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

Abstract

internal covariate shift
parameters of the previous layers change
distribution of each layer’s inputs changes during training
saturating nonlinearities

해결법은 normalizing layer inputs
for each training mini-batch

batch normalizatino의 장점
much higher learning rates
be less careful about initialization
regularizer, in some cases eliminating the need for Dropout

Introduction

SGD

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

minibatch x1...m of size m
approximate the gradient of the loss function with respect to the parameters,

$$\frac{1}{m} \frac{\partial \ell(\mathrm{x}_i, \Theta)}{\partial \Theta}.$$

Using mini-batches of examples 장점
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
parallelism

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
-> covariate shift

it is advantageous for the distribution of x to remain fixed over time
Then, Θ 2 does 1 not have to readjust to compensate for the change in the distribution of x

Fixed distribution of inputs to a sub-network would have positive consequences for the layers outside the subnetwork, as well

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
-> ReLU
but 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

Internal Covariate Shift를 제거하면 faster training을 보장할 수 있음
-> Batch Normalization fixes the means and variances of layer inputs
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
더 높은 learning rate 사용 가능
batch normalization regularizes the model
reduces the need for Dropout
saturating nonlinearities 사용 가능

Towards Reducing Internal Covariate Shift
Internal Covariate Shift
the change in the distribution of network activations due to the change in network parameters during training
we seek to reduce the internal covariate shift

network training converges faster if its inputs are whitened
linearly transformed to have zero means and unit variances, and decorrelated
it is advantageous to achieve the same whitening of the inputs of each layer
whitening the inputs to each layer -> achieving the fixed distributions of inputs -> remove internal covariate shift.

when whitening?
if these modifications are interspersed with the optimization steps the effect of the gradient step reduces
근데 그러면 b가 무한대로 커질 수 있음 왜? 정규화 과정에서 b를 빼주니까
the model blows up when the normalization parameters are computed outside the gradient descent step
왜? gradient descent optimization does not take into account the fact that the normalization takes place.
따라서 always produces activations with the desired distribution
그러면 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

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

input 전체를 고려하면 whitening 비용이 너무 비쌈
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

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

Normalization via Mini-Batch Statistics

두 가지 simplication을 했다

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 the variance of 1

speeds up convergence, even when the features are not decorrelated

Note that simply normalizing each input of a layer may change what the layer can represent

따라서 the transformation inserted in the network can represent the identity transform

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

두번째

each training step is based on the entire training set

this is impractical when using stochastic optimization

we use mini-batches in stochastic gradient training, each mini-batch produces estimates of the mean and variance of each activation

the statistics used for normalization can fully participate in the gradient backpropagation

use of minibatches is enabled by computation of per-dimension variances rather than joint covariances

in the joint case, regularization would be required

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

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

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

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

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

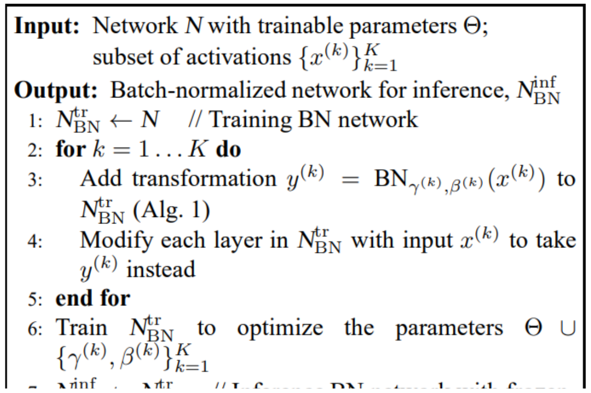
Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

BN depends on
the training example and the other examples in the mini-batch

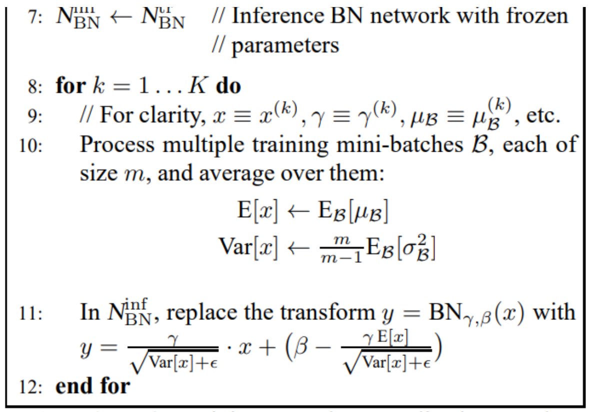
BN transform is a differentiable transformation that introduces normalized activations into the network

less internal covariate shift
accelerating the training
BN to represent identity transform and presrves the network capacity

Training and Inference with Batch-Normalized Networks
insert BN transform to each of them
batch gradient descent, or Stochastic Gradient Descent



```
Input: Network $N$ with trainable parameters $\Theta$;
       subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N_{\mathrm{BN}}^{\mathrm{inf}}$
1: $N_{\mathrm{BN}}^{\mathrm{tr}} \leftarrow N$   // Training BN network
2: for $k = 1 \ldots K$ do
3:     Add transformation $y^{(k)} = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to
       $N_{\mathrm{BN}}^{\mathrm{tr}}$ (Alg. 1)
4:     Modify each layer in $N_{\mathrm{BN}}^{\mathrm{tr}}$ with input $x^{(k)}$ to take
       $y^{(k)}$ instead
5: end for
6: Train $N_{\mathrm{BN}}^{\mathrm{tr}}$ to optimize the parameters $\Theta \cup$
   $\{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\mathrm{BN}}^{\mathrm{inf}} \leftarrow N_{\mathrm{BN}}^{\mathrm{tr}}$   // Inference BN network with frozen
```



```
7:  $N^{\text{inf}}_{\text{BN}} \leftarrow N^{\text{tr}}_{\text{BN}}$   // 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:
            $\mathrm{E}[x] \leftarrow \mathrm{E}_\mathcal{B}[\mu_\mathcal{B}]$
            $\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_\mathcal{B}[\sigma_\mathcal{B}^2]$
11:     In $N^{\text{inf}}_{\text{BN}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with
        $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$
12: end for
```

Algorithm 2: Training a Batch-Normalized Network

Batch-Normalized Convolutional Network

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

for convolution layers
we additionally want the normalization to obey the convolutional property
different elements of the same feature map, at different locations, are normalized in the same way
in a minibatch, over all locations

Batch Normalization enables higher learning rates
explode or vanish
stuck in poor local minima
normalization - it prevents small changes to the parameters from amplifying into larger and suboptimal changes in activations in gradients

also makes training more resilient to the parameter scale
원래는 large learning rates may increase the scale of layer parameters
그런데 batch normalization을 쓰면 backpropagation through a layer is unaffected by the scale of its parameters

The scale does not affect the layer Jacobian nor, consequently, the gradient propagation.
Moreover, larger weights lead to smaller gradients, and Batch Normalization will stabilize the parameter growth.

layer Jacobians to have singular values close to 1, which is known to be beneficial for training
even if real operation is not linear

Batch Normalization regularizes the model

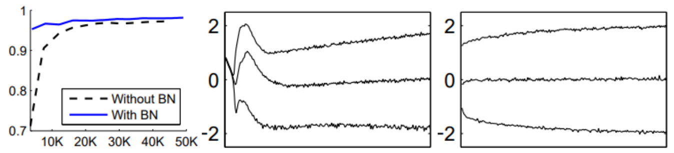
a training example is seen in conjunction with other examples in the mini-batch

the training network no longer producing deterministic values for a given training example

Experiments

Activations over time

MNIST



(a) (b) Without BN (c) With BN

The batch-normalized network enjoys the higher test accuracy

we studied inputs to the sigmoid

The distributions in the original network change significantly over time

the distributions in the batchnormalized network are much more stable as training progresses

ImageNet Classification

Accelerating BN Networks

Increase learning rate

Remove Dropout

Reduce the L2 weight regularization.

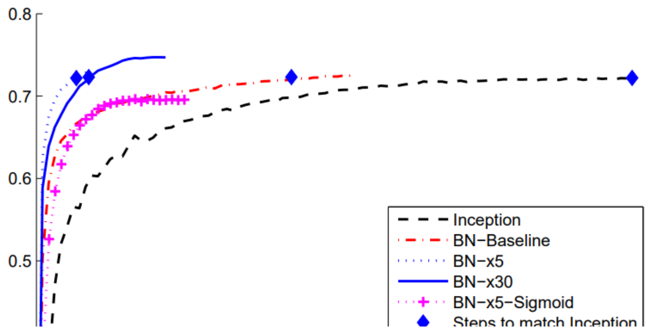
Accelerate the learning rate decay

Remove Local Response Normalization

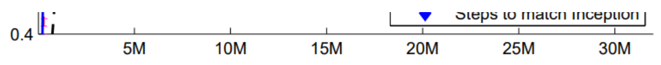
Shuffle training examples more thoroughly

Reduce the photometric distortions

Single-Network Classification

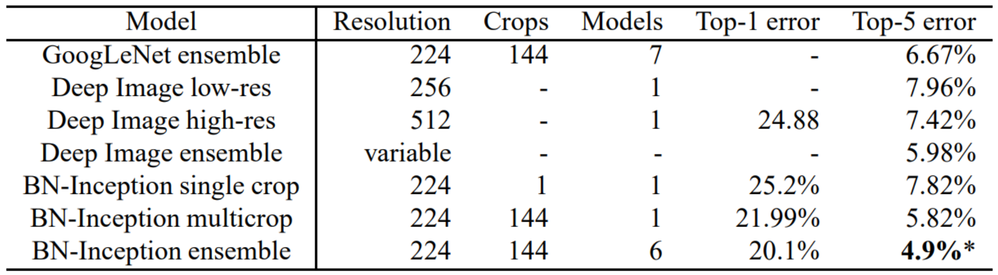


Model	Steps to 72.2%	Max accuracy
Inception	$31.0 \cdot 10^6$	72.2%
BN-Baseline	$13.3 \cdot 10^6$	72.7%
BN-x5	$2.1 \cdot 10^6$	73.0%
BN-x30	$2.7 \cdot 10^6$	74.8%
BN-x5-Sigmoid		69.8%



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

Ensemble Classification



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	variable	-	-	-	5.98%
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%	**4.9%***

Conclusion

novel mechanism for dramatically accelerating the training
premise that covariate shift applies to sub-networks and layers, and removing it from internal activations of the network may aid in training

normalizing activations
incorporating this normalization in the network architecture itself
-> 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
backpropagate the gradients through the normalization parameters

adds only two extra parameters per activation, and in doing so preserves the representation ability of the network

networks can be trained with saturating nonlinearities
more tolerant to increased training rates
do not require Dropout for regularization

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.

achieve a stable distribution of activation values throughout training
we apply it before the nonlinearity

차후 가능성
Recurrent Neural Networks
domain adaptation

theoretical analysis of the algorithm