Weight Normalization

A Simple Reparameterization to Accelerate Training of Deep Neural Networks

Tim Salimans & Diederik P.Kingma

Tim Salimans

FOLLOW

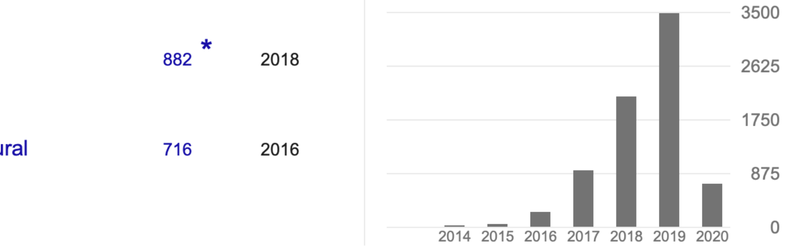
Google Brain Amsterdam
Verified email at google.com

generative models reinforcement learning deep learning approximate Bayesian infer…

GET MY OWN PROFILE

Cited by

	All	Since 2015
Citations	7674	7627
h-index	14	13
i10-index	20	17



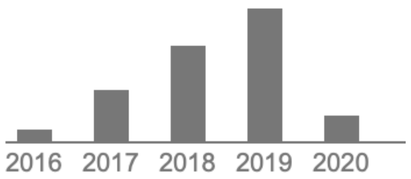
TITLE	CITED BY	YEAR
Improved techniques for training gans T Salimans, I Goodfellow, W Zaremba, V Cheung, A Radford, X Chen Advances in neural information processing systems, 2234-2242	3170	2016
Improving language understanding by generative pre-training * A Radford, K Narasimhan, T Salimans, I Sutskever URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers …	882	2018
Weight normalization: A simple reparameterization to accelerate training of deep neural networks T Salimans, DP Kingma Advances in neural information processing systems, 901-909	716	2016
Improved variational inference with inverse autoregressive flow DP Kingma, T Salimans, R Jozefowicz, X Chen, I Sutskever, M Welling Advances in neural information processing systems, 4743-4751	652	2016

Publication date 2016

Conference Advances in Neural Information Processing Systems

Pages 901-901

Total citations Cited by 713



Diederik P. Kingma

FOLLOW

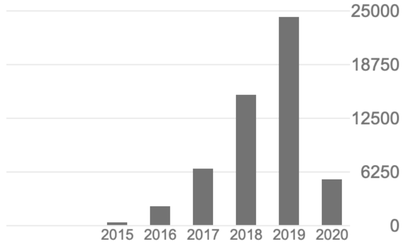
Research Scientist, Google Brain
Verified email at google.com - Homepage

Machine Learning Bayesian Inference Deep Learning Neural Networks

GET MY OWN PROFILE

Cited by

	All	Since 2015
Citations	54805	54293
h-index	18	18
i10-index	20	20



TITLE	CITED BY	YEAR
Adam: A Method for Stochastic Optimization DP Kingma, J Ba Proceedings of the 3rd International Conference on Learning Representations …	41488	2014
Auto-Encoding Variational Bayes DP Kingma, M Welling Proceedings of the 2nd International Conference on Learning Representations …	8287	2013
Semi-Supervised Learning with Deep Generative Models DP Kingma, S Mohamed, DJ Rezende, M Welling Advances in Neural Information Processing Systems, 3581-3589	1371	2014
Weight Normalization: A Simple Reparameterization to Accelerate Training of Deep Neural Networks T Salimans, DP Kingma Advances in Neural Information Processing Systems, 901-901	713	2016
Improved Variational Inference with Inverse Autoregressive Flow DP Kingma, T Salimans, R Jozefowicz, X Chen, I Sutskever, M Welling Advances in Neural Information Processing Systems, 4743-4751	652	2016

Co-authors VIEW ALL

Jimmy Ba
University of Toronto

Main contributions:

Present weight normalization, a reparameterization of the weight vectors to improve the conditioning of the optimization problem and speed up convergence of SGD.

Validate weight normalization in supervised image recognition, generative modeling and deep reinforcement learning.

Main Idea:

- Weight Normalization (WN) is a normalization technique, inspired by Batch Normalization (BN)
- It normalizes each layer's weights
- It can accelerate the convergence of stochastic gradient descent optimization

Weight Normalization:

$$\mathbf{w} = \frac{g}{||\mathbf{v}||}\mathbf{v}$$

$$\nabla_g L = \frac{\nabla_{\mathbf{w}} L \cdot \mathbf{v}}{||\mathbf{v}||}, \qquad \nabla_{\mathbf{v}} L = \frac{g}{||\mathbf{v}||}\nabla_{\mathbf{w}} L - \frac{g\nabla_g L}{||\mathbf{v}||^2}\mathbf{v},$$

$$\nabla_{\mathbf{v}} L = \frac{g}{||\mathbf{v}||} M_{\mathbf{w}} \nabla_{\mathbf{w}} L, \quad \text{with} \quad \mathrm{M}_{\mathbf{w}} = \mathrm{I} - \frac{\mathbf{w}\mathbf{w}'}{||\mathbf{w}||^2},$$

- Reparameterize weights w to g and v, where v represents orientation of w and g represents magnitude
- Calculate the gradient of loss function L with respect to g and v
- Rewrite the gradient of L with respect to v using M_w, which is a projection matrix that projects onto the complement of w

Weight normalization	**Batch normalization**
$\mathbf{w} = \frac{g}{\|\|\mathbf{v}\|\|}\mathbf{v}$	$t' = \frac{t - \mu[t]}{\sigma[t]}$
Work on each sample	Work on whole batch
- WN is much cheaper because CNN has fewer weights than pre-activations	
- WN can be applied more easily to models like RNNs, LSTMs and noise-sensitive applications like reinforcement learning	
- BN is robust against parameter initializations, while WN is not	

Data-dependent initialization of parameters

$$t = \frac{\mathbf{v} \cdot \mathbf{x}}{||\mathbf{v}||}, \quad \text{and} \quad y = \phi\left(\frac{t - \mu[t]}{\sigma[t]}\right),$$

$$g \leftarrow \frac{1}{\sigma[t]}, \qquad b \leftarrow \frac{-\mu[t]}{\sigma[t]},$$

$$y = \phi(\mathbf{w} \cdot \mathbf{x} + b)$$

- μ[t] and σ[t] are the mean and standard deviation of the pre-activation t over the examples in the minibatch
- Initialize b and g parameters to fix the minibatch statistics of all pre-activations
- Main idea: ensure that all features initially have zero mean and unit variance before application of the nonlinearity, like in BN

Mean-only batch normalization

$$t = \mathbf{w} \cdot \mathbf{x}, \qquad \tilde{t} = t - \mu[t] + b, \qquad y = \phi(\tilde{t})$$

$$\nabla_t L = \nabla_{\tilde{t}} L - \mu[\nabla_{\tilde{t}} L],$$

- w is the weight vector, parameterized using weight normalization
- μ[t] is the minibatch mean of the pre-activation t
- μ[.] denotes the operation of taking the minibatch mean
- Main idea: combine WN and mean-only BN because in WN the means of the neuron activations still depend on v

Experiments

Supervised classification: CIFAR-10

Settings:

Dataset : CIFAR-10

Architecture: ConvPool-CNN-C with small modifications

Optimizer: Adam

Batch size: 100

A fixed learning rate and momentum of 0.9 for the first 100 epochs

Linearly decay the learning rate to 0 and momentum of 0.5 for the last 100 epochs

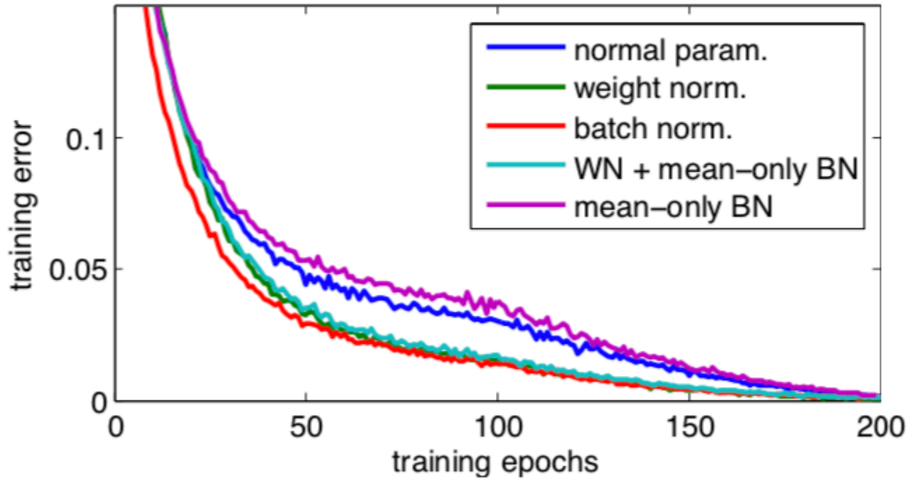


Figure 1: Training error for CIFAR-10 using different network parameterizations. For *weight normalization*, *batch normalization*, and *mean-only batch normalization* we show results using Adam with a learning rate of 0.003. For the normal parameterization we instead use 0.0003 which works best in this case. For the last 100 epochs the learning rate is linearly decayed to zero.

Model	Test Error
Maxout [6]	11.68%
Network in Network [17]	10.41%
Deeply Supervised [16]	9.6%
ConvPool-CNN-C [26]	9.31%
ALL-CNN-C [26]	9.08%
our CNN, mean-only B.N.	8.52%
our CNN, weight norm.	8.46%
our CNN, normal param.	8.43%
our CNN, batch norm.	8.05%
ours, W.N. + mean-only B.N.	**7.31%**

Figure 2: Classification results on CIFAR-10 without data augmentation.

Supervised classification: CIFAR-10

- WN & BN faster than the standard parameterization
- BN Batch makes slightly more progress per epoch than WN early on, however BN was about 16% slower than the standard parameterization while WN was not noticeably slower
- Later WN & BN optimize at about the same speed, with the normal parameterization lagging behind
- WN, the normal parameterization, and mean-only batch normalization have similar test accuracy (≈ 8.5% error). BN does significantly better at 8.05% error. Mean-only batch normalization combined with weight normalization has the best performance at 7.31% test error

Experiments

Generative modelling: Convolutional VAE

Settings:

Dataset: MNIST and CIFAR-10

Optimizer: Adamax

For MNIST, the encoder consists of 3 sequences of two ResNet blocks each, the first sequence acting on 16 feature maps, the others on 32 feature maps. The first two sequences are followed by a 2-times subsampling operation implemented using 2×2 stride, while the third sequence is followed by a fully connected layer with 450 units. The decoder has a similar architecture, but with reversed direction.

For CIFAR-10, we used a neural architecture with ResNet units and multiple intermediate stochastic layers. We used Adamax with $\alpha = 0.002$ for optimization, in combination with Polyak averaging in the form of an exponential moving average that averages parameters over approximately 10 epochs.

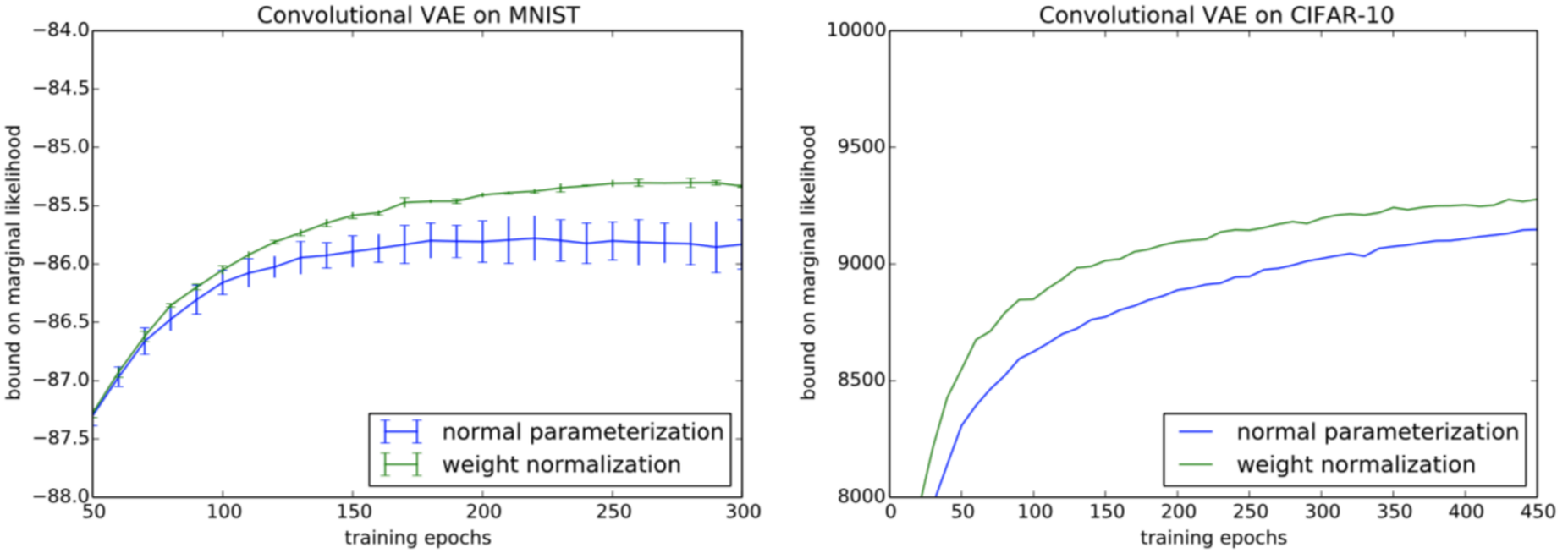


Figure 3: Marginal log likelihood lower bound on the MNIST (top) and CIFAR-10 (bottom) test sets for a convolutional VAE during training, for both the *standard* implementation as well as our modification with *weight normalization*. For MNIST, we provide standard error bars to indicate variance based on different initial random seeds.

Generative modelling: Convolutional VAE

- WN has lower variance and converges to a better optimum

Experiments

Generative modelling: DRAW

Settings:

DRAW is a recurrent generative model, which is similar to the previous model, but with both the encoder and decoder consisting of a recurrent neural network comprised of Long Short-Term Memory (LSTM) units.

Generative modelling: DRAW

- WN significantly speeds up convergence of the optimization procedure

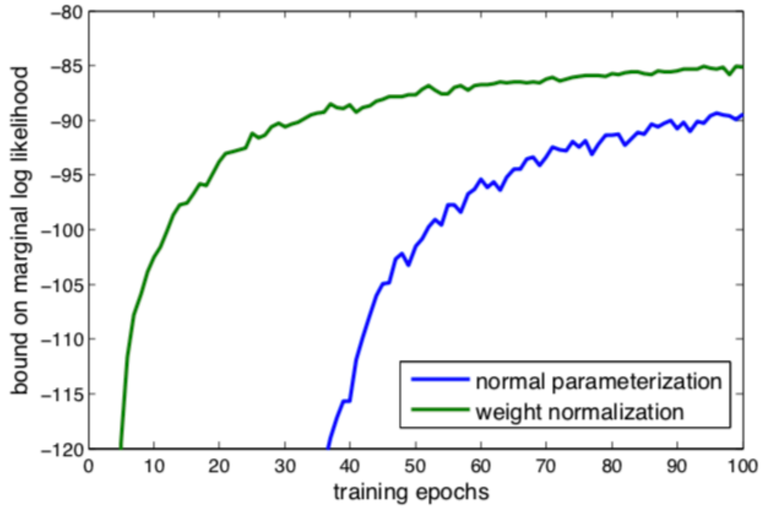


Figure 4: Marginal log likelihood lower bound on the MNIST test set for DRAW during training, for both the *standard* implementation as well as our modification with *weight normalization*. 100 epochs is not sufficient for convergence for this model, but the implementation using weight normalization clearly makes progress much more quickly than with the standard parameterization.

Experiments

Reinforcement learning: DQN

Settings:

Architecture: [1]

Optimizer: Stochastic gradient learning is performed using Adamax with momentum of 0.5.

Learning rate: 0.0003 for weight normalization and 0.0001 for the normal parameterization

Batch size: 64

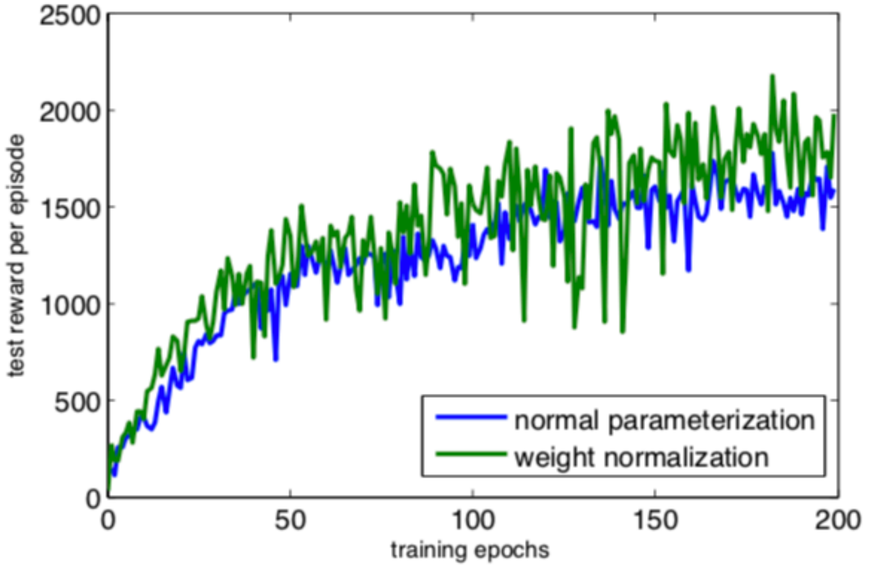


Figure 5: Evaluation scores for Space Invaders obtained by DQN after each epoch of training, for both the standard parameterization and using weight normalization. Learning rates for both cases were selected to maximize the highest achieved test score.

Game	normal	weightnorm	Mnih
Breakout	410	403	401
Enduro	1,250	1,448	302
Seaquest	7,188	7,375	5,286
Space Invaders	1,779	2,179	1,975

Figure 6: Maximum evaluation scores obtained by DQN, using either the normal parameterization or using weight normalization. The scores indicated by *Mnih et al.* are those reported by [21]: Our normal parameterization is approximately equivalent to their method. Differences in scores may be caused by small differences in our implementation. Specifically, the difference in our score on Enduro and that reported by [21] might be due to us not using a play-time limit during evaluation.

Reinforcement learning: DQN

- WN progresses more quickly and reaches a better final result
- WN on average improves the performance of DQN

Discussions

Discussions:

- “By decoupling the norm of the weight vector (g) from the direction of the weight vector (v/||v||), we speed up convergence of our stochastic gradient descent optimization.”— —Why?
- How does weight normalization work?
- How does batch normalization work?

Discussions:

- Do you think these experiments are enough to prove the author's idea?

Discussions:

- What trade-offs do you think that weight normalization has?
- If there are situations where weight normalization causes a slow-down or more computation for some reason?

Discussions:

- Other comments and questions?

References:

- V.Mnih, K.Kavukcuoglu, D.Silver,A.A.Rusu, J.Veness, M.G.Bellemare, A.Graves, M.Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.