始的激活值，如果这是要做的最优的事。

In the batch setting where each training step is based on the entire training set, we would use the whole set to normalize activations. However, this is impractical when using stochastic optimization. Therefore, we make the second simplification: since we use mini-batches in stochastic gradient training, *each mini-batch produces estimates of the mean and variance* of each activation. This way, the statistics used for normalization can fully participate in the gradient backpropagation. Note that the use of mini-batches is enabled by computation of per-dimension variances rather than joint covariances; in the joint case, regularization would be required since the mini-batch size is likely to be smaller than the number of activations being whitened, resulting in singular covariance matrices.

每个训练步骤的批处理设置是基于整个训练集的，我们将使用整个训练集来标准化激活值。然而，当使用随机优化时，这是不切实际的。因此，我们做了第二个简化：由于我们在随机梯度训练中使用小批量，*每个小批量产生每次激活平均值和方差*的估计。这样，用于标准化的统计信息可以完全参与梯度反向传播。注意，通过计算每一维的方差而不是联合协方差，可以实现小批量的使用；在联合情况下，将需要正则化，因为小批量大小可能小于白化的激活值的数量，从而导致单个协方差矩阵。

Consider a mini-batch $\mathcal{B}$ of size m. Since the normalization is applied to each activation independently, let us focus on a particular activation $x^{(k)}$ and omit k for clarity. We have m values of this activation in the mini-batch,

$$\mathcal{B} = \{x_{1\ldots m}\}.$$

Let the normalized values be $\hat{x}_{1\ldots m}$, and their linear transformations be $y_{1\ldots m}$. We refer to the transform

$$BN_{\gamma,\beta} : x_{1\ldots m} \rightarrow y_{1\ldots m}$$

as the *Batch Normalizing Transform*. We present the BN Transform in Algorithm 1. In the algorithm, ϵ is a constant added to the mini-batch variance for numerical stability.

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

考虑一个大小为m的小批量数据$\mathcal{B}$。由于标准化被单独地应用于每一个激活，所以让我们集中在一个特定的激活$x^{(k)}$，为了清晰忽略k。在小批量数据里我们有这个激活的m个值，

$$\mathcal{B} = \{x_{1...m}\}.$$

设标准化值为$\hat{x}_{1...m}$，它们的线性变换为$y_{1...m}$。我们把变换

$$BN_{\gamma,\beta} : x_{1...m} \rightarrow y_{1...m}$$

看作*批标准化变换*。我们在算法1中提出了BN变换。在算法中，为了数值稳定，ϵ是一个加到小批量数据方差上的常量。

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

The BN transform can be added to a network to manipulate any activation. In the notation $y = BN_{\gamma,\beta}(x)$, we indicate that the parameters γ and β are to be learned, but it should be noted that the BN transform does not independently process the activation in each training example. Rather, $BN_{\gamma,\beta}(x)$ depends both on the training example *and the other examples in the mini-batch*. The scaled and shifted values y are passed to other network layers. The normalized activations $\hat{x}$ are internal to our transformation, but their presence is crucial. The distributions of values of any $\hat{x}$ has the expected value of 0 and the variance of 1, as long as the elements of each mini-batch are sampled from the same distribution, and if we neglect ϵ. This can be seen by observing that $\sum_{i=1}^{m}\hat{x}_i = 0$ and $\frac{1}{m}\sum_{i=1}^{m}\hat{x}_i^2 = 1$, and taking expectations. Each normalized activation $\hat{x}^{(k)}$ can be viewed as an input to a sub-network composed of the linear transform $y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}$, followed by the other processing done by the original network. These sub-network inputs all have fixed means and variances, and although the joint distribution of these normalized $\hat{x}^{(k)}$ can change over the course of training, we expect that the introduction of normalized inputs accelerates the training of the sub-network and, consequently, the network as a whole.

BN变换可以添加到网络上来操纵任何激活。在公式$y = BN_{\gamma,\beta}(x)$中，我们指出参数γ和β需要进行学习，但应该注意到在每一个训练样本中BN变换不单独处理激活。相反，$BN_{\gamma,\beta}(x)$取决于训练样本和*小批量数据中的其它样本*。缩放和移动的值y传递到其它的网络层。标准化的激活值$\hat{x}$在我们的变换内部，但它们的存在至关重

要。只要每个小批量的元素从相同的分布中进行采样，如果我们忽略ϵ，那么任何$\hat{x}$值的分布都具有期望为0，方差为1。这可以通过观察$\sum_{i=1}^{m}\hat{x}_i=0$和$\frac{1}{m}\sum_{i=1}^{m}\hat{x}_i^2=1$看到，并取得预期。每一个标准化的激活值$\hat{x}^{(k)}$可以看作由线性变换$y^{(k)}=\gamma^{(k)}\hat{x}^{(k)}+\beta^{(k)}$组成的子网络的输入，接下来是原始网络的其它处理。所有的这些子网络输入都有固定的均值和方差，尽管这些标准化的$\hat{x}^{(k)}$的联合分布可能在训练过程中改变，但我们预计标准化输入的引入会加速子网络的训练，从而加速整个网络的训练。

During training we need to backpropagate the gradient of loss ℓ through this transformation, as well as compute the gradients with respect to the parameters of the BN transform. We use chain rule, as follows (before simplification):

$$
\begin{aligned}
\frac{\partial \ell}{\partial \hat{x}_i} &= \frac{\partial \ell}{\partial y_i}\cdot\gamma \\
\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} &= \sum_{i=1}^{m}\frac{\partial \ell}{\partial \hat{x}_i}\cdot(x_i-\mu_{\mathcal{B}})\cdot\frac{-1}{2}(\sigma_{\mathcal{B}}^2+\epsilon)^{-3/2} \\
\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} &= \sum_{i=1}^{m}\frac{\partial \ell}{\partial \hat{x}_i}\cdot\frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}} \\
\frac{\partial \ell}{\partial x_i} &= \sum_{i=1}^{m}\frac{\partial \ell}{\partial \hat{x}_i}\cdot\frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2+\epsilon}}+\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2}\cdot\frac{2(x_i-\mu_{\mathcal{B}})}{m}+\frac{\partial \ell}{\partial \mu_{\mathcal{B}}}\cdot\frac{1}{m} \\
\frac{\partial \ell}{\partial \gamma} &= \sum_{i=1}^{m}\frac{\partial \ell}{\partial y_i}\cdot\hat{x}_i \\
\frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^{m}\frac{\partial \ell}{\partial y_i}
\end{aligned}
$$

Thus, BN transform is a differentiable transformation that introduces normalized activations into the network. This ensures that as the model is training, layers can continue learning on input distributions that exhibit less internal covariate shift, thus accelerating the training. Furthermore, the learned affine transform applied to these normalized activations allows the BN transform to represent the identity transformation and preserves the network capacity.

在训练过程中我们需要通过这个变换反向传播损失ℓ的梯度，以及计算关于BN变换参数的梯度。我们使用的链式法则如下（简化之前）：

$$
\begin{aligned}
\frac{\partial \ell}{\partial \hat{x}_i} &= \frac{\partial \ell}{\partial y_i} \cdot \gamma \\
\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-1}{2} (\sigma_{\mathcal{B}}^2 + \epsilon)^{-3/2} \\
\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \\
\frac{\partial \ell}{\partial x_i} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial \hat{x}_i} \cdot \frac{-1}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}^2} \cdot \frac{2(x_i - \mu_{\mathcal{B}})}{m} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m} \\
\frac{\partial \ell}{\partial \gamma} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \hat{x}_i \\
\frac{\partial \ell}{\partial \beta} &= \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}
\end{aligned}
$$

因此，BN变换是将标准化激活引入到网络中的可微变换。这确保了在模型训练时，层可以继续学习输入分布，表现出更少的内部协变量转移，从而加快训练。此外，应用于这些标准化的激活上的学习到的仿射变换允许BN变换表示恒等变换并保留网络的能力。

3.1. Training and Inference with Batch-Normalized Networks

To *Batch-Normalize* a network, we specify a subset of activations and insert the BN transform for each of them, according to Alg.1. Any layer that previously received x as the input, now receives $BN(x)$. A model employing Batch Normalization can be trained using batch gradient descent, or Stochastic Gradient Descent with a mini-batch size $m > 1$, or with any of its variants such as Adagrad (Duchi et al., 2011). The normalization of activations that depends on the mini-batch allows efficient training, but is neither necessary nor desirable during inference; we want the output to depend only on the input, deterministically. For this, once the network has been trained, we use the normalization

$$
\hat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}
$$

using the population, rather than mini-batch, statistics. Neglecting ϵ, these normalized activations have the same mean 0 and variance 1 as during training. We use the unbiased variance estimate $Var[x] = \frac{m}{m-1} \cdot E_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$, where the expectation is over training mini-batches of size m and $\sigma_{\mathcal{B}}^2$ are their sample variances. Using moving averages instead, we can track the accuracy of a model as it trains. Since the means and variances are fixed during inference, the normalization is simply a linear transform applied to each activation. It may further be composed with the scaling by γ and shift by β, to yield a single linear transform that replaces $BN(x)$. Algorithm 2 summarizes the procedure for training batch-normalized networks.

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network

3.1 批标准化网络的训练和推断

为了*批标准化*一个网络，根据算法1，我们指定一个激活的子集，然后在每一个激活中插入BN变换。任何以前接收x作为输入的层现在接收$BN(x)$作为输入。采用批标准化的模型可以使用批梯度下降，或者用小批量数据大小为$m > 1$的随机梯度下降，或使用它的任何变种例如Adagrad (Duchi et al., 2011)进行训练。依赖小批量数

据的激活值的标准化可以有效地训练，但在推断过程中是不必要的也是不需要的；我们希望输出只确定性地取决于输入。为此，一旦网络训练完成，我们使用总体统计来进行标准化

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}}$$

，而不是小批量数据统计。跟训练过程中一样，如果忽略ϵ，这些标准化的激活具有相同的均值0和方差1。我们使用无偏方差估计$Var[x] = \frac{m}{m-1} \cdot E_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$，其中期望是在大小为$m$的小批量训练数据上得到的，$\sigma_{\mathcal{B}}^2$是其样本方差。使用这些值移动平均，我们在训练过程中可以跟踪模型的准确性。由于均值和方差在推断时是固定的，因此标准化是应用到每一个激活上的简单线性变换。它可以进一步由缩放γ和转移β组成，以产生代替$BN(x)$的单线性变换。算法2总结了训练批标准化网络的过程。

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$

Output: Batch-normalized network for inference, $N_{\text{BN}}^{\text{inf}}$

1: $N_{\text{BN}}^{\text{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \text{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\text{BN}}^{\text{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\text{BN}}^{\text{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\text{BN}}^{\text{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\text{BN}}^{\text{inf}} \leftarrow N_{\text{BN}}^{\text{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\text{E}[x] \leftarrow \text{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\text{Var}[x] \leftarrow \frac{m}{m-1}\text{E}_{\mathcal{B}}[\sigma_{\mathcal{B}}^2]$$

11: In $N_{\text{BN}}^{\text{inf}}$, replace the transform $y = \text{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\text{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\text{E}[x]}{\sqrt{\text{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network

3.2. Batch-Normalized Convolutional Networks

Batch Normalization can be applied to any set of activations in the network. Here, we focus on transforms that consist of an affine transformation followed by an element-wise nonlinearity:

$$z = g(Wu + b)$$

where W and b are learned parameters of the model, and $g(\cdot)$ is the nonlinearity such as sigmoid or ReLU. This formulation covers both fully-connected and convolutional layers. We add the BN transform immediately before the nonlinearity, by normalizing $x = Wu + b$. We could have also normalized the layer inputs u, but since u is likely the output of another nonlinearity, the shape of its distribution is likely to change during training, and constraining its first and second moments would not eliminate the covariate shift. In contrast, $Wu + b$ is more likely to have a symmetric, non-sparse distribution, that is "more Gaussian" (Hyvärinen & Oja, 2000); normalizing it is likely to produce activations with a stable distribution.

3.2. 批标准化卷积网络

批标准化可以应用于网络的任何激活集合。这里我们专注于仿射变换和元素级非线性组成的变换：

$$z = g(Wu + b)$$

其中W和b是模型学习的参数，$g(\cdot)$是非线性例如sigmoid或ReLU。这个公式涵盖了全连接层和卷积层。我们在非线性之前通过标准化$x = Wu + b$加入BN变换。我们也可以标准化层输入u，但由于u可能是另一个非线性的输出，它的分布形状可能在训练过程中改变，并且限制其第一矩或第二矩不能去除协变量转移。相比之下，$Wu + b$更可能具有对称，非稀疏分布，即"更高斯"（Hyvärinen & Oja，2000）；对其标准化可能产生具有稳定分布的激活。

Note that, since we normalize $Wu + b$, the bias b can be ignored since its effect will be canceled by the subsequent mean subtraction (the role of the bias is subsumed by β in Alg.1). Thus, $z = g(Wu + b)$ is replaced with

$$z = g(BN(Wu))$$

where the BN transform is applied independently to each dimension of $x = Wu$, with a separate pair of learned parameters $\gamma^{(k)}$, $\beta^{(k)}$ per dimension.

注意，由于我们对$Wu + b$进行标准化，偏置b可以忽略，因为它的效应将会被后面的中心化取消（偏置的作用会归入到算法1的β）。因此，$z = g(Wu + b)$被

$$z = g(BN(Wu))$$

替代，其中BN变换独立地应用到$x = Wu$的每一维，每一维具有单独的成对学习参数$\gamma^{(k)}$，$\beta^{(k)}$。

For convolutional layers, we additionally want the normalization to obey the convolutional property —— so that different elements of the same feature map, at different locations, are normalized in the same way. To achieve this, we jointly normalize all the activations in a mini-batch, over all locations. In Alg.1, we let $\mathcal{B}$ be the set of all values in a feature map across both the elements of a mini-batch and spatial locations —— so for a mini-batch of size m and feature maps of size $p \times q$, we use the effective mini-batch of size $m' = |\mathcal{B}| = m \cdot p\,q$. We learn a

pair of parameters $\gamma^{(k)}$ and $\beta^{(k)}$ per feature map, rather than per activation. Alg.2 is modified similarly, so that during inference the BN transform applies the same linear transformation to each activation in a given feature map.

另外，对于卷积层我们希望标准化遵循卷积特性——为的是同一特征映射的不同元素，在不同的位置，以相同的方式进行标准化。为了实现这个，我们在所有位置联合标准化了小批量数据中的所有激活。在算法1中，我们让$\mathcal{B}$是跨越小批量数据的所有元素和空间位置的特征图中所有值的集合——因此对于大小为m的小批量数据和大小为$p \times q$的特征映射，我们使用有效的大小为$m' = |\mathcal{B}| = m \cdot p\,q$的小批量数据。我们每个特征映射学习一对参数$\gamma^{(k)}$和$\beta^{(k)}$，而不是每个激活。算法2进行类似的修改，以便推断期间BN变换对在给定的特征映射上的每一个激活应用同样的线性变换。

3.3. Batch Normalization enables higher learning rates

In traditional deep networks, too high a learning rate may result in the gradients that explode or vanish, as well as getting stuck in poor local minima. Batch Normalization helps address these issues. By normalizing activations throughout the network, it prevents small changes in layer parameters from amplifying as the data propagates through a deep network. For example, this enables the sigmoid nonlinearities to more easily stay in their non-saturated regimes, which is crucial for training deep sigmoid networks but has traditionally been hard to accomplish.

3.3. 批标准化可以提高学习率

在传统的深度网络中，学习率过高可能会导致梯度爆炸或梯度消失，以及陷入差的局部最小值。批标准化有助于解决这些问题。通过标准化整个网络的激活值，在数据通过深度网络传播时，它可以防止层参数的微小变化被放大。例如，这使sigmoid非线性更容易保持在它们的非饱和状态，这对训练深度sigmoid网络至关重要，但在传统上很难实现。

Batch Normalization also makes training more resilient to the parameter scale. Normally, large learning rates may increase the scale of layer parameters, which then amplify the gradient during backpropagation and lead to the model explosion. However, with Batch Normalization, backpropagation through a layer is unaffected by the scale of its parameters. Indeed, for a scalar a,

$$BN(Wu) = BN((aW)u)$$

and thus $\frac{\partial BN((aW)u)}{\partial u} = \frac{\partial BN(Wu)}{\partial u}$, so the scale does not affect the layer Jacobian nor, consequently, the gradient propagation. Moreover, $\frac{\partial BN((aW)u)}{\partial (aW)} = \frac{\partial BN(Wu)}{\partial W}$ so larger weights lead to *smaller* gradients, and Batch Normalization will stabilize the parameter growth.

批标准化也使训练对参数的缩放更有弹性。通常，大的学习率可能会增加层参数的缩放，这会在反向传播中放大梯度并导致模型爆炸。然而，通过批标准化，通过层的反向传播不受其参数缩放的影响。实际上，对于标量 a，

$$BN(Wu) = BN((aW)u)$$

因此 $\frac{\partial BN((aW)u)}{\partial u} = \frac{\partial BN(Wu)}{\partial u}$，因此标量不影响层的雅可比行列式，从而不影响梯度传播。此外，$\frac{\partial BN((aW)u)}{\partial (aW)} = \frac{1}{a} \cdot \frac{\partial BN(Wu)}{\partial W}$，因此更大的权重会导致*更小的*梯度，并且批标准化会稳定参数的增长。

We further conjecture that Batch Normalization may lead the layer Jacobians to have singular values close to 1, which is known to be beneficial for training (Saxe et al., 2013). Consider two consecutive layers with normalized inputs, and the transformation between these normalized vectors: $\hat{z} = F(\hat{x})$. If we assume that $\hat{x}$ and $\hat{z}$ are Gaussian and uncorrelated, and that $F(\hat{x}) \approx J\hat{x}$ is a linear transformation for the given model parameters, then both $\hat{x}$ and $\hat{z}$ have unit covariances, and $I = Cov[\hat{z}] = JCov[\hat{x}]J^T = JJ^T$. Thus, J is orthogonal, which preserves the gradient magnitudes during backpropagation. Although the above assumptions are not true in reality, we expect Batch Normalization to help make gradient propagation better behaved. This remains an area of further study.

我们进一步推测，批标准化可能会导致雅可比行列式的奇异值接近于1，这被认为对训练是有利的(Saxe et al., 2013)。考虑具有标准化输入的两个连续的层，并且变换位于这些标准化向量之间：$\hat{z} = F(\hat{x})$。如果我们假设$\hat{x}$和$\hat{z}$是高斯分布且不相关的，那么$F(\hat{x}) \approx J\hat{x}$是对给定模型参数的一个线性变换，$\hat{x}$和$\hat{z}$有单位方差，并且 $I = Cov[\hat{z}] = JCov[\hat{x}]J^T = JJ^T$。因此，$J$是正交的，其保留了反向传播中的梯度大小。尽管上述假设在现实中不是真实的，但我们希望批标准化有助于梯度传播更好的执行。这有待于进一步研究。

4. Experiments

4.1. Activations over time

To verify the effects of internal covariate shift on training, and the ability of Batch Normalization to combat it, we considered the problem of predicting the digit class on the MNIST dataset (LeCun et al., 1998a). We used a very simple network, with a 28x28 binary image as input, and 3 fully-connected hidden layers with 100 activations each. Each hidden layer computes $y = g(Wu + b)$ with sigmoid nonlinearity, and the weights W initialized to small random Gaussian values. The last hidden layer is followed by a fully-connected layer with 10 activations (one per class) and cross-entropy loss. We trained the network for 50000 steps, with 60 examples per mini-batch. We added Batch Normalization to each hidden layer of the network, as in Sec.3.1. We were interested in the comparison between the baseline and batch-normalized networks, rather than achieving the state of the art performance on MNIST (which the described architecture does not).

4. 实验

4.1. 随时间激活

为了验证内部协变量转移对训练的影响，以及批标准化对抗它的能力，我们考虑了在MNIST数据集上预测数字类别的问题(LeCun et al., 1998a)。我们使用非常简单的网络，28x28的二值图像作为输入，以及三个全连接层，每层100个激活。每一个隐藏层用sigmoid非线性计算$y = g(Wu + b)$，权重W初始化为小的随机高斯值。最后的隐藏层之后是具有10个激活（每类1个）和交叉熵损失的全连接层。我们训练网络50000次迭代，每份小批量数据中有60个样本。如第3.1节所述，我们在网络的每一个隐藏层后添加批标准化。我们对基准线和批标准化网络之间的比较感兴趣，而不是实现在MNIST上的最佳性能（所描述的架构没有）。

Figure 1(a) shows the fraction of correct predictions by the two networks on held-out test data, as training progresses. The batch-normalized network enjoys the higher test accuracy. To investigate why, we studied inputs to the sigmoid, in the original network N and batch-normalized network N^{tr}_{BN} (Alg. 2) over the course of training. In Fig. 1(b,c) we show, for one typical activation from the last hidden layer of each network, how its distribution evolves. The distributions in the original network change significantly over time, both in their mean and the variance, which complicates the training of the subsequent layers. In contrast, the distributions in the batch-normalized network are much more stable as training progresses, which aids the training.

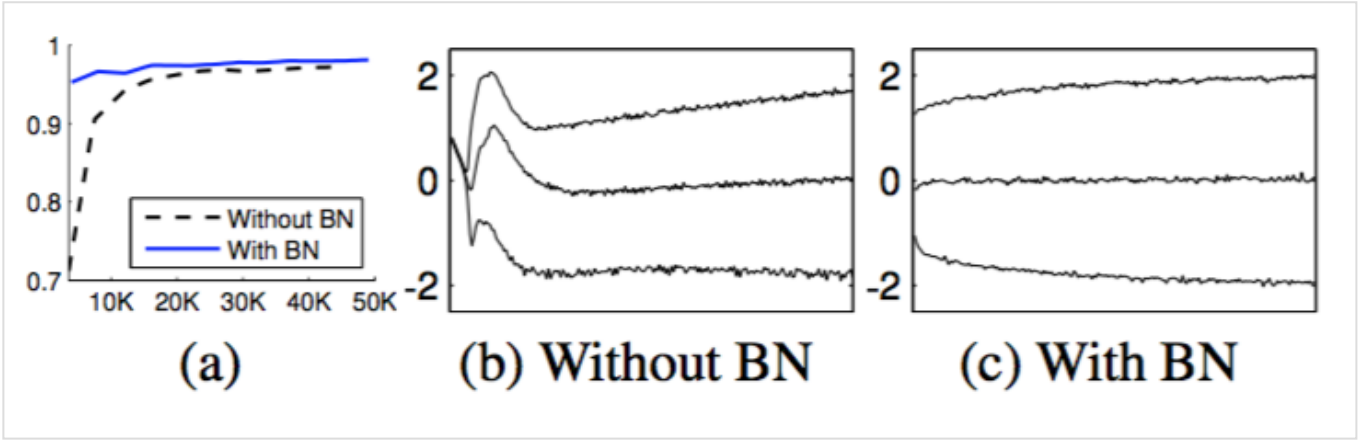

Figure 1. (a) The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy. (b, c) The evolution of input distributions to a typical sigmoid, over the course of training, shown as {15, 50, 85}th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.

图1(a)显示了随着训练进行，两个网络在提供的测试数据上正确预测的分数。批标准化网络具有更高的测试准确率。为了调查原因，我们在训练过程中研究了原始网络N和批标准化网络N^{tr}_{BN}(Alg. 2)中的sigmoid输入。在图1(b，c)中，我们显示，对于来自每个网络的最后一个隐藏层的一个典型的激活，其分布如何演变。原始网络中的分布随着时间的推移而发生显著变化，无论是平均值还是方差，都会使后面的层的训练复杂化。相比之下，随着训练的进行，批标准化网络中的分布更加稳定，这有助于训练。

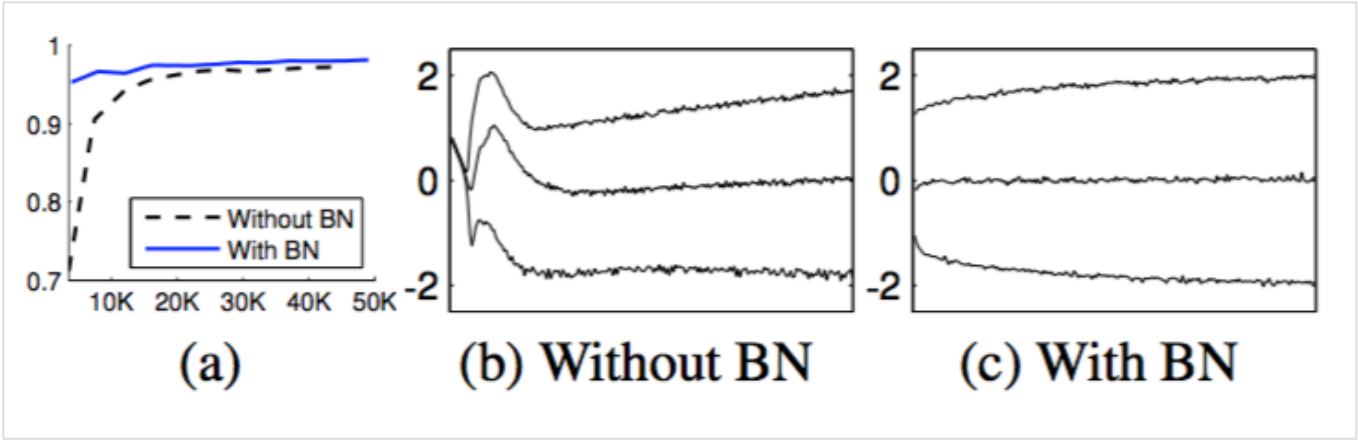

图1。(a)使用批标准化和不使用批标准化训练的网络在MNIST上的测试准确率，以及训练的迭代次数。批标准化有助于网络训练的更快，取得更高的准确率。(b，c)典型的sigmoid在训练过程中输入分布的演变，显示为15%，50%，85%。批标准化使分布更稳定并降低了内部协变量转移。

4.2. ImageNet classification

We applied Batch Normalization to a new variant of the Inception network (Szegedy et al., 2014), trained on the ImageNet classification task (Russakovsky et al., 2014). The network has a large number of convolutional and pooling layers, with a softmax layer to predict the image class, out of 1000 possibilities. Convolutional layers use ReLU as the nonlinearity. The main difference to the network described in (Szegedy et al., 2014) is that the 5x5 convolutional layers are replaced by two consecutive layers of 3x3 convolutions with up to 128 filters. The network contains $13.6 \cdot 10^6$ parameters, and, other than the top softmax layer, has no fully-connected layers. We refer to this model as Inception in the rest of the text. The training was performed on a large-scale, distributed architecture (Dean et al., 2012), using 5 concurrent steps on each of 10 model replicas, using asynchronous SGD with momentum (Sutskever et al.,2013), with the mini-batch size of 32. All networks are evaluated as training progresses by computing the validation accuracy @1, i.e. the probability of predicting the correct label out of 1000 possibilities, on a held-out set, using a single crop per image.

4.2. ImageNet分类

我们将批标准化化应用于在ImageNet分类任务（Russakovsky等，2014）上训练的Inception网络的新变种（Szegedy等，2014）。网络具有大量的卷积和池化层，和一个softmax层用来在1000个可能之中预测图像的类别。卷积层使用ReLU作为非线性。与（Szegedy等人，2014年）中描述的网络的主要区别是5×5卷积层被两个连续的3x3卷积层替换，最多可以有128个滤波器。该网络包含$13.6 \cdot 10^6$个参数，除了顶部的softmax层之外，没有全连接层。在其余的文本中我们将这个模型称为Inception。训练在大型分布式架构（Dean et al。，2012）上进行，10个模型副本中的每一个都使用了5个并行步骤，使用异步带动量的SGD（Sutskever等，2013），小批量数据大小为32。随着训练进行，所有网络都通过计算验证准确率@1来评估，即每幅图像使用单个裁剪图像，在1000个可能性中预测正确标签的概率。

In our experiments, we evaluated several modifications of Inception with Batch Normalization. In all cases, Batch Normalization was applied to the input of each nonlinearity, in a convolutional way, as described in section 3.2, while keeping the rest of the architecture constant.

在我们的实验中，我们评估了几个带有批标准化的Inception修改版本。在所有情况下，如第3.2节所述，批标准化以卷积方式应用于每个非线性的输入，同时保持架构的其余部分不变。

4.2.1. ACCELERATING BN NETWORKS

Simply adding Batch Normalization to a network does not take full advantage of our method. To do so, we applied the following modifications:

4.2.1. 加速BN网络

将批标准化简单添加到网络中不能充分利用我们方法的优势。为此，我们进行了以下修改：

Increase learning rate. In a batch-normalized model, we have been able to achieve a training speedup from higher learning rates, with no ill side effects (Sec. 3.3).

提高学习率。在批标准化模型中，我们已经能够从高学习率中实现训练加速，没有不良的副作用（第3.3节）。

Remove Dropout. We have found that removing Dropout from BN-Inception allows the network to achieve higher validation accuracy. We conjecture that Batch Normalization provides similar regularization benefits as Dropout, since the activations observed for a training example are affected by the random selection of examples in the same mini-batch.

删除丢弃。我们发现从BN-Inception中删除丢弃可以使网络实现更高的验证准确率。我们推测，批标准化提供了类似丢弃的正则化收益，因为对于训练样本观察到的激活受到了同一小批量数据中样本随机选择的影响。

Shuffle training examples more thoroughly. We enabled within-shard shuffling of the training data, which prevents the same examples from always appearing in a mini-batch together. This led to about 1% improvement in the validation accuracy, which is consistent with the view of Batch Normalization as a regularizer: the randomization inherent in our method should be most beneficial when it affects an example differently each time it is seen.

更彻底地搅乱训练样本。我们启用了分布内部搅乱训练数据，这样可以防止同一个例子一起出现在小批量数据中。这导致验证准确率提高了约1%，这与批标准化作为正则化项的观点是一致的：它每次被看到时都会影响一个样本，在我们的方法中内在的随机化应该是最有益的。

Reduce the L2 weight regularization. While in Inception an L2 loss on the model parameters controls overfitting, in modified BN-Inception the weight of this loss is reduced by a factor of 5. We find that this improves the accuracy on the held-out validation data.

减少L2全中正则化。虽然在Inception中模型参数的L2损失会控制过拟合，但在修改的BN-Inception中，损失的权重减少了5倍。我们发现这提高了在提供的验证数据上的准确性。

Accelerate the learning rate decay. In training Inception, learning rate was decayed exponentially. Because our network trains faster than Inception, we lower the learning rate 6 times faster.

加速学习率衰减。在训练Inception时，学习率呈指数衰减。因为我们的网络训练速度比Inception更快，所以我们将学习速度降低加快6倍。

Remove Local Response Normalization While Inception and other networks (Srivastava et al., 2014) benefit from it, we found that with Batch Normalization it is not necessary.

删除局部响应归一化。虽然Inception和其它网络（Srivastava等人，2014）从中受益，但是我们发现使用批标准化它是不必要的。

Reduce the photometric distortions. Because batch-normalized networks train faster and observe each training example fewer times, we let the trainer focus on more "real" images by distorting them less.

减少光照扭曲。因为批标准化网络训练更快，并且观察每个训练样本更少的次数，所以通过更少地扭曲它们，我们让训练器关注更多的"真实"图像。

4.2.2. SINGLE-NETWORK CLASSIFICATION

We evaluated the following networks, all trained on the LSVRC2012 training data, and tested on the validation data:

4.2.2. **单网络分类**

我们评估了下面的网络，所有的网络都在LSVRC2012训练数据上训练，并在验证数据上测试：

Inception: the network described at the beginning of Section 4.2, trained with the initial learning rate of 0.0015.

Inception：在4.2小节开头描述的网络，以0.0015的初始学习率进行训练。

BN-Baseline: Same as Inception with Batch Normalization before each nonlinearity.

BN-Baseline：每个非线性之前加上批标准化，其它的与Inception一样。

BN-x5: Inception with Batch Normalization and the modifications in Sec. 4.2.1. The initial learning rate was increased by a factor of 5, to 0.0075. The same learning rate increase with original Inception caused the model parameters to reach machine infinity.

BN-x5：带有批标准化的Inception，修改在4.2.1小节中。初始学习率增加5倍到了0.0075。原始Inception增加同样的学习率会使模型参数达到机器无限大。

BN-x30: Like *BN-x5*, but with the initial learning rate 0.045 (30 times that of Inception).

BN-x30：类似于*BN-x5*，但初始学习率为0.045（Inception学习率的30倍）。

BN-x5-Sigmoid: Like *BN-x5*, but with sigmoid nonlinearity $g(t) = \frac{1}{1+\exp(-x)}$ instead of ReLU. We also attempted to train the original Inception with sigmoid, but the model remained at the accuracy equivalent to chance.

BN-x5-Sigmoid：类似于*BN-x5*，但使用sigmoud非线性$g(t) = \frac{1}{1+\exp(-x)}$来代替ReLU。我们也尝试训练带有sigmoid的原始Inception，但模型保持在相当于机会的准确率。

In Figure 2, we show the validation accuracy of the networks, as a function of the number of training steps. Inception reached the accuracy of 72.2% after $31 \cdot 10^6$ training steps. The Figure 3 shows, for each network, the number of training steps required to reach the same 72.2% accuracy, as well as the maximum validation accuracy reached by the network and the number of steps to reach it.

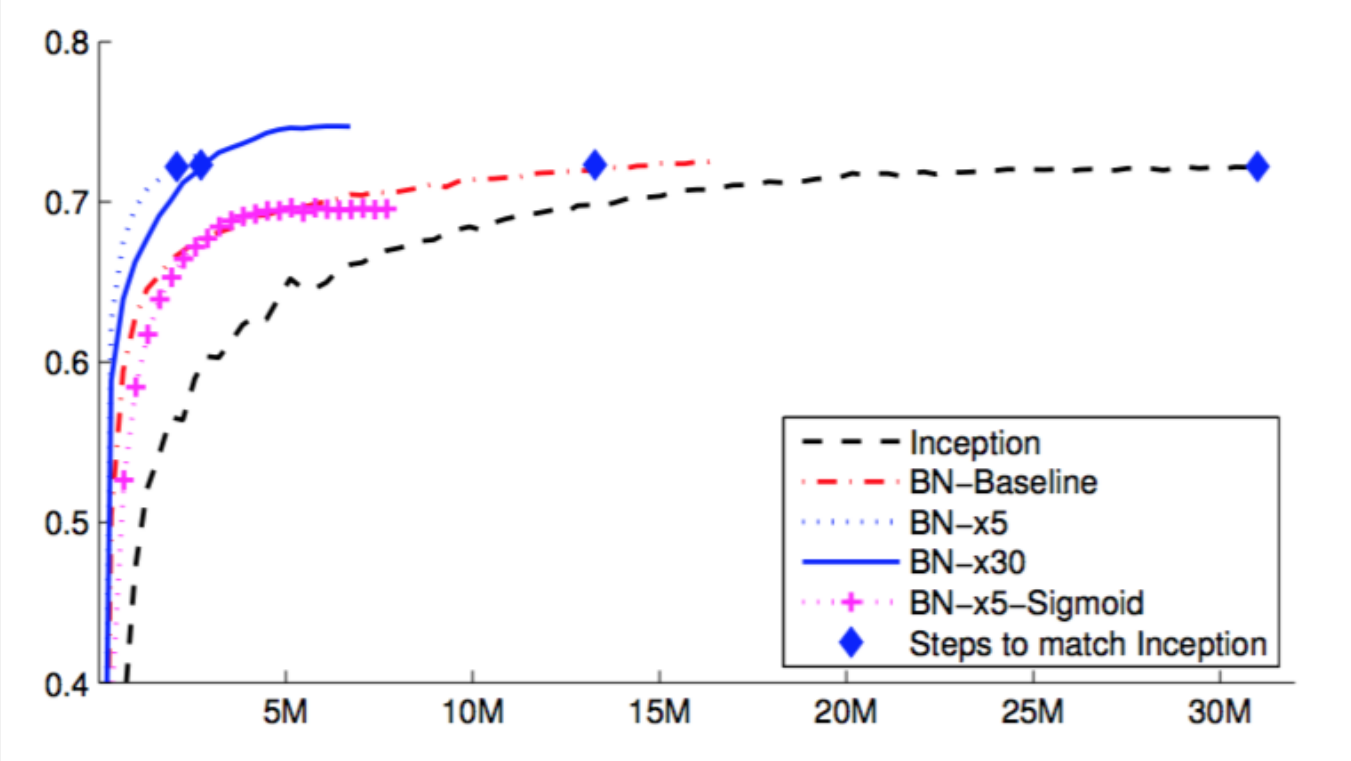

Figure 2. Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.

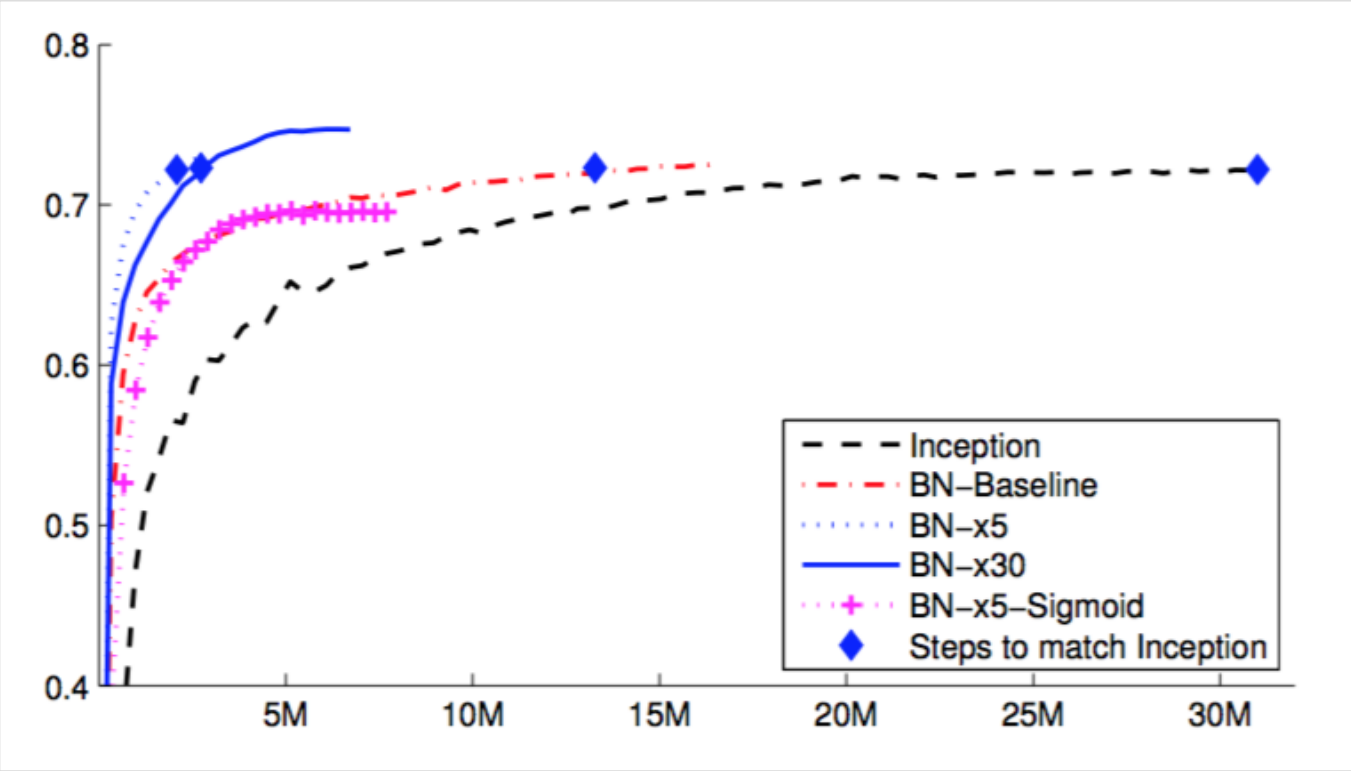

Figure 3. For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.

在图2中，我们显示了网络的验证集准确率，作为训练步骤次数的函数。Inception网络在$31 \cdot 10^6$次训练步骤后达到了72.2%的准确率。图3显示，对于每个网络，达到同样的72.2%准确率需要的训练步骤数量，以及网络达到的最大验证集准确率和达到该准确率的训练步骤数量。

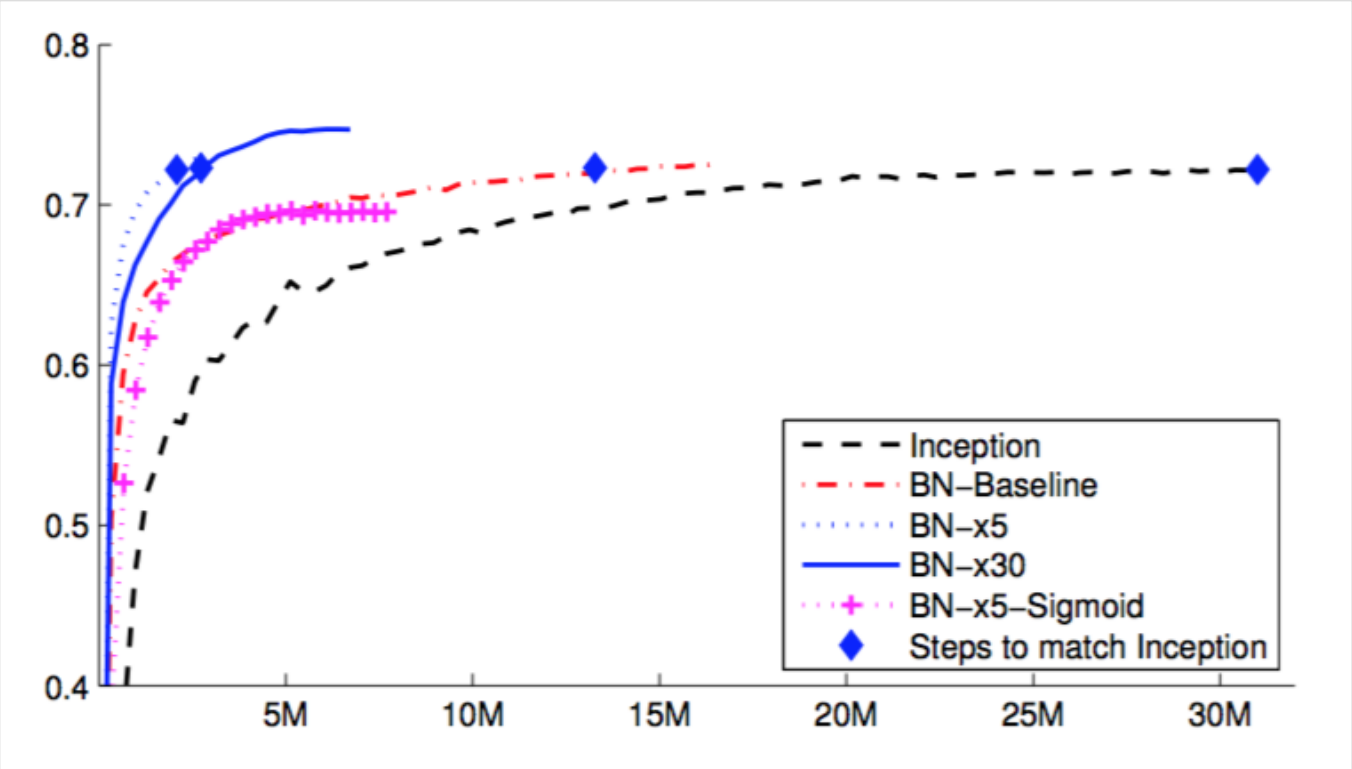

图2。 Inception和它的批标准化变种在单个裁剪图像上的验证准确率以及训练步骤的数量。

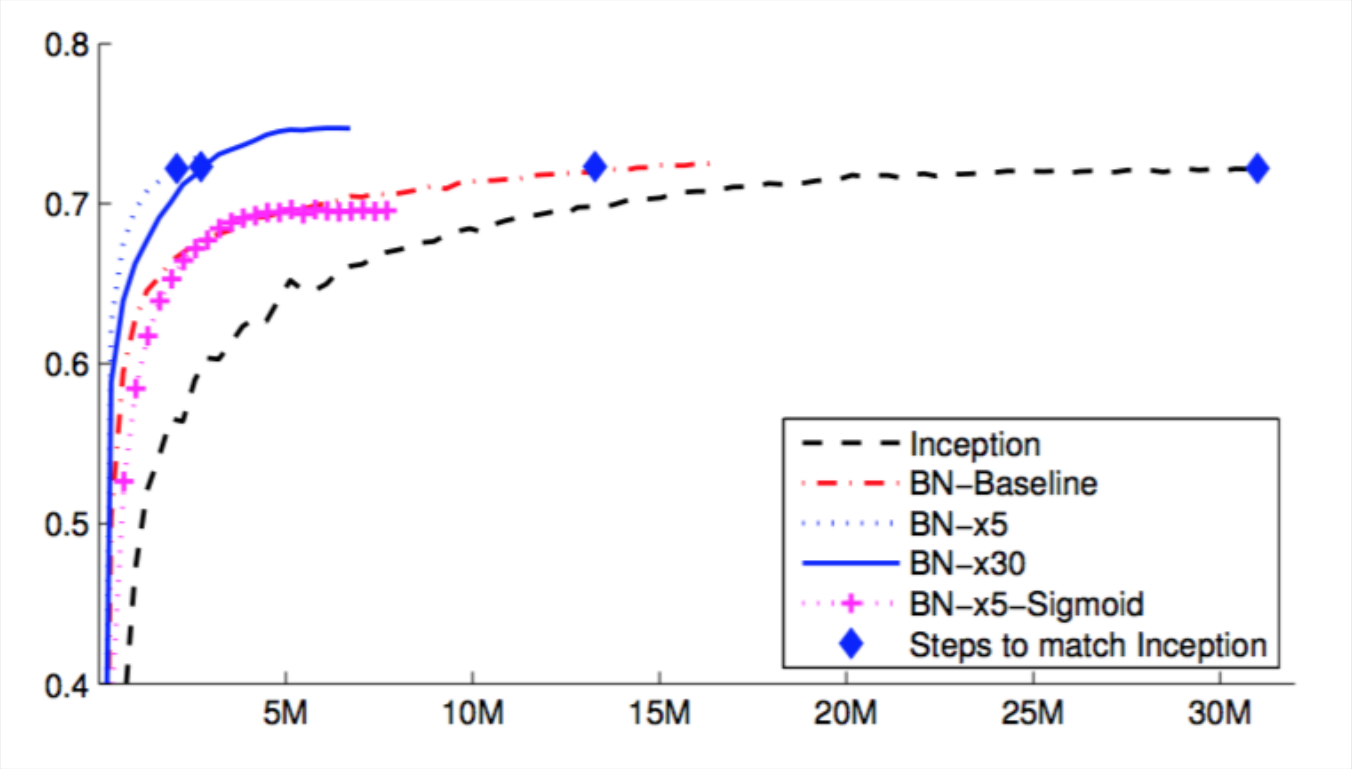

图3。对于Inception和它的批标准化变种，达到Inception最大准确率(72.2%)所需要的训练步骤数量，以及网络取得的最大准确率。

By only using Batch Normalization (BN-Baseline), we match the accuracy of Inception in less than half the number of training steps. By applying the modifications in Sec. 4.2.1, we significantly increase the training speed of the network. *BN-x5* needs 14 times fewer steps than Inception to reach the 72.2% accuracy. Interestingly,

increasing the learning rate further (BN-x30) causes the model to train somewhat slower initially, but allows it to reach a higher final accuracy. This phenomenon is counterintuitive and should be investigated further. BN-x30 reaches 74.8% after $6 \cdot 10^6$ steps, i.e. 5 times fewer steps than required by Inception to reach 72.2%.

通过仅使用批标准化（BN-Baseline），我们在不到Inception一半的训练步骤数量内将准确度与其相匹配。通过应用4.2.1小节中的修改，我们显著提高了网络的训练速度。*BN-x5*需要比Inception少14倍的步骤就达到了72.2%的准确率。有趣的是，进一步提高学习率（BN-x30）使得该模型最初训练有点慢，但可以使其达到更高的最终准确率。这种现象是违反直觉的，应进一步调查。在$6 \cdot 10^6$步骤之后，BN-x30达到74.8%的准确率，即比Inception达到72.2%的准确率所需的步骤减少了5倍。

We also verified that the reduction in internal covariate shift allows deep networks with Batch Normalization to be trained when sigmoid is used as the nonlinearity, despite the well-known difficulty of training such networks. Indeed, BN-x5-Sigmoid achieves the accuracy of 69.8%. Without Batch Normalization, Inception with sigmoid never achieves better than 1/1000 accuracy.

我们也证实了尽管训练这样的网络是众所周知的困难，但是当使用sigmoid作为非线性时，内部协变量转移的减少允许具有批标准化的深层网络被训练。的确，BN-x5-Sigmoid取得了69.8%的准确率达。没有批标准化，使用sigmoid的Inception从未达到比1/1000准确率更好的结果。

4.2.3. ENSEMBLE CLASSIFICATION

The current reported best results on the ImageNet Large Scale Visual Recognition Competition are reached by the Deep Image ensemble of traditional models (Wu et al., 2015) and the ensemble model of (He et al., 2015). The latter reports the error of 4.94%, as evaluated by the ILSVRC test server. Here we report a test error of 4.82% on test server. This improves upon the previous best result, and exceeds the estimated accuracy of human raters according to (Russakovsky et al., 2014).

4.2.3. 组合分类

目前在ImageNet大型视觉识别竞赛中报道的最佳结果是传统模型（Wu et al。，2015）的Deep Image组合和（He等，2015）的组合模型。后者报告了ILSVRC测试服务器评估的`4.94%`的top-5错误率。这里我们在测试服务器上报告`4.82%`的测试错误率。这提高了以前的最佳结果，并且根据（Russakovsky等，2014）这超过了人类评估者的评估准确率。

For our ensemble, we used 6 networks. Each was based on BN-x30, modified via some of the following: increased initial weights in the convolutional layers; using Dropout (with the Dropout probability of 5% or 10%, vs. 40% for the original Inception); and using non-convolutional Batch Normalization with last hidden layers of the model. Each network achieved its maximum accuracy after about $6 \cdot 10^6$ training steps. The ensemble

prediction was based on the arithmetic average of class probabilities predicted by the constituent networks. The details of ensemble and multi-crop inference are similar to (Szegedy et al., 2014).

对于我们的组合，我们使用了6个网络。每个都是基于BN-x30的，进行了以下一些修改：增加卷积层中的初始重量；使用Dropout（丢弃概率为5%或10%，而原始Inception为40%）；模型最后的隐藏层使用非卷积批标准化。每个网络在大约$6 \cdot 10^6$个训练步骤之后实现了最大的准确率。组合预测是基于组成网络的预测类概率的算术平均。组合和多裁剪图像推断的细节与（Szegedy et al，2014）类似。

We demonstrate in Fig. 4 that batch normalization allows us to set new state-of-the-art on the ImageNet classification challenge benchmarks.

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	up to 512	-	-	-	5.98%
MSRA multicrop	up to 480	-	-	-	5.71%
MSRA ensemble	up to 480	-	-	-	4.94%*
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.82%***

Figure 4. Batch-Normalized Inception comparison with previous state of the art on the provided validation set comprising 50000 images. Ensemble results are test server evaluation results on the test set. The BN-Inception ensemble has reached 4.9% top-5 error on the 50000 images of the validation set. All other reported results are on the validation set.

我们在图4中证实了批标准化使我们能够在ImageNet分类挑战基准上设置新的最佳结果。

Model	Resolution	Crops	Models	Top-1 error	Top-5 error
GoogLeNet ensemble	224	144	7	-	6.67%
Deep Image low-res	256	-	1	-	7.96%
Deep Image high-res	512	-	1	24.88	7.42%
Deep Image ensemble	up to 512	-	-	-	5.98%
MSRA multicrop	up to 480	-	-	-	5.71%
MSRA ensemble	up to 480	-	-	-	4.94%*
BN-Inception single crop	224	1	1	25.2%	7.82%
BN-Inception multicrop	224	144	1	21.99%	5.82%
BN-Inception ensemble	224	144	6	20.1%	**4.82%***

图4。批标准化Inception与以前的最佳结果在提供的包含5万张图像的验证集上的比较。组合结果是在测试集上由测试服务器评估的结果。BN-Inception组合在验证集的5万张图像上取得了`4.9% top-5`的错误率。所有报道的其它结果是在验证集上。

5. Conclusion

We have presented a novel mechanism for dramatically accelerating the training of deep networks. It is based on the premise that covariate shift, which is known to complicate the training of machine learning systems, also applies to sub-networks and layers, and removing it from internal activations of the network may aid in training. Our proposed method draws its power from normalizing activations, and from incorporating this normalization in the network architecture itself. This ensures that the normalization is appropriately handled by any optimization method that is being used to train the network. To enable stochastic optimization methods commonly used in deep network training, we perform the normalization for each mini-batch, and backpropagate the gradients through the normalization parameters. Batch Normalization adds only two extra paramters per activation, and in doing so preserves the representation ability of the network. We presented an algorithm for constructing, training, and performing inference with batch-normalized networks. The resulting networks can be trained with saturating nonlinearities, are more tolerant to increased training rates, and often do not require Dropout for regularization.

5. 结论

我们提出了一个新的机制，大大加快了深度网络的训练。它是基于前提协变量转移的，已知其会使机器学习系统的训练复杂化，也适用于子网络和层，并且从网络的内部激活中去除它可能有助于训练。我们提出的方法从其标准化激活中获取其功能，并将这种标准化合并到网络架构本身。这确保了标准化可以被用来训练网络的任何优化方法进行恰当的处理。为了让深度网络训练中常用的随机优化方法可用，我们对每个小批量数据执行标准化，并通过标准化参数来反向传播梯度。批标准化每个激活只增加了两个额外的参数，这样做可以保持网络的表示能力。我们提出了一个用于构建，训练和执行推断的批标准化网络算法。所得到的网络可以用饱和非线性进行训练，能更容忍增加的训练率，并且通常不需要丢弃来进行正则化。

Merely adding Batch Normalization to a state-of-the-art image classification model yields a substantial speedup in training. By further increasing the learning rates, removing Dropout, and applying other modifications afforded by Batch Normalization, we reach the previous state of the art with only a small fraction of training steps —— and then beat the state of the art in single-network image classification. Furthermore, by combining multiple models trained with Batch Normalization, we perform better than the best known system on ImageNet, by a significant margin.

仅仅将批标准化添加到了最新的图像分类模型中便在训练中取得了实质的加速。通过进一步提高学习率，删除丢弃和应用批标准化所提供的其它修改，我们只用了少部分的训练步骤就达到了以前的技术水平——然后在单

网络图像分类中击败了最先进的技术。此外，通过组合多个使用批标准化训练的模型，我们在ImageNet上的表现显著优于最好的已知系统。

Our method bears similarity to the standardization layer of (Gülçehre & Bengio, 2013), though the two address different goals. Batch Normalization seeks a stable distribution of activation values throughout training, and normalizes the inputs of a nonlinearity since that is where matching the moments is more likely to stabilize the distribution. On the contrary, the standardization layer is applied to the output of the nonlinearity, which results in sparser activations. We have not observed the nonlinearity inputs to be sparse, neither with nor without Batch Normalization. Other notable differences of Batch Normalization include the learned scale and shift that allow the BN transform to represent identity, handling of convolutional layers, and deterministic inference that does not depend on the mini-batch.

我们的方法与（Gülçehre＆Bengio，2013）的标准化层相似，尽管这两个方法解决的目标不同。批标准化寻求在整个训练过程中激活值的稳定分布，并且对非线性的输入进行归一化，因为这时更有可能稳定分布。相反，标准化层被应用于非线性的输出，这导致了更稀疏的激活。我们没有观察到非线性输入是稀疏的，无论是有批标准化还是没有批标准化。批标准化的其它显著差异包括学习到的缩放和转移允许BN变换表示恒等，卷积层处理以及不依赖于小批量数据的确定性推断。

In this work, we have not explored the full range of possibilities that Batch Normalization potentially enables. Our future work includes applications of our method to Recurrent Neural Networks (Pascanu et al., 2013), where the internal covariate shift and the vanishing or exploding gradients may be especially severe, and which would allow us to more thoroughly test the hypothesis that normalization improves gradient propagation (Sec. 3.3). More study is needed of the regularization properties of Batch Normalization, which we believe to be responsible for the improvements we have observed when Dropout is removed from BN-Inception. We plan to investigate whether Batch Normalization can help with domain adaptation, in its traditional sense —— i.e. whether the normalization performed by the network would allow it to more easily generalize to new data distributions, perhaps with just a recomputation of the population means and variances (Alg. 2). Finally, we believe that further theoretical analysis of the algorithm would allow still more improvements and applications.

在这项工作中，我们没有探索批标准化可能实现的全部可能性。我们的未来工作包括将我们的方法应用于循环神经网络（Pascanu et al.，2013），其中内部协变量转移和梯度消失或爆炸可能特别严重，这将使我们能够更彻底地测试假设标准化改善了梯度传播（第3.3节）。需要对批标准化的正则化属性进行更多的研究，我们认为这是BN-Inception中删除丢弃时我们观察到的改善的原因。我们计划调查批标准化是否有助于传统意义上的域自适应——即网络执行标准化是否能够更容易泛化到新的数据分布，也许仅仅是对总体均值和方差的重新计算（Alg.2）。最后，我们认为，该算法的进一步理论分析将允许更多的改进和应用。

Acknowledgments

We thank Vincent Vanhoucke and Jay Yagnik for help and discussions, and the reviewers for insightful comments.

致谢

我们感谢Vincent Vanhoucke和Jay Yagnik的帮助和讨论，以及审稿人的深刻评论。

References

Bengio, Yoshua and Glorot, Xavier. Understanding the difficulty of training deep feedforward neural networks. In Proceedings of AISTATS 2010, volume 9, pp. 249–256, May 2010.

Dean, Jeffrey, Corrado, Greg S., Monga, Rajat, Chen, Kai, Devin, Matthieu, Le, Quoc V., Mao, Mark Z., Ranzato, Marc'Aurelio, Senior, Andrew, Tucker, Paul, Yang, Ke, and Ng, Andrew Y. Large scale distributed deep networks. In NIPS, 2012.

Desjardins, Guillaume and Kavukcuoglu, Koray. Natural neural networks. (unpublished).

Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res., 12:2121–2159, July 2011. ISSN 1532-4435.

Gü lç ehre, Ç aglar and Bengio, Yoshua. Knowledge matters: Importance of prior information for optimization. CoRR, abs/1301.4083, 2013.

He, K., Zhang, X., Ren, S., and Sun, J. Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. ArXiv e-prints, February 2015.

Hyvä rinen, A. and Oja, E. Independent component analysis: Algorithms and applications. Neural Netw., 13(4-5): 411–430, May 2000.
Jiang, Jing. A literature survey on domain adaptation of statistical classifiers, 2008.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998a.

LeCun, Y., Bottou, L., Orr, G., and Muller, K. Efficient backprop. In Orr, G. and K., Muller (eds.), Neural Networks: Tricks of the trade. Springer, 1998b.

Lyu, S and Simoncelli, E P. Nonlinear image representation using divisive normalization. In Proc. Computer Vision and Pattern Recognition, pp. 1–8. IEEE Computer Society, Jun 23-28 2008. doi:

10.1109/CVPR.2008.4587821.

Nair, Vinod and Hinton, Geoffrey E. Rectified linear units improve restricted boltzmann machines. In ICML, pp. 807–814. Omnipress, 2010.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. In Proceedings of the 30th International Conference on Machine Learning, ICML 2013, Atlanta, GA, USA, 16-21 June 2013, pp. 1310–1318, 2013.

Povey, Daniel, Zhang, Xiaohui, and Khudanpur, Sanjeev. Parallel training of deep neural networks with natural gradient and parameter averaging. CoRR, abs/1410.7455, 2014.

Raiko, Tapani, Valpola, Harri, and LeCun, Yann. Deep learning made easier by linear transformations in perceptrons. In International Conference on Artificial Intelligence and Statistics (AISTATS), pp. 924–932, 2012.

Russakovsky, Olga, Deng, Jia, Su, Hao, Krause, Jonathan, Satheesh, Sanjeev, Ma, Sean, Huang, Zhiheng, Karpathy, Andrej, Khosla, Aditya, Bernstein, Michael, Berg, Alexander C., and Fei-Fei, Li. ImageNet Large Scale Visual Recognition Challenge, 2014.

Saxe, Andrew M., McClelland, James L., and Ganguli, Surya. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. CoRR, abs/1312.6120, 2013.

Shimodaira, Hidetoshi. Improving predictive inference under covariate shift by weighting the log-likelihood function. Journal of Statistical Planning and Inference, 90 (2):227–244, October 2000.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res., 15(1):1929–1958, January 2014.

Sutskever, Ilya, Martens, James, Dahl, George E., and Hinton, Geoffrey E. On the importance of initialization and momentum in deep learning. In ICML (3), volume 28 of JMLR Proceedings, pp. 1139–1147. JMLR.org, 2013.

Szegedy, Christian, Liu, Wei, Jia, Yangqing, Sermanet, Pierre, Reed, Scott, Anguelov, Dragomir, Erhan, Dumitru, Vanhoucke, Vincent, and Rabinovich, Andrew. Going deeper with convolutions. CoRR, abs/1409.4842, 2014.

Wiesler, Simon and Ney, Hermann. A convergence analysis of log-linear training. In Shawe-Taylor, J., Zemel, R.S., Bartlett, P., Pereira, F.C.N., and Weinberger, K.Q. (eds.), Advances in Neural Information Processing Systems 24, pp. 657–665, Granada, Spain, December 2011.

Wiesler, Simon, Richard, Alexander, Schlü ter, Ralf, and Ney, Hermann. Mean-normalized stochastic gradient for large-scale deep learning. In IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 180–184, Florence, Italy, May 2014.

Wu, Ren, Yan, Shengen, Shan, Yi, Dang, Qingqing, and Sun, Gang. Deep image: Scaling up image recognition, 2015.

Deep Learning

‹ Batch Normalization论文翻译——中文版

Python中的编码 ›