Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models

Sergey Ioffe
Google Inc., *sioffe@google.com*
翻译:smile(初)，管枫 (复)，seed(审)

Abstract

批规范化可以十分有效地加速和提升深度模型的训练。但是，当训练的 minibatch 较小或不是由独立的样本组成时，它的有效性会降低。我们假设这是由 minibatch 中所有样本上的模型层输入的依赖性和训练与推理之间产生的不同激活所造成的。我们提出了一种简单却有效的扩展——批再规范化，以确保训练和推理模型可以基于单个样本而非整个 minibatch 而生成同样的输出。当用小或 non-i.i.d.（非独立同分布）minibatch 训练时，用批再规范化训练的模型的表现显著优于批规范化训练的模型。与此同时，批再规范化还保留了批规范化的优点，例如对初始化不敏感和训练效率。Batch Normalization is quite effective at accelerating and improving the training of deep models. However, its effectiveness diminishes when the training minibatches are small, or do not consist of independent samples. We hypothesize that this is due to the dependence of model layer inputs on all the examples in the minibatch, and different activations being produced between training and inference. We propose Batch Renormalization, a simple and effective extension to ensure that the training and inference models generate the same outputs that depend on individual examples rather than the entire minibatch. Models trained with Batch Renormalization perform substantially better than batchnorm when training with small or non-i.i.d. minibatches. At the same time, Batch Renormalization retains the benefits of batchnorm such as insensitivity to initialization and training efficiency.

1 Introduction

批规范化近年来已经成为训练深度网络的标准工具包的一部分。通过规范化激活，批规范化帮助固定了作为模型训练的内部激活分布，使用了更高的学习率，减少了初始化敏感度。这些影响帮助加速了训练速度，有时有戏剧性的效果。批规范化成功应用于先进的架构比如残差网络。

Batch Normalization ("batchnorm" [6]) has recently become a part of the standard toolkit for training deep networks. By normalizing activations, batch normalization helps stabilize the distributions of internal activations as the model trains. Batch normalization also makes it possible to use significantly higher learning rates, and reduces the sensitivity to initialization. These effects help accelerate the training, sometimes dramatically so. Batchnorm has been successfully used to enable state-of-the-art architectures such as residual networks [5].

批规范化对 minibatch 样本做随机梯度训练，并使用这些 minibatch 样本的均值和方差去规范化激活。具体来说，考虑在深度网络中的一个特定节点，为每个输入样本产生一个标量值。给定 m 个样本的一个 minibatch 为 $\mathcal{B}$，考虑这个节点的值 $x_1 \dots x_m$。然后批规范化值如下所示：

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}}$$

Batchnorm works on minibatches in stochastic gradient training, and uses the mean and variance of the minibatch to normalize the activations. Specifically, consider a particular node in the deep network, producing a scalar value for each input example. Given a minibatch $\mathcal{B}$ of m examples, consider the values of this node, $x_1 \dots x_m$. Then batchnorm takes the form:

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}}$$

这里的 $\mu_{\mathcal{B}}$ 是 $x_1 \dots x_m$ 这 m 个样本的方差，$\sigma^2_{\mathcal{B}}$ 是样本的方差 (事实上，为了数值的稳定性增加了一个很小的值 ϵ)。可以清楚的看到，对于输入一个样本的规范化激活将取决于 minibatch 中的其他样本，这在推断阶段是不可取的，因此可以用对整个样本计算均值和方差来代替只使用 minibatch。在实践中，模型经常会对 minibatch 均值和方差维持一个移动平均，在推断阶段使用这些移动平均来代替 minibatch 统计。where $\mu_{\mathcal{B}}$ is the sample mean of $x_1 \dots x_m$, and $\sigma^2_{\mathcal{B}}$ is the sample variance (in practice, a small ϵ is added to it for numerical stability). It is clear that the normalized activations corresponding to an input example will depend on the other examples in the minibatch. This is undesirable during inference, and therefore the mean and variance computed over all training data can be used instead. In practice, the model usually maintains moving averages of minibatch means and variances, and during

inference uses those in place of the minibatch statistics.

虽然批规范化似乎在推断阶段用整个数据取代了 minibatch 统计，改变了网络中的激活。事实上，这意味着上层表示以 minibatch 规范化作为输入的训练阶段跟以样本的统计规范化作为输入的推断阶段不同。而当 minibatch 太大或者训练样本的分布是独立同分布的时候，这种差异是很小的，而实际更有利于一般化，但是，minibatch-wise 归一化可能会有一些明显的缺点。While it appears to make sense to replace the minibatch statistics with whole-data ones during inference, this changes the activations in the network. In particular, this means that the upper layers (whose inputs are normalized using the minibatch) are trained on representations different from those computed in inference (when the inputs are normalized using the population statistics). When the minibatch size is large and its elements are i.i.d. samples from the training distribution, this difference is small, and can in fact aid generalization. However, minibatch-wise normalization may have significant drawbacks:

对于小的 minibatch，均值和方差的估计变得不太精确。这些不确定性再加上深度，会降低最终模型结果的准确度。而且。由于被用作计算方差的每一个样本又被用在它自己的规范化过程中，这种被用在推断阶段的规范化仿射变换是一种不太好的近似。For *small minibatches*, the estimates of the mean and variance become less accurate. These inaccuracies are compounded with depth, and reduce the quality of resulting models. Moreover, as each example is used to compute the variance used in its own normalization, the normalization operation is less well approximated by an affine transform, which is what is used in inference.

非独立同分布 minibatch 可能对模型的规范化产生不利影响。例如，在一个度量学习的场景中，经常会偏差化 minibatch 样本包括显著相关的样本集。例如，对于一个大小为 32 的 minibatch，我们可能随机选择 16 个标签，然后从每个标签中选出两个样本。没有规范化的话，对于 minibatch 的损失计算与整个样本解耦，在我们的采样机制中可能会引入批内依赖，最差的情况下，会增加 minibatch 梯度的方差。有了规范化的话，在每层间的样本活动可能导致模型过拟合来适应那些特别的 minibatch 分布，而对于个别样本表现糟糕。*Non-i.i.d. minibatches* can have a detrimental effect on models with batchnorm. For example, in a metric learning scenario (e.g. [4]), it is common to bias the minibatch sampling to include sets of examples that are known to be related. For instance, for a minibatch of size 32, we may randomly select 16 labels, then choose 2 examples for each of those labels. Without batchnorm, the loss computed for the minibatch decouples over the examples, and the intra-batch dependence introduced by our sampling mechanism may, at worst, increase the variance of the minibatch gradient. With batchnorm, however, the examples interact at every layer, which may cause the model to overfit to the specific distribution of minibatches and suffer when used on individual examples. 依赖于整个 minibatch 批规范化激活使批规范化更有效，但是这也是它的缺陷来源。已经提出几种方法来减弱它的缺点。但是，不像批规范化那样易于被应用在网络中，那些方法可能要求仔细的非线性分析与 minibatch 中心结合，可能改变模型的函数表示类别。另一个方法是 gan，使用一个独立固定的 minibatch 来计算规范化参数，但是，它使训练代价更大。The dependence of the batch-normalized activations on the entire minibatch makes batchnorm powerful, but it is also the source of its drawbacks. Several approaches [1, 2, 10] have been proposed to alleviate this. However, unlike batchnorm which can be easily applied to a network, these methods may require careful analysis of nonlinearities, may need to be combined with minibatch centering, and may change the class of functions representable by the model. Another alternative [9] is to use a separate and fixed minibatch to compute the normalization parameters, but this makes the training more expensive. 本文中，我们提出批再规范化，对于批规范化的一种新扩展。我们的方法保证与推断过程的激活计算一样，在训练步骤的前趟激活计算只依赖于单个样本。这明显的改善了小 minibatch 和非独立同分布的训练，跟批规范化相比，没有额外的开销。In this paper we propose *Batch Renormalization*, a new extension to batchnorm. Our method ensures that the activations computed in the forward pass of the training step depend only on a single example and are identical to the activations computed in inference. This significantly improves the training on non-i.i.d. or small minibatches, compared to batchnorm, without incurring extra cost.

2 Prior Work: Batch Normalization

先前工作：批规范化我们对深度网络随机梯度优化有兴趣，这个任务是为了最小化误差，分解训练样本。

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

We are interested in stochastic gradient optimization of deep networks. The task is to minimize the loss, which decomposes over training examples:

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

这里 ℓ_i 是训练样本的损失，Θ 是模型权重向量。在每个训练步骤，m 个训练样本的一个 minibatch 被用来计算梯度

$$\frac{1}{m} \frac{\partial \ell_i(\Theta)}{\partial \Theta}$$

，对梯度的优化来调整 Θ。where ℓ_i is the loss incurred on the ith training example, and Θ is the vector of model weights. At each training step, a minibatch of m examples is used to compute the gradient

$$\frac{1}{m}\frac{\partial \ell_i(\Theta)}{\partial \Theta}$$

which the optimizer uses to adjust Θ.

考虑深度网络中的一个特定节点 x。我们注意到计算 x 依赖所有模型参数，模型参数一改变，x 的分布也会改变。由于它自己影响所有在它上层的损失，分布的改变会导致以上层的训练复杂化，这种被称为内部变量转移。Consider a particular node x in a deep network. We observe that x depends on all the model parameters that are used for its computation, and when those change, the distribution of x also changes. Since x itself affects the loss through all the layers above it, this change in distribution complicates the training of the layers above. This has been referred to as internal covariate shift.

批规范化强调。考虑到一个 minibatch$\mathcal{B} = \{x_{1...m}\}$ 的值 x，然后按照以下公式规范化它们：Batch Normalization [6] addresses it by considering the values of x in a minibatch $\mathcal{B} = \{x_{1...m}\}$. It then normalizes them as follows:

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}} \leftarrow \sqrt{\frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 + \epsilon}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}(x_i)$$

这里的 γ 和 β 是训练参数 (在同一过程中学习，例如随机梯度下降，跟所有其他的模型权重一样)，ϵ 是一个很小的常数。最重要的是，样本均值 $\mu_{\mathcal{B}}$ 和样本标准差的计算是模型结构的一部分，它们是模型参宿自己的函数，参与反向传播。对于对于 batchnorm 传播公式推导简单的链式法则是引用了 citebatchnorm。Here γ and β are trainable parameters (learned using the same procedure, such as stochastic gradient descent, as all the other model weights), and ϵ is a small constant. Crucially, the computation of the sample mean $\mu_{\mathcal{B}}$ and sample standard deviation $\sigma_{\mathcal{B}}$ are part of the model architecture, are themselves functions of the model parameters, and as such participate in backpropagation. The backpropagation formulas for batchnorm are easy to derive by chain rule and are given in [6].

当应用批规范化到一个激活层 x，规范化产生各个维度独立 (或者，卷积中每个通道特征映射) When applying batchnorm to a layer of activations x, the normalization takes place independently for each dimension (or, in the convolutional case, for each channel or feature map). 当 x 它自己是对前一层进行线性变化的结果，批规范化使模型保持不变对于 W 的规模 (忽略小 ϵ)，这种不变形使权重初始化可能不被挑选，并使用更大的学习率。When x is itself a result of applying a linear transform W to the previous layer, batchnorm makes the model invariant to the scale of W (ignoring the small ϵ). This invariance makes it possible to not be picky about weight initialization, and to use larger learning rates.

除了内部变量转变的减少，批规范化的另一个影响，直觉可以通过看梯度计算在不同层间的方向传播。考虑规范化层 $\hat{\mathrm{x}}$，元素可能有 0 均值或者 1 方差。对于一个思想实验，让我们假设 $\hat{\mathrm{x}}$ 的维度是高斯独立的，而且，让我们将损失 ℓ 做一个线性函数近似 $\hat{\mathrm{x}}$: $\ell = \mathrm{g}^T\hat{\mathrm{x}}$，其中 $\mathrm{g} = \frac{\partial \ell}{\partial \hat{\mathrm{x}}}$. 随后 $\|\mathrm{g}\|^2 = \mathrm{Var}[\ell]$ 只取决于模型的损失，而不是我们挑选的层。这意味着梯度的规范化 w.r.t. $\|\frac{\partial \ell}{\partial \hat{\mathrm{x}}}\|$ 规范化层跟不同的规范化层相同。因此，梯度将流经整个网络，不发生梯度爆炸也不发生梯度消失，从而促进训练。独立性假设中，高斯和线性在实践中表现不好，梯度流实际上明显的改变了批规范化模型 Besides the reduction of internal covariate shift, an intuition for another effect of batchnorm can be obtained by looking at the gradients computed by backpropagation for different layers. Consider the normalized layer $\hat{\mathrm{x}}$, whose elements all have zero mean and unit variance. For a thought experiment, let us assume that the dimensions of $\hat{\mathrm{x}}$ are Gaussian and independent. Further, let us approximate the loss ℓ as a linear function of $\hat{\mathrm{x}}$: $\ell = \mathrm{g}^T\hat{\mathrm{x}}$, where $\mathrm{g} = \frac{\partial \ell}{\partial \hat{\mathrm{x}}}$. It then follows that $\|\mathrm{g}\|^2 = \mathrm{Var}[\ell]$ which only depends on the model loss and not the layer we picked. This means that the norm of the gradient w.r.t. a normalized layer $\|\frac{\partial \ell}{\partial \hat{\mathrm{x}}}\|$ is the same for different normalized layers. Therefore the gradients, as they flow through the network, do not explode nor vanish, thus facilitating the training. While the assumptions of independence, Gaussianity and linearity do not hold in practice, the gradient flow is in fact significantly improved in batch-normalized models.

在推断阶段，标准的做法是使用移动平均 μ, σ^2 而不是 minibatch 的均值 $\mu_{\mathcal{B}}$ 方差 $\sigma^2_{\mathcal{B}}$ 来规范化激活。During inference, the standard practice is to normalize the activations using the moving averages μ, σ^2 instead of minibatch mean $\mu_{\mathcal{B}}$ and variance $\sigma^2_{\mathcal{B}}$:

$$y_{\text{inference}} = \frac{x - \mu}{\sigma} \cdot \gamma + \beta$$

which depends only on a single input example rather than requiring a whole minibatch. 这是很自然的的一个问题，是否我们可以简单的使用移动平均 μ, σ 来表示训练过程的规范化，既然它将杉树规范化激活对于 minibatch 中另一个样本的依赖。但是，已经观察到这样将导致模型爆炸。正如批规范化讨论的洋洋，使用移动平均将导致梯度优化与规范化相互抵消。例如，梯度步骤可能增加一个偏差或者缩放的卷积权重，尽管事实上规范化将消除这些变化对损失的影响。这将导致无限增长的模型参数，但实

际上没有改善损失。因此，至关重要的是使用 mini-batch 的时刻和通过他们反向传播。It is natural to ask whether we could simply use the moving averages μ, σ to perform the normalization during training, since this would remove the dependence of the normalized activations on the other example in the minibatch. This, however, has been observed to lead to the model blowing up. As argued in [6], such use of moving averages would cause the gradient optimization and the normalization to counteract each other. For example, the gradient step may increase a bias or scale the convolutional weights, in spite of the fact that the normalization would cancel the effect of these changes on the loss. This would result in unbounded growth of model parameters without actually improving the loss. It is thus crucial to use the minibatch moments, and to backpropagate through them.

3 Batch Renormalization

With batchnorm, the activities in the network differ between training and inference, since the normalization is done differently between the two models. Here, we aim to rectify this, while retaining the benefits of batchnorm.

Let us observe that if we have a minibatch and normalize a particular node x using either the minibatch statistics or their moving averages, then the results of these two normalizations are related by an affine transform. Specifically, let μ be an estimate of the mean of x, and σ be an estimate of its standard deviation, computed perhaps as a moving average over the last several minibatches. Then, we have:

$$\frac{x_i - \mu}{\sigma} = \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} \cdot r + d, \quad \text{where } r = \frac{\sigma_{\mathcal{B}}}{\sigma}, \quad d = \frac{\mu_{\mathcal{B}} - \mu}{\sigma}$$

If $\sigma = \mathrm{E}[\sigma_{\mathcal{B}}]$ and $\mu = \mathrm{E}[\mu_{\mathcal{B}}]$, then $\mathrm{E}[r] = 1$ and $\mathrm{E}[d] = 0$ (the expectations are w.r.t. a minibatch $\mathcal{B}$). Batch Normalization, in fact, simply sets $r = 1$, $d = 0$.

We propose to retain r and d, but treat them as constants for the purposes of gradient computation. In other words, we augment a network, which contains batch normalization layers, with a per-dimension affine transformation applied to the normalized activations. We treat the parameters r and d of this affine transform as fixed, even though they were computed from the minibatch itself. It is important to note that this transform is identity in expectation, as long as $\sigma = \mathrm{E}[\sigma_{\mathcal{B}}]$ and $\mu = \mathrm{E}[\mu_{\mathcal{B}}]$. We refer to batch normalization augmented with this affine transform as *Batch Renormalization*: the fixed (for the given minibatch) r and d correct for the fact that the minibatch statistics differ from the population ones. This allows the above layers to observe the "correct" activations – namely, the ones that would be generated by the inference model.

Input: Values of x over a training mini-batch $\mathcal{B} = \{x_{1\ldots m}\}$; parameters γ, β; current moving mean μ and standard deviation σ; moving average update rate Δ; maximum allowed correction $r_{\max}$, $d_{\max}$.
Output: $\{y_i = \mathrm{BatchRenorm}(x_i)\}$; updated μ, σ.

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$$

$$\sigma_{\mathcal{B}} \leftarrow \sqrt{\epsilon + \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2}$$

$$r \leftarrow \texttt{stop\_gradient}\left(\mathrm{clip}_{[1/r_{\max}, r_{\max}]}\left(\frac{\sigma_{\mathcal{B}}}{\sigma}\right)\right)$$

$$d \leftarrow \texttt{stop\_gradient}\left(\mathrm{clip}_{[-d_{\max}, d_{\max}]}\left(\frac{\mu_{\mathcal{B}} - \mu}{\sigma}\right)\right)$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sigma_{\mathcal{B}}} \cdot r + d$$

$$y_i \leftarrow \gamma\,\widehat{x}_i + \beta$$

$$\mu := \mu + \Delta(\mu_{\mathcal{B}} - \mu) \qquad \text{// Update moving averages}$$

$$\sigma := \sigma + \Delta(\sigma_{\mathcal{B}} - \sigma)$$

Inference: $\quad y \leftarrow \gamma \cdot \dfrac{x - \mu}{\sigma} + \beta$

Algorithm 1: *Training (top) and inference (bottom) with Batch Renormalization, applied to activation x over a mini-batch. During backpropagation, standard chain rule is used. The values marked with* `stop_gradient` *are treated as constant for a given training step, and the gradient is not propagated through them.*

In practice, it is beneficial to train the model for a certain number of iterations with batchnorm alone, without the correction, then ramp up the amount of allowed correction. We do this by imposing bounds on r and d, which initially constrain them to 1 and 0, respectively, and then are gradually relaxed.

Algorithm 1 presents Batch Renormalization. Unlike batchnorm, where the moving averages are computed during training but used only for inference, Batch Renorm does use μ and σ during training to perform the correction. We use a fairly high rate of update Δ for these averages, to ensure that they benefit from averaging multiple batches but do not become stale relative to the model parameters. We explicitly update the exponentially-decayed moving averages μ and σ, and optimize the rest of the model using gradient optimization, with the gradients cal-

culated via backpropagation:

$$\frac{\partial \ell}{\partial \widehat{x}_i} = \frac{\partial \ell}{\partial y_i} \cdot \gamma$$
$$\frac{\partial \ell}{\partial \sigma_{\mathcal{B}}} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot (x_i - \mu_{\mathcal{B}}) \cdot \frac{-r}{\sigma_{\mathcal{B}}^2}$$
$$\frac{\partial \ell}{\partial \mu_{\mathcal{B}}} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{-r}{\sigma_{\mathcal{B}}}$$
$$\frac{\partial \ell}{\partial x_i} = \frac{\partial \ell}{\partial \widehat{x}_i} \cdot \frac{r}{\sigma_{\mathcal{B}}} + \frac{\partial \ell}{\partial \sigma_{\mathcal{B}}} \cdot \frac{x_i - \mu_{\mathcal{B}}}{m\sigma_{\mathcal{B}}} + \frac{\partial \ell}{\partial \mu_{\mathcal{B}}} \cdot \frac{1}{m}$$
$$\frac{\partial \ell}{\partial \gamma} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i} \cdot \widehat{x}_i$$
$$\frac{\partial \ell}{\partial \beta} = \sum_{i=1}^{m} \frac{\partial \ell}{\partial y_i}$$

These gradient equations reveal another interpretation of Batch Renormalization. Because the loss ℓ is unaffected when all x_i are shifted or scaled by the same amount, the functions $\ell(\{x_i + t\})$ and $\ell(\{x_i \cdot (1 + t)\})$ are constant in t, and computing their derivatives at $t = 0$ gives $\sum_{i=1}^{m} \frac{\partial \ell}{\partial x_i} = 0$ and $\sum_{i=1}^{m} x_i \frac{\partial \ell}{\partial x_i} = 0$. Therefore, if we consider the m-dimensional vector $\left\{\frac{\partial \ell}{\partial x_i}\right\}$ (with one element per example in the minibatch), and further consider two vectors $\mathrm{p}_0 = (1, \ldots, 1)$ and $\mathrm{p}_1 = (x_1, \ldots, x_m)$, then $\left\{\frac{\partial \ell}{\partial x_i}\right\}$ lies in the null-space of p_0 and p_1. In fact, it is easy to see from the Batch Renorm backprop formulas that to compute the gradient $\left\{\frac{\partial \ell}{\partial x_i}\right\}$ from $\left\{\frac{\partial \ell}{\partial \widehat{x}_i}\right\}$, we need to first scale the latter by $r/\sigma_{\mathcal{B}}$, then project it onto the null-space of p_0 and p_1. For $r = \frac{\sigma_{\mathcal{B}}}{\sigma}$, this is equivalent to the backprop for the transformation $\frac{x-\mu}{\sigma}$, but combined with the null-space projection. In other words, Batch Renormalization allows us to normalize using moving averages μ, σ in training, and makes it work using the extra projection step in backprop.

Batch Renormalization shares many of the beneficial properties of batchnorm, such as insensitivity to initialization and ability to train efficiently with large learning rates. Unlike batchnorm, our method ensures that that all layers are trained on internal representations that will be actually used during inference.

4 Results

To evaluate Batch Renormalization, we applied it to the problem of image classification. Our baseline model is Inception v3 [12], trained on 1000 classes from ImageNet training set [8], and evaluated on the ImageNet validation data. In the baseline model, batchnorm was used after convolution and before the ReLU [7]. To apply Batch Renorm, we simply swapped it into the model in place of batchnorm. Both methods normalize each feature map over examples as well as over spatial locations. We fix the scale $\gamma = 1$, since it could be propagated through the ReLU and absorbed into the next layer.

The training used 50 synchronized workers [3]. Each worker processed a minibatch of 32 examples per training step. The gradients computed for all 50 minibatches were aggregated and then used by the RMSProp optimizer [13]. As is common practice, the inference model used exponentially-decayed moving averages of all model parameters, including the μ and σ computed by both batchnorm and Batch Renorm.

For Batch Renorm, we used $r_{\max} = 1$, $d_{\max} = 0$ (i.e. simply batchnorm) for the first 5000 training steps, after which these were gradually relaxed to reach $r_{\max} = 3$ at 40k steps, and $d_{\max} = 5$ at 25k steps. These final values resulted in clipping a small fraction of rs, and none of ds. However, at the beginning of training, when the learning rate was larger, it proved important to increase $r_{\max}$ slowly: otherwise, large gradients were observed to suddenly and severely increase the loss. We have found clipping to be less important for those Batch Renorm layers that are followed by another normalization layer. The intuition is that for such layers, scaling up all the corrections r will leave the model output and gradients unchanged since the next normalization layer will cancel out the effects the larger correction.

All the hyperparameters other than those related to normalization were fixed between the models and across experiments.

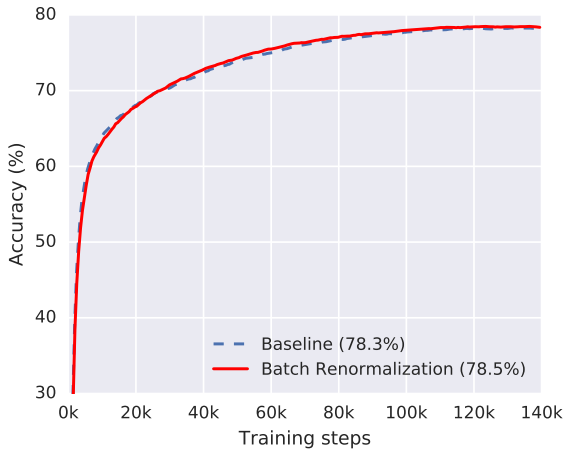
4.1 Baseline

As a baseline, we trained the batchnorm model using the minibatch size of 32. More specifically, batchnorm was applied to each of the 50 minibatches; each example was normalized using 32 examples, but the resulting gradients were aggregated over 50 minibatches. This model achieved the top-1 validation accuracy of 78.3% after 130k training steps.

To verify that Batch Renorm does not diminish performance on such minibatches, we also trained the model with Batch Renorm, see Figure 1. The test accuracy of this model closely tracked the baseline, achieving a slightly higher test accuracy (78.5%) after the same number of steps.

4.2 Small minibatches

To investigate the effectiveness of Batch Renorm when training on small minibatches, we reduced the number of examples used for normalization to 4. Each minibatch of size 32 was thus broken into "microbatches" each having 4 examples; each microbatch was normalized independently, but the loss for each minibatch was computed as before. In other words,



分别使用批规范化和批再规范化的 Inception-v3 模型的验证 top-1 准确度（validation top-1 accuracy），模型是在 50 个同步的工作器（worker）上训练的，其中每一个工作器处理大小为 32 的 minibatch。使用批再规范化的模型实现了略高一点的验证准确度。

Figure 1: *Validation top-1 accuracy of Inception-v3 model with batchnorm and its Batch Renorm version, trained on 50 synchronized workers, each processing minibatches of size 32. The Batch Renorm model achieves a marginally higher validation accuracy.*

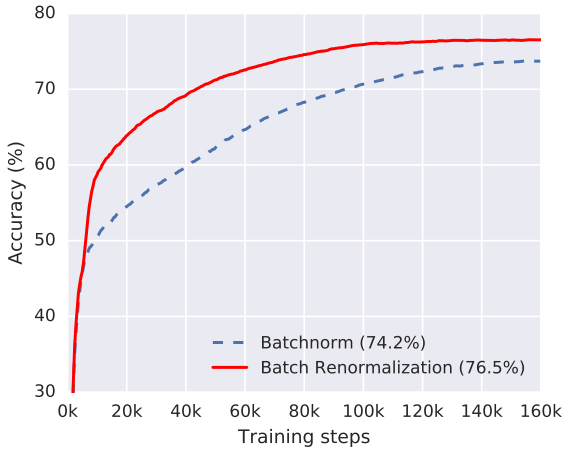
the gradient was still aggregated over 1600 examples per step, but the normalization involved groups of 4 examples rather than 32 as in the baseline. Figure 2 shows the results.

The validation accuracy of the batchnorm model is significantly lower than the baseline that normalized over minibatches of size 32, and training is slow, achieving 74.2% at 210k steps. We obtain a substantial improvement much faster (76.5% at 130k steps) by replacing batchnorm with Batch Renorm, However, the resulting test accuracy is still below what we get when applying either batchnorm or Batch Renorm to size 32 minibatches. Although Batch Renorm improves the training with small minibatches, it does not eliminate the benefit of having larger ones.

4.3 Non-i.i.d. minibatches

When examples in a minibatch are not sampled independently, batchnorm can perform rather poorly. However, sampling with dependencies may be necessary for tasks such as for metric learning [4, 11]. We may want to ensure that images with the same label have more similar representations than otherwise, and to learn this we require that a reasonable number of same-label image pairs can be found within the same minibatch.

In this experiment (Figure 3), we selected each



分别使用批规范化和批再规范化的模型的验证准确度，其中规范化是在有 4 个样本的集合上执行的（但带有被 50 个工作器处理过的在所有 50×32 个样本上聚集的梯度）。批再规范化让模型能更快地训练和实现更高的准确度，尽管规范化 32 个样本的集合会表现更好。

Figure 2: *Validation accuracy for models trained with either batchnorm or Batch Renorm, where normalization is performed for sets of 4 examples (but with the gradients aggregated over all* 50×32 *examples processed by the 50 workers). Batch Renorm allows the model to train faster and achieve a higher accuracy, although normalizing sets of 32 examples performs better.*

minibatch of size 32 by randomly sampling 16 labels (out of the total 1000) with replacement, then randomly selecting 2 images for each of those labels. When training with batchnorm, the test accuracy is much lower than for i.i.d. minibatches, achieving only 67%. Surprisingly, even the *training* accuracy is much lower (72.8%) than the *test* accuracy in the i.i.d. case, and in fact exhibits a drop that is consistent with overfitting. We suspect that this is in fact what happens: the model learns to predict labels for images that come in a set, where each image has a counterpart with the same label. This does not directly translate to classifying images individually, thus producing a drop in the accuracy computed on the training data. To verify this, we also evaluated the model in the "training mode", i.e. using minibatch statistics $\mu_{\mathcal{B}}$, $\sigma_{\mathcal{B}}$ instead of moving averages μ, σ, where each test minibatch had size 50 and was obtained using the same procedure as the training minibatches – 25 labels, with 2 images per label. As expected, this does much better, achieving 76.5%, though still below the baseline accuracy. Of course, this evaluation scenario is usually infeasible, as we want the image representation to be a deterministic function of that image alone.

We can improve the accuracy for this problem by splitting each minibatch into two halves of size 16 each, so that for every pair of images belonging to the same class, one image is assigned to the first half-minibatch, and the other to the second. Each half is then more i.i.d., and this achieves a much better test accuracy (77.4% at 140k steps), but still below the baseline. This method is only applicable when the number of examples per label is small (since this determines the number of microbatches that a minibatch needs to be split into).

With Batch Renorm, we simply trained the model with minibatch size of 32. The model achieved the same test accuracy (78.5% at 120k steps) as the equivalent model on i.i.d. minibatches, vs. 67% obtained with batchnorm. By replacing batchnorm with Batch Renorm, we ensured that the inference model can effectively classify individual images. This has completely eliminated the effect of overfitting the model to image sets with a biased label distribution.

5 Conclusions

我们已经证明，批规范化虽然有效，但不适合小 minibatch 或者非独立同分布训练样本。我们假设这些缺点是由于在实际中，模型的激活反过来作为其他层的输入，梯度计算训练过程跟推断过程不同。我们提出了批再规范化，替代批规范化模型的输出计算在训练和推断阶段都只仅仅依赖于个别样本而不是整个的 minibatch。We have demonstrated that Batch Normalization, while effective, is not well suited to small or non-i.i.d. training minibatches. We hypothesized that these drawbacks are due to the fact that the activations in the model, which are in turn used by other layers as inputs, are computed differently during training than during inference. We address this with Batch Renormalization, which replaces batchnorm and ensures that the outputs computed by the model are dependent only on the individual examples and not the entire minibatch, during both training and inference. 批再规范化对批规范化进行了每维纠正扩展，来保证训练网络和推断网络的激活匹配。这种修正期待是理想的。参数从 minibatch 中被计算但是被当做一个常数优化。不像批规范化，推断过程均值和方差不需要计算直到训练完成，批再规范化受益于让统计数据直接参与训练。与批规范化一样，批再规范化易于实现，训练和推断速度也一样，明显的改善了小 minibatch 和非独立同分布 minibatch 的训练。我们的方法有额外的超参数：移动平均的更新率 Δ，纠正限制时刻表 $d_{\max}$，$r_{\max}$。对于这些超参数的影响更深入的调查是未来工作的一部分。Batch Renormalization extends batchnorm with a per-dimension correction to ensure that the activations match between the training and inference networks. This correction is identity in expectation; its parameters are computed from the minibatch but are treated as constant by the optimizer. Unlike batchnorm, where the means and variances used during inference do not need to be computed until the training has completed, Batch Renormalization benefits from having these statistics directly participate in the training. Batch Renormalization is as easy to implement as batchnorm itself, runs at the same speed during both training and inference, and significantly improves training on small or non-i.i.d. minibatches. Our method does have extra hyperparameters: the update rate Δ for the moving averages, and the schedules for correction limits $d_{\max}$, $r_{\max}$. A more extensive investigation of the effect of these is a part of future work.

批再规范化承诺会改善任何正常使用批规范化模型的性能。包括残差网络，另一个应用是生成式对抗网络，批规范化对于非确定性的引入被发现是个问题，而批再规范化可能提供了这个问题的一个解决方案。Batch Renormalization offers a promise of improving the performance of any model that would normally use batchnorm. This includes Residual Networks [5]. Another application is Generative Adversarial Networks [9], where the non-determinism introduced by batchnorm has been found to be an issue, and Batch Renorm may provide a solution.

最后，批再规范化可能对那些应用批规范化困难的应用有效，例如递归神经网络。批规范化要求每个时间戳独立正则化，而批再规范化让用同样的运行均值来规范化整个时间戳成为可能，然后使用所有时间戳来更新那些均值。这仍然是未来值得深入探索的一个领域。Finally, Batch Renormalization may benefit applications where applying batch normaliza-

tion has been difficult – such as recurrent networks. There, batchnorm would require each timestep to be normalized independently, but Batch Renormalization may make it possible to use the same running averages to normalize all timesteps, and then update those averages using all timesteps. This remains one of the areas that warrants further exploration.

References

[1] D. Arpit, Y. Zhou, B. U. Kota, and V. Govindaraju. Normalization propagation: A parametric technique for removing internal covariate shift in deep networks. *arXiv preprint arXiv:1603.01431*, 2016.

[2] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] J. Chen, R. Monga, S. Bengio, and R. Jozefowicz. Revisiting distributed synchronous sgd. *arXiv preprint arXiv:1604.00981*, 2016.

[4] J. Goldberger, S. Roweis, G. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In *Advances in Neural Information Processing Systems 17*, 2004.

[5] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[6] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 448–456, 2015.

[7] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, pages 807–814. Omnipress, 2010.

[8] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

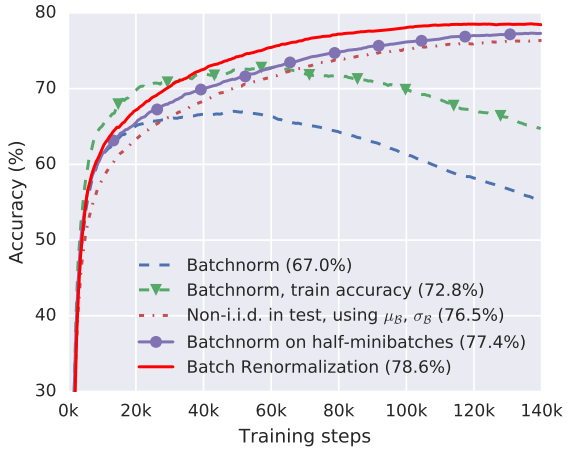
[9] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2226–2234, 2016.

[10] T. Salimans and D. P. Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 901–901, 2016.

[11] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[12] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2818–2826, 2016.

[13] T. Tieleman and G. Hinton. Lecture 6.5 - rmsprop. COURSERA: Neural Networks for Machine Learning, 2012.



当在 non-i.i.d. minibatch 上训练时所得到的验证准确度，这是通过为 16 个（取自共 1000 个）随机标签中的每一个标签采样 2 张图像得到的。这种分布偏差不仅会导致测试的低准确度，也会导致在训练集上的低准确度，最后还会下降。这表明在这个特定的 minibatch 分布上出现了过拟合，这通过我们的改进得到了证实——当测试 minibatch 也是每个标签包含 2 张图像，并且批规范化在推理过程中使用了 minibatch 统计 μB、 B 时。如果批规范化被分别应用于一个训练 minibatch 的两半，使其中每一半都更 i.i.d.，那么它还会进一步提升。最后，通过使用批再规范化，我们可以仅通过正常的训练和评估就实现如我们在图 1 中的 i.i.d. minibatch 上所得到的同样的验证准确度。

Figure 3: *Validation accuracy when training on non-i.i.d. minibatches, obtained by sampling 2 images for each of 16 (out of total 1000) random labels. This distribution bias results not only in a low test accuracy, but also low accuracy on the training set, with an eventual drop. This indicates overfitting to the particular minibatch distribution, which is confirmed by the improvement when the test minibatches also contain 2 images per label, and batchnorm uses minibatch statistics $\mu_{\mathcal{B}}$, $\sigma_{\mathcal{B}}$ during inference. It improves further if batchnorm is applied separately to 2 halves of a training minibatch, making each of them more i.i.d. Finally, by using Batch Renorm, we are able to just train and evaluate normally, and achieve the same validation accuracy as we get for i.i.d. minibatches in Fig. 1.*