Unsupervised Batch Normalization

Citation for published version:

Digital Object Identifier (DOI):
10.1109/CVPRW50498.2020.00467

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)

General rights
Copyright for the publications made accessible via the Edinburgh Research Explorer is retained by the author(s) and / or other copyright owners and it is a condition of accessing these publications that users recognise and abide by the legal requirements associated with these rights.

Take down policy
The University of Edinburgh has made every reasonable effort to ensure that Edinburgh Research Explorer content complies with UK legislation. If you believe that the public display of this file breaches copyright please contact openaccess@ed.ac.uk providing details, and we will remove access to the work immediately and investigate your claim.
Unsupervised Batch Normalization

Mustafa Taha Koçyiğit, Laura Sevilla-Lara, Timothy M. Hospedales, Hakan Bilen
University of Edinburgh, UK
{taha.kocyigit,l.sevilla,t.hospedales,h.bilen}@ed.ac.uk

Abstract

Batch Normalization is a widely used tool in neural networks to improve the generalization and convergence of training. However, on small datasets due to the difficulty of obtaining unbiased batch statistics it cannot be applied effectively. In some cases, even if there is only a small labeled dataset available, there are larger unlabeled datasets from the same distribution. We propose using such unlabeled examples to calculate batch normalization statistics, which we call Unsupervised Batch Normalization (UBN). We show that using unlabeled examples for batch statistic calculations results in a reduction of the bias of the statistics, as well as regularization leveraging the data manifold. UBN is easy to implement, computationally inexpensive and can be applied to a variety of problems. We report results on monocular depth estimation, where obtaining dense labeled samples is difficult and expensive. Using unlabeled samples, and UBN, we obtain an increase in accuracy of more than 6% on the KITTI dataset, compared to using traditional batch normalization only on the labeled samples.

1. Introduction

Large-scale labeled data and modern learnable representations have driven progress on many computer vision problems in the last decade. The dependency on labeled data is a fundamental bottleneck for many vision problems, as obtaining ground truth annotations can either be extremely expensive (e.g., pixel-wise annotations for semantic segmentation) or require specialized equipment and controlled environments (e.g., depth estimation, 3D pose estimation, optical flow).

In these cases, research often resorts to different strategies. A common approach is to pre-train from scratch a deep network on “realistic” synthetic data and then fine-tune on the available labeled data. This strategy is commonly used for example by state-of-the-art optical flow methods [24]. While this method alleviates overfitting, it requires creating such realistic data, and does not provide the opportunity to leverage the availability of unlabeled data. While data may be difficult and expensive to label, sometimes additional unlabeled data samples (e.g., images from the same distribution) can be easily available for free. Another popular technique is transfer learning, where large models are pre-trained on self-supervised tasks (e.g., [32]) and then fine-tuned on the small amount of labeled data available. Although the pre-training stage helps the optimization starting point, the fine-tuning stage still risks overfitting to the small amount of labeled data. Another effective strategy is semi-supervised learning [34], which aims at learning both from labeled and unlabeled data at the same time. This approach is successful but is usually not general, and requires specific adaptation to each vision problem.

In this paper we propose a method that is general, does not require adaptation to any new vision problem, leverages additional unlabeled data and does not require fine-tuning. In particular, we adapt the widely used batch normalization (BN) technique to use unlabeled data.

BN is widely used on many architectures to normalize the statistics of the features, and therefore stabilize the optimization process. However, these statistics can also overfit to the labeled dataset and not reflect the statistics that would normalize the actual dataset. This would cause new images to appear as out of distribution examples for the model.

Instead, we use unlabeled samples, which typically are a much larger number than the labeled ones, to compute the normalization statistics, in a process we call Unsupervised Batch Normalization (UBN). We test our method in the problem of monocular depth estimation, and obtain more than 6% improvement over the baseline of using vanilla batch normalization. We hope that the simplicity of our method will make it a useful component of future learning systems.

2. Related Work

Normalization methods. Batch normalization [13] is the most standard normalization method in the state-of-the-art classification architectures on the ImageNet LSVRC [25, 30]. Recently, new alternatives have been developed to extend its scope in different ways and making BN more general. Batch Renormalization [12] enables training with
smaller batch sizes and correlated batches by calculating moving averages of batch statistics during training. Mode Norm [6] enables the use of BN for multi-modal data. This is done by calculating multiple batch statistics and adaptively normalizing examples through a gating function and calculating weighted normalization statistic for each example. Instance Norm [28] has been used for style transfer and normalizes the style of the content image [11] by calculating statistics over each channel and sample in the dataset. Adaptive Instance Norm [11] transfers style between two images by calculating the batch statistics form one example and applying them to another image. Weight Norm [22] conditions the optimization problem by reparameterizing the weight values into magnitude and direction while Layer Normalization [2] calculates normalization statistics per example for feature maps in each layer to enable normalization in recurrent models.

While all these variants make BN more usable and stable, to our knowledge, our is the first to extend BN to leverage unlabeled data.

Understanding Batch Normalization. While BN has been widely and successfully used, there remains some questions about why and when it works. The initial explanation of Ioffe and Szegedy [13] that BN reduces internal co-variate shift has been questioned and explored in recent work.

For example, Bjorck et. al. [4] argues that BN enables training with larger learning rates by preventing exploding and vanishing gradients. They also show that unnormalized networks have ill-conditioned activations due to random initialization which results in large singular values of the activation matrices and predictions which are independent from the inputs. Similarly Santurkar et. al. [23] argue that the loss landscape of the normalized networks are smoother. Luo et. al. [18] showed that batch normalization applies population normalization with data dependant gamma decay that promotes uncorrelated features.

Monte Carlo Batch Normalization [27] shows that uncertainty estimates can be obtained from networks with BN layers by sampling batch statistics.

Training on small datasets. Training the classification networks on ImageNet requires millions of balanced labeled images which many other vision problems lack. Many semi-supervised methods that are designed for classification problems [3, 29, 17, 26] are not readily applicable to regression due to the difficulty in generating pseudo-labels and augmentation strategies. Training unsupervised methods like geometric consistency based unsupervised training [16, 9, 1] with supervised training in order to mitigate labeled data requirements has also results in problem specific solutions. While pre-training is a widely used technique in segmentation and detection [5] when the target tasks is not similar to the source task the transfer learning has limited success [31]. Another approach is to use synthetic data to pre-train the networks and fine-tune on real data. For tasks like optical flow [24] and depth estimation [15, 19, 10] synthetic data is much more accessible and does not contain the various artifacts and sampling noises that real data has. However, this approach suffers from the domain shift problem where the data distribution of synthetic and real data can differ.

3. Method

Since our method is a simple modification of the standard BN method, we start recalling BN and then describe our UBN method.

3.1. Batch Normalization

Given an input batch of height $H$ and width $W$ with $N$ samples and $C$ channels $x \in \mathbb{R}^{N \times C \times H \times W}$, BN normalizes the mean and standard deviation for each individual feature channel $c$ during training:

$$
BN(x)_c = \gamma_c \left( \frac{x_c - \mu(x_c)}{\sigma(x_c)} \right) + \beta_c \tag{1}
$$

where $\gamma, \beta \in \mathbb{R}^C$ are affine parameters learned from data; $\mu, \sigma$ are the mean and standard deviation, computed across batch size and spatial dimensions independently for each feature channel.

The moving average of the mean $\tilde{\mu}$ and standard deviation $\tilde{\sigma}$ are updated using a momentum rate $\lambda$ during training and used to normalize feature maps during testing.

$$
\tilde{\mu}_c = \lambda \tilde{\mu}_c + (1 - \lambda) \mu(x_c) \tag{2}
$$

$$
\tilde{\sigma}_c = \lambda \tilde{\sigma}_c + (1 - \lambda) \sigma(x_c) \tag{3}
$$

3.2. Unsupervised Batch Normalization

UBN is based on updating first the batch statistics than the weights. Given an batch $x \in L$ from labeled dataset $L$ and a batch $y \in U$ from unlabeled dataset $U$ the first step
builds a combined batch $n = \{x, y\}$ and makes a forward pass to update the normalization statistics. The second step makes forward-backward pass while using $x$ and the updated normalization statistics calculated at the earlier step.

UBN normalizes the mean and standard deviation for each individual feature channel with:

$$UBN(x_c) = \gamma \left( \frac{x_c - \mu(n_c)}{\sigma(n_c)} \right) + \beta$$  \hspace{1cm} (4)

where $\mu(n_c), \sigma(n_c)$ are the mean and standard deviation, computed across batch size and spatial dimensions independently for each feature channel from the combined batch $n$. The $\hat{\mu}_c, \hat{\sigma}_c$ values are calculated similarly to BN using a moving average.

### 3.3. Analyzing UBN with a Toy Example

We illustrate the principles behind UBN and how it differs from BN using a toy example. We train a simple 5-layer fully connected network using BN layers on the Spiral Dataset\(^1\), with 50 labeled examples and 400 unlabeled examples. Using this simple dataset allows us to do a deeper analysis which would not be straightforward using the depth estimation task.

We see from Fig. 2 that supervised training with BN learns the distribution of the labeled examples well and therefore learns accurate decision boundaries. However, those decision boundaries do not reflect the true data distribution. Instead, using UBN the network learns to align its decision boundary with the data manifold and learns to interpolate between the labeled examples by utilizing the batch statistics of unlabeled examples. In the coming sections we will examine the mechanism behind UBN through this example.

#### 3.3.1 Bias in Batch Statistics

Batch statistics are calculated from labeled examples in BN. When there is only a limited number of labeled examples, this may cause the normalization statistics to be biased and end up hurting the model performance. We recorded the feature distributions from different layers of regularly spaced points, labeled points and unlabeled points while the network normalization statistics using the unlabeled examples.

We have observed from the histogram of feature maps that the distribution of unlabeled and labeled examples are drastically different from each other. Calculating batch statistics with respect to only the labeled examples will cause a significant amount of bias in the normalization values. We postulate that the difficulty of obtaining correct batch statistics will be an important cause for the failure of Deep Learning methods in problems where the data is limited. Our method alleviates this problem by calculating the normalization values with respect to a larger variety of examples that better reflect the true data distribution.

#### 3.3.2 Regularization in UBN

Our understanding is that the regularizing effect of UBN and similarly of BN comes from the noise that is induced by changing batch norm statistics which makes the network invariant to this type of noise. This specific noise is also different from simpler random variations that is applied to gradients [21] and feature maps [7] of the network since it comes from the actual data distribution and extends the data boundaries in the manifold that is predicted by the network. Other method for injecting learned noises are also not applicable in the semi-supervised setting where the small number of labels doesn’t allow for such networks to be trained.

We visualized the uncertainty that is induced by changing batch statistics by making multiple predictions using the

---

\(^1\)http://playground.tensorflow.org
same inputs while updating the batch statistics [27]. This gives us a distribution of prediction from which we can calculate the standard deviation of our predictions. We observe that the uncertainty of the network follows the data manifold and aligns well with incorrect predictions and regions where there are no data examples 4.

We analyse the induced noise of changing batch statistics by making a forward pass of the network using labelled examples saving the network predictions updating the batch statistics and updating the inputs using back propagation in the feature space until the predictions are the same with the earlier batch statistics. This gives a distribution of starting points that have the same effect as changing the batch statistics. We observe that the augmentation follows the data manifold except for regions where the decision boundary is too close the the data manifold where in that case it pushes the decision boundary away from the labeled examples 5. Thus, Batch normalization layers apply an implicit augmentation that is dependant on the data manifold.

4. Experiments

KITTI dataset [8] contains stereo RGB images at 256x832 resolution and ground truth depth maps collected from Velodyne HDL-64E rotating 3D laser scanner at 10Hz. We have utilized the Kitti Raw dataset with 42304 training images and 2480 testing images.

The architecture for our method is based based on the SfMLearner architecture [33] which uses a modified version of the DispNet architecture [20]. It is a fully convolutional encoder-decoder architecture with skip connections and predictions in multiple resolutions. The building block of the architecture consists of 3x3 convolution operation, BN layers and a ReLU activation. The loss is calculated in different resolution by scaling the depth maps to match the prediction resolutions. The last layer before a prediction a sigmoid activation is used. The result of the sigmoid function is than scaled, shifted and inverted to obtain the depth prediction. The depth prediction $D_i$ for a feature map $X_i$ using a scaling factor of $s > 0$ and a minimum disparity value $m$ such that $0 < m < 1$ can be calculated as follows

$$\hat{D}_i = \frac{1}{s \cdot \text{sigm}(X_i) + m}$$

The network is trained with a learning rate of 0.0002 using the ADAM optimizer [14] for 160 epochs with batch size of 128. The scaling parameter $s$ is 0.4 and the minimum disparity value $m$ is 0.0125 (max depth 80m) throughout the experiments.

In order to evaluate the effect of unlabeled examples on the BN statistics we compare the performance of the depth prediction network on the KITTI dataset where a certain fraction of the depth maps are missing while we keep the number of images the same.

Removing depth annotation in this way let us keep the variation in the dataset stable while reducing the total number of annotated examples drastically. The missing depth maps are designed to mimic a depth sensor with lower frame rate.

For the metrics we used accuracy $\delta$ with threshold $\tau$:

$$\delta = \max(\frac{\hat{D}_i}{D_i}, \frac{D_i}{\hat{D}_i}) < \tau$$ for $\tau = 1.25$.

UBN shows progressively larger improvement as the number of labeled examples shrink 6 and outperforms the supervised BN training. If the number of labeled images is close to the unlabeled images the two methods become effectively the same which makes our method more suitable for problems with limited labels.

5. Conclusion

We have shown that UBN reduces the bias in batch statistics and applies a regularization that utilizes the data manifold. Our method is easy to implement, computationally inexpensive and is effective in problems where obtaining annotations is difficult.
References


