Advantages of Batch Normalization

Advantages of Batch Normalization

1. Speeds up the training process by normalizing the hidden layer activations

$$\mu = \frac{1}{m} \sum h_i$$

$$\sigma = \left[\frac{1}{m} \sum (h_i - \mu)^2 \right]^{1/2}$$

$$h_{i(norm)} = \frac{(h_i - \mu)}{\sigma + \varepsilon}$$

$$h_i = \gamma\, h_{i(norm)} + \beta$$

Advantages of Batch Normalization

1. Speeds up the training process by normalizing the hidden layer activations

2. Solves internal covariate shift

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

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

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

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.

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.

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.

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

Internal Covariate Shift

Dog	Non-Dog
Y = 1	Y = 0






Internal Covariate Shift

Dog

Y = 1

Non-Dog

Y = 0

Dog

Y = 1

Internal Covariate Shift

Dog

Y = 1

Non-Dog

Y = 0

Internal
Covariate Shift

Dog

Y = 1

How Does Batch Normalization Help Optimization?

How Does Batch Normalization Help Optimization?

Shibani Santurkar* MIT shibani@mit.edu
Dimitris Tsipras* MIT tsipras@mit.edu
Andrew Ilyas* MIT ailyas@mit.edu
Aleksander Mądry MIT madry@mit.edu

Abstract

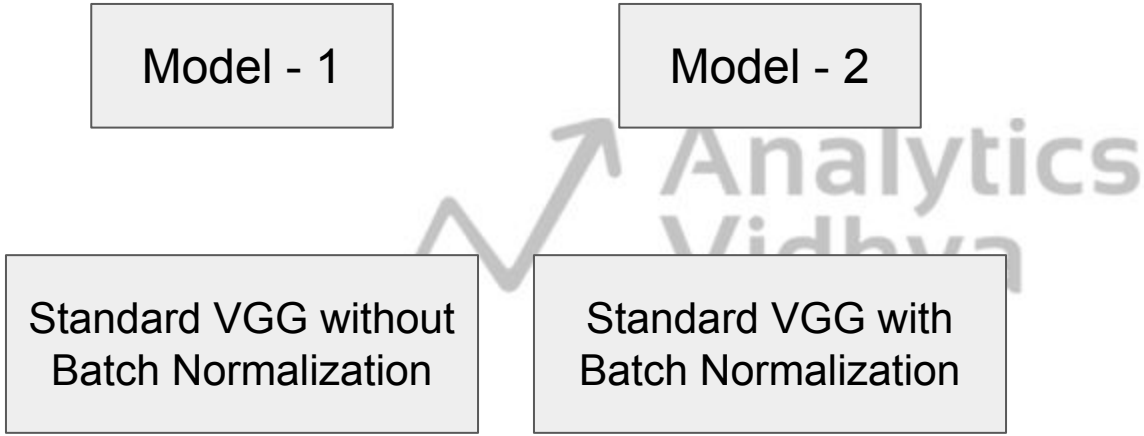
Batch Normalization (BatchNorm) is a widely adopted technique that enables faster and more stable training of deep neural networks (DNNs). Despite its pervasiveness, the exact reasons for BatchNorm's effectiveness are still poorly understood. The popular belief is that this effectiveness stems from controlling the change of the layers' input distributions during training to reduce the so-called "internal covariate shift". In this work, we demonstrate that such distributional stability of layer inputs has little to do with the success of BatchNorm. Instead, we uncover a more fundamental impact of BatchNorm on the training process: it makes the optimization landscape significantly smoother. This smoothness induces a more predictive and stable behavior of the gradients, allowing for faster training.

How Does Batch Normalization Help Optimization?

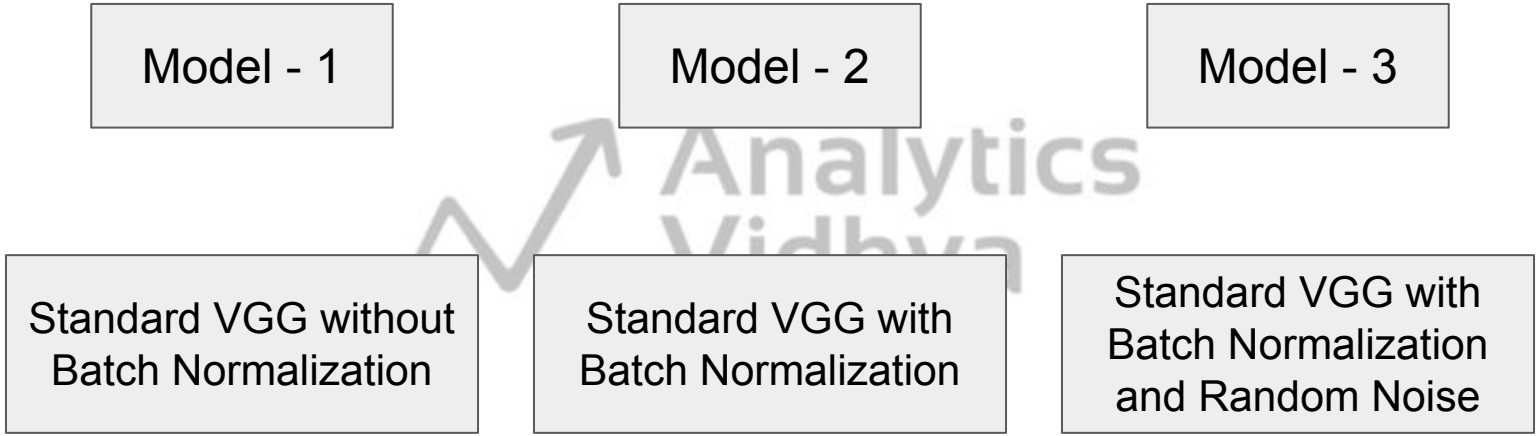
Model - 1

Standard VGG without Batch Normalization

How Does Batch Normalization Help Optimization?



How Does Batch Normalization Help Optimization?



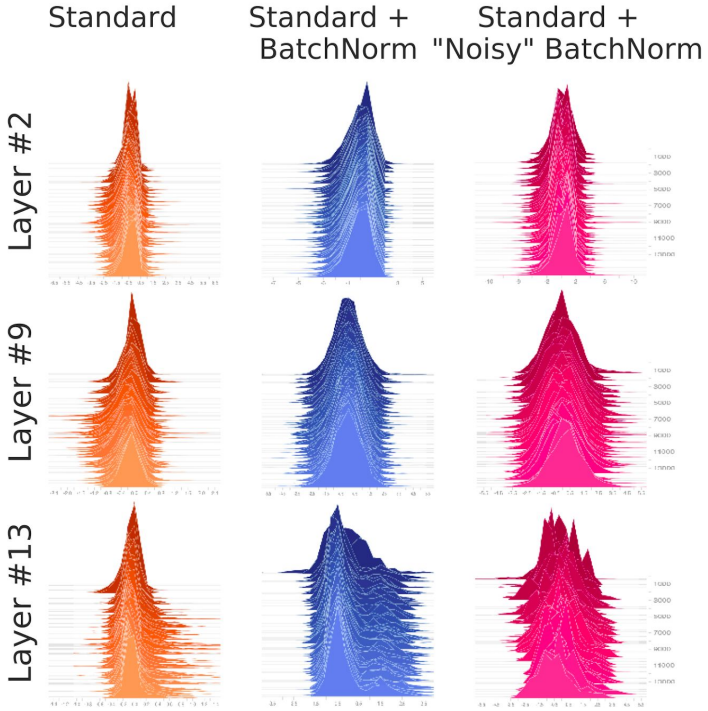
How Does Batch Normalization Help Optimization?

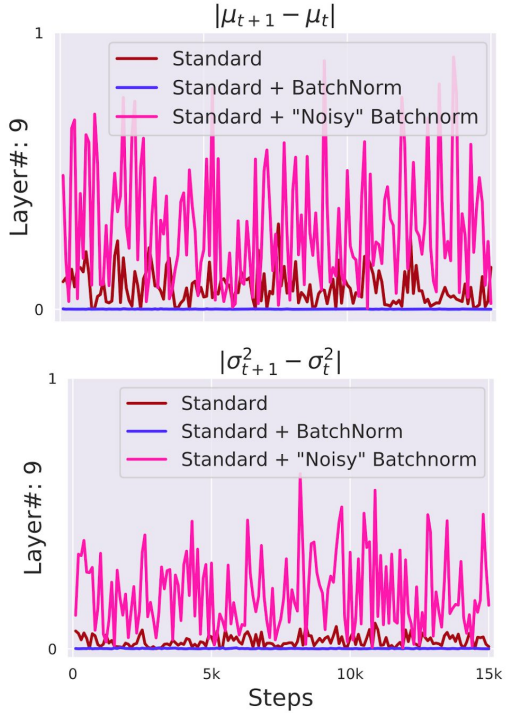
Conclusions:

How Does Batch Normalization Help Optimization?

Conclusions:

1. Noisy model has less stable distributions across all the layers



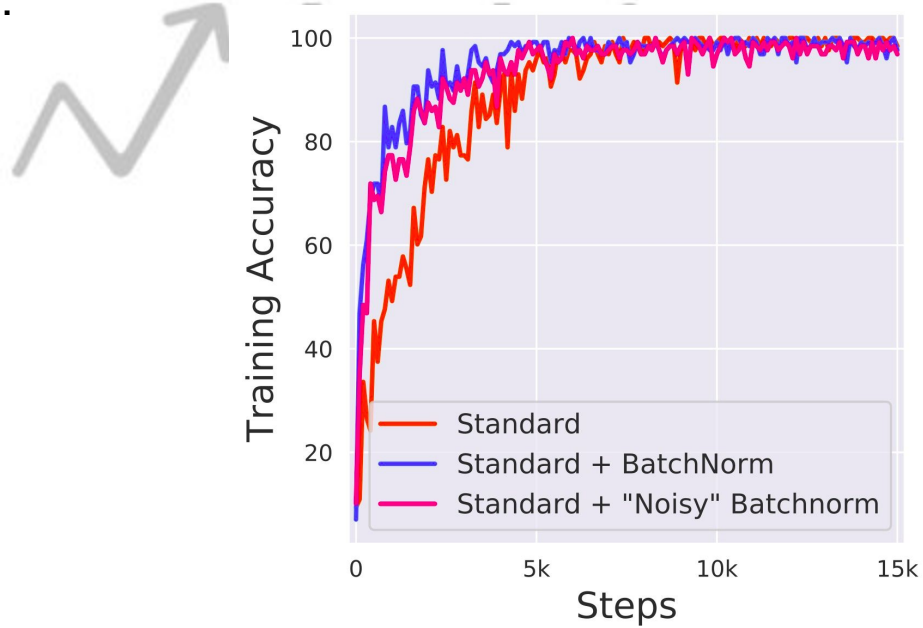


How Does Batch Normalization Help Optimization?

Conclusions:

1. Noisy model has less stable distributions across all the layers

2. Training accuracy of second and third model are similar, both higher than that of first model.



Advantages of Batch Normalization

1. Speeds up the training process by normalizing the hidden layer activations

2. Normalizes the distribution of hidden layer activations

3. Smoothens the loss function

Thank You!