Mathematical Institute

Improving the DNN loss landscape: batch-normalization and convex-ification

Theories of Deep Learning: C6.5, Video 11

Prof. Jared Tanner
Mathematical Institute
University of Oxford

Oxford Mathematics

Backpropogation: weight initialisation (Glorot et al.' 10)

Observed vanishing gradient

Mathematical Institute

Vanishing/exploding gradients was considered by Xavier Glorot and Yoshua Bengio (2010), using the same model assumptions as Pennington, $h^{(\ell)}$ being approximately $\mathcal{N}(0, \sigma_\ell^2)$ and:

"Our objective heres is to understand why standard gradient descent from random initialization is doing so poorly with deep neural networks.... we study how activations and gradients vary across layers and during training, with the idea that training may be more difficult when the singular values of the Jacobian associated with each layer are far from 1."

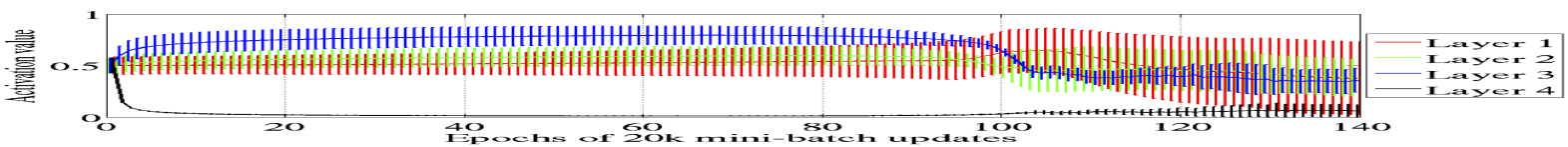


Figure 2: *Mean and standard deviation (vertical bars) of the activation values (output of the sigmoid) during supervised learning, for the different hidden layers of a deep architecture. The top hidden layer quickly saturates at 0 (slowing down all learning), but then slowly desaturates around epoch 100.*

`http://proceedings.mlr.press/v9/glorot10a.html`

Backpropogation: weight initialization (Glorot et al.' 10)

Variance normalization; precursor to Pennington with $\sigma_b = 0$

UNIVERSITY OF OXFORD
Mathematical Institute

Glorot initialization follows from seeking the variance of both the backpropogation gradient and forward activations to maintain the same variance per layer: for $W^{(\ell)} \in \mathbb{R}^{n \times n}$ need $\sigma_w^2 = 1/3n$.

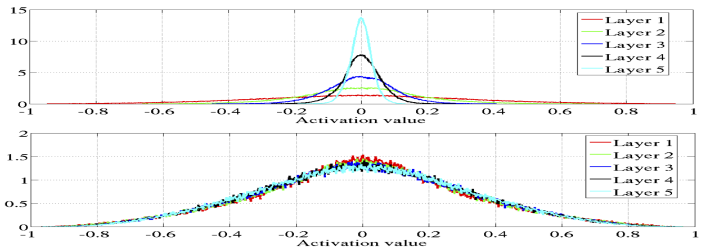


Figure 6: *Activation values normalized histograms with hyperbolic tangent activation, with standard (top) vs normalized initialization (bottom). Top: 0-peak increases for higher layers.*

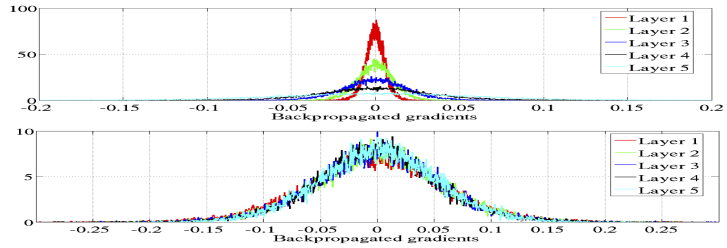


Figure 7: *Back-propagated gradients normalized histograms with hyperbolic tangent activation, with standard (top) vs normalized (bottom) initialization. Top: 0-peak decreases for higher layers.*

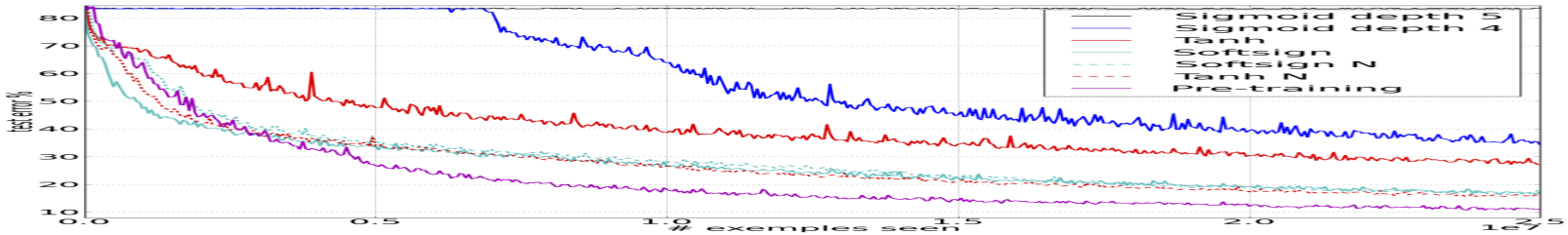


Figure 11: *Test error during online training on the* **Shapeset**-3 × 2 *dataset, for various activation functions and initialization schemes (ordered from top to bottom in decreasing final error). N after the activation function name indicates the use of normalized initialization.*

http://proceedings.mlr.press/v9/glorot10a.html

Batch normalization (Ioffe et al. 15')

Bulk normalization hyperparameters

University of Oxford
Mathematical Institute

Alternatively, bulk weight and bias normalizations, γ, and β, can be learned as part of the net parameters θ.

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

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

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

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

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

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

https://arxiv.org/pdf/1502.03167.pdf

Batch normalization experiment (Ioffe et al. 15')

Improved initial convergence rates

Mathematical Institute

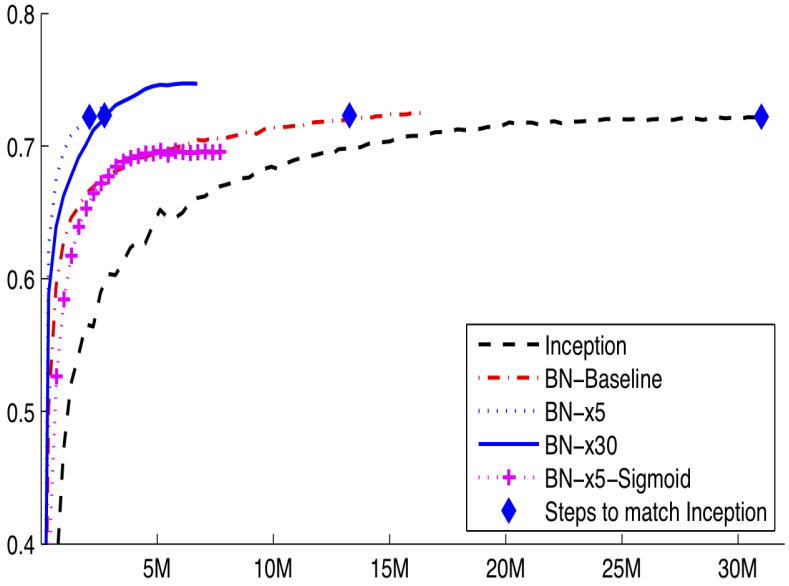


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

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%

Figure 3: *For Inception and the batch-normalized variants, the number of training steps required to reach the maximum accuracy of Inception (72.2%), and the maximum accuracy achieved by the network.*

`https://arxiv.org/pdf/1502.03167.pdf`

Convexifying CNN parameters pt. 1 (Zhang et al. 16')

The CNN structure has further non-convexity

Mathematical Institute

Consider a two layer convolutional neural network composed of one convolutional layer followed by a fully connected layer.
Rather than working with x directly, form P vectors $z_p(x)$ for $p = 1, \ldots, P$ where $z_p(x)$ is the portion of x on patch p of the convolutional layer. Then the k^{th} component of $H(x, \theta)$ is given by

$$H(x,\theta)_k = \sum_{j=1}^{r} \sum_{p=1}^{p} \alpha_{k,j,p} \sigma(w_j^T z_p(x)).$$

Alternatively if we exclude the nonlinearity we can express this by:

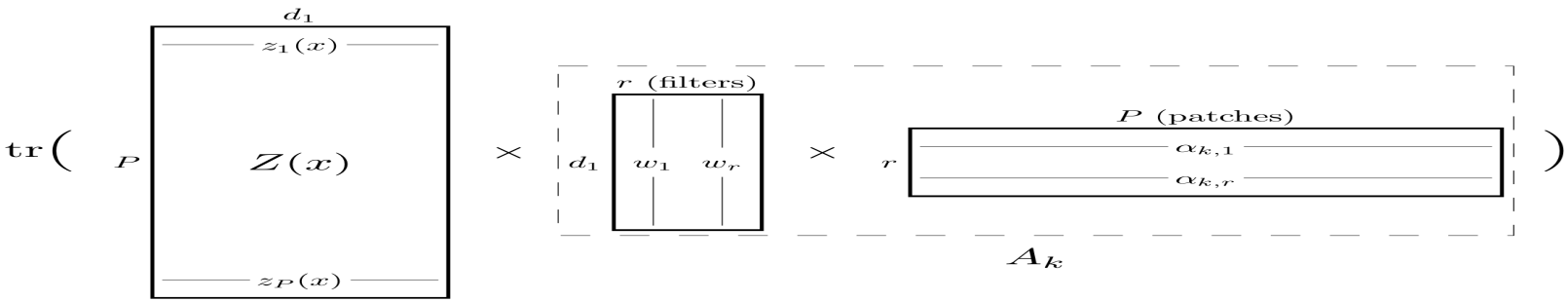
$$\sum_{j=1}^{r} \sum_{p=1}^{p} \alpha_{k,j,p} \sigma(w_j^T z_p(x)) = \sum_{j=1}^{r} Z(x) w_j$$

where $Z(x)$ has $z_p(x)$ as its p^{th} row.
`https://arxiv.org/pdf/1609.01000.pdf`

University of Oxford
Mathematical Institute

Using the trace formula this can be further condensed to

$$H(x,\theta)_k = \mathrm{tr}\left(Z(x)\left(\sum_{j=1}^{r} w_j \alpha_{k,j}^T\right)\right) = \mathrm{tr}\left(Z(x)A_k\right)$$



The network parameters are given by A_k nonlinearity is imposed by the A_k having rank r, and we can express all of the parameters of the matrix by A which is similarly rank r.
`https://arxiv.org/pdf/1609.01000.pdf`

Convexifying CNN parameters pt. 3 (Zhang et al. 16')

Convex relaxations are commonly used regularisers

UNIVERSITY OF OXFORD
Mathematical Institute

One can impose the network structure through A, but remove the non-convex rank constraint by replacing a convexification, that is the sum of the singular values of A (Schatten-1, or nuclear, norm).

If the convolutional filters and fully connected rows are uniformly bounded in ℓ^2 by B_1 and B_2 respectively, then one can replace then the sum of the singular values of A are bounded by $B_1 B_2 r\sqrt{n}$ where n is the network output dimension, the network parameters can be considered by varying the nuclear norm bound between 0 and $B_1 B_2 r\sqrt{n}$.

The resulting learning programme is fully convex and can be efficiently solved. The above can be extended to nonlinear activations and multiple layers, learning one layer at a time.
`https://arxiv.org/pdf/1609.01000.pdf`

Convexified CNN: MNIST (Zhang et al. 16')

Improved accuracy for shallow nets

UNIVERSITY OF OXFORD
Mathematical Institute

	basic	rand	rot	img	img+rot
SVM_{rbf} [44]	3.03%	14.58%	**11.11%**	22.61%	55.18%
NN-1 [44]	4.69%	20.04%	18.11%	27.41%	62.16%
CNN-1 (ReLU)	3.37%	9.83%	18.84%	14.23%	45.96%
CCNN-1	**2.38%**	**7.45%**	13.39%	**10.40%**	**42.28%**
TIRBM [38]	-	-	**4.20%**	-	35.50%
SDAE-3 [44]	2.84%	10.30%	9.53%	16.68%	43.76%
ScatNet-2 [8]	1.27%	12.30%	7.48%	18.40%	50.48%
PCANet-2 [9]	**1.06%**	6.19%	7.37%	10.95%	35.48%
CNN-2 (ReLU)	2.11%	5.64%	8.27%	10.17%	32.43%
CNN-2 (Quad)	1.75%	5.30%	8.83%	11.60%	36.90%
CCNN-2	1.38%	**4.32%**	6.98%	**7.46%**	**30.23%**

Table 1: Classification error on the basic MNIST and its four variations. The best performance within each block is bolded. The tag "ReLU" and "Quad" means ReLU activation and quadratic activation, respectively.

`https://arxiv.org/pdf/1609.01000.pdf`

Convexified CNN: CIFAR10 (Zhang et al. 16')

Monotonically decreasing training objective

University of Oxford
Mathematical Institute

	Error rate
CNN-1	34.14%
CCNN-1	**23.62%**
CNN-2	24.98%
CCNN-2	**20.52%**
$\text{SVM}_{\text{Fastfood}}$ [27]	36.90%
PCANet-2 [9]	22.86%
CKN [30]	21.70%
CNN-3	21.48%
CCNN-3	**19.56%**

Table 3: Classification error on the CIFAR-10 dataset. The best performance within each block is bolded.

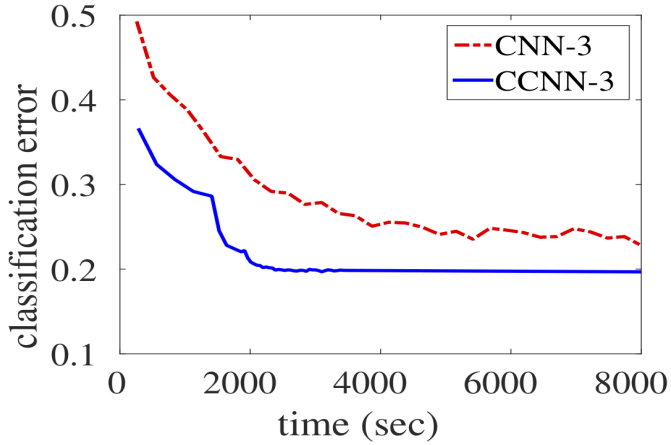


Figure 4: The convergence of CNN-3 and CCNN-3 on the CIFAR-10 dataset.

	CNN-1	CNN-2	CNN-3
Original	34.14%	24.98%	21.48%
Convexified	**23.62%**	**21.88%**	**18.18%**

Table 4: Comparing the original CNN and the one whose top convolution layer is convexified by CCNN. The classification errors are reported on CIFAR-10.

`https://arxiv.org/pdf/1609.01000.pdf`