Observations of the loss landscape: impact of parameterization and architecture

Theories of Deep Learning: C6.5, Lecture / Video 10

Prof. Jared Tanner
Mathematical Institute
University of Oxford

Oxford Mathematics

University of Oxford Mathematical Institute

DNN Loss function and trainable parameters

High dimensional loss function

Consider a fully connected L layer deep net given by

$$h^{(\ell)} = W^{(\ell)} z^{(\ell)} + b^{(\ell)}, \qquad z^{(\ell+1)} = \phi(h^{\ell}), \qquad \ell = 0, \dots, L-1,$$

for $\ell = 1, \dots, L$ with nonlinear activation $\phi(\cdot)$ and $W^{(\ell)} \in \mathbb{R}^{n_\ell \times n_\ell}$. The trainable parameters for the DNN, $\theta := \{W^{(\ell)}, b^{(\ell)}\}_{\ell=1}^{L}$ are learned by minimizing a high dimensional, $|\theta| \sim n^2 L$, loss function such as

$$\mathcal{L}(\theta; X, Y) = (2m)^{-1} \sum_{\mu=1}^{m} \sum_{i=1}^{n_L} (H(x_\mu(i); \theta) - y_{i,\mu})^2.$$

The shape of $\mathcal{L}(\theta)$ and our knowledge about a good initial minimizer $\theta^{(0)}$ strongly influence our ability to learn the parameters θ for the DNN.

Landscape loss function: VGG9 (Li et al. 18')

One dimensional views of a loss landscape

DNN loss $\mathcal{L}(\theta)$ between two minimizers, $\theta^s(1-\alpha)+\alpha\theta^l$ trained with small and large batches; horizontal axis α in (a) and (d).

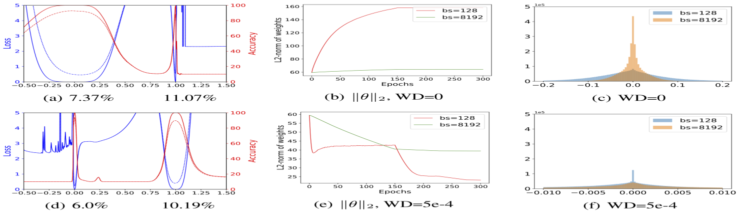


Figure 2: (a) and (d) are the 1D linear interpolation of VGG-9 solutions obtained by small-batch and large-batch training methods. The blue lines are loss values and the red lines are accuracies. The solid lines are training curves and the dashed lines are for testing. Small batch is at abscissa 0, and large batch is at abscissa 1. The corresponding test errors are shown below. (b) and (e) shows the change of weights norm $||\theta||_2$ during training. When weight decay is disabled, the weight norm grows steadily during training without constraints (c) and (f) are the weight histograms, which verify that small-batch methods produce more large weights with zero weight decay and more small weights with non-zero weight decay.

VGG9 is a CNN (Simonyan et al. 15')

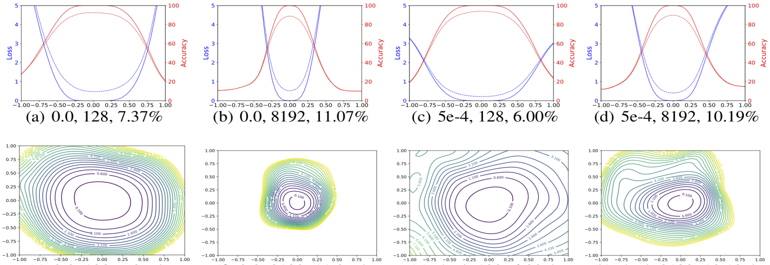
`https://arxiv.org/pdf/1409.1556.pdf`

`http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf`

Landscape loss function: VGG9 (Li et al. 18')

One and two dimensional landscape near SGD minima

Impact of training rate weight decay and batch size on level curves of $\mathcal{L}(\theta^* + \alpha\delta + \beta\eta)$. Larger batch size narrows the loss function. Weight decay broadens the loss function.



(a) 0.0, 128, 7.37% (b) 0.0, 8192, 11.07% (c) 5e-4, 128, 6.00% (d) 5e-4, 8192, 10.19%

(e) 0.0, 128, 7.37% (f) 0.0, 8192, 11.07% (g) 5e-4, 128, 6.00% (h) 5e-4, 8192, 10.19%

Figure 3: The 1D and 2D visualization of solutions obtained using SGD with different weight decay and batch size. The title of each subfigure contains the weight decay, batch size, and test error.

`http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf`

Loss landscape example: ResNet skip (Li et al. 18')

Architecture influences landscape: depth and skip connections

No-short is a standard fully connected DNN, ResNet (He et al. 15') has additional connections between every second layer.

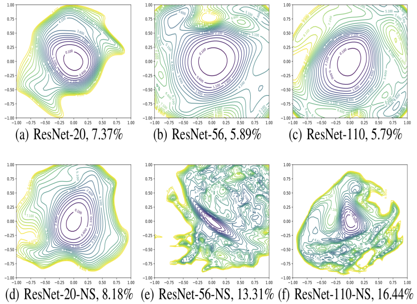


Figure 5: 2D visualization of the loss surface of ResNet and ResNet-noshort with different depth.

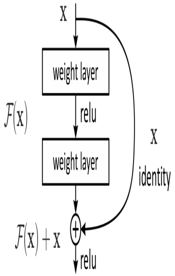


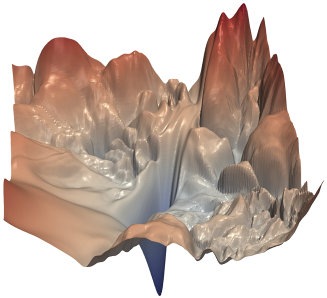
Figure 2. Residual learning: a building block.

https://arxiv.org/pdf/1512.03385.pdf

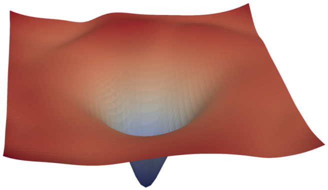
http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf

Loss landscape example: ResNet-56 (Li et al. 18')

Loss landscapes are generally highly non-convex



(a) without skip connections



(b) with skip connections

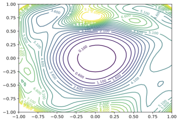
Figure 1: The loss surfaces of ResNet-56 with/without skip connections. The proposed filter normalization scheme is used to enable comparisons of sharpness/flatness between the two figures.

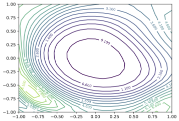
`http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf`

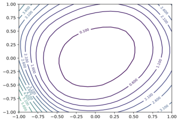
Loss landscape example: ResNet skip (Li et al. 18')

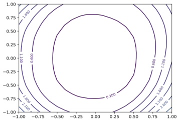
Architecture influences landscape: width

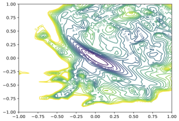
Increasing width over parameterises the net and broadens minima.

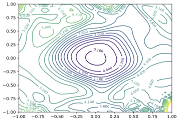

(a) $k = 1$, 5.89%

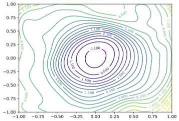

(b) $k = 2$, 5.07%


(c) $k = 4$, 4.34%


(d) $k = 8$, 3.93%


(e) $k = 1$, 13.31%


(f) $k = 2$, 10.26%


(g) $k = 4$, 9.69%

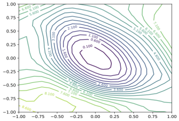

(h) $k = 8$, 8.70%

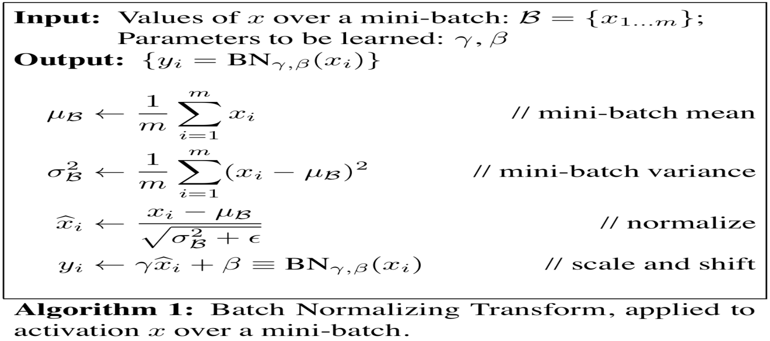
Figure 6: Wide-ResNet-56 on CIFAR-10 both with shortcut connections (top) and without (bottom). The label $k = 2$ means twice as many filters per layer. Test error is reported below each figure.

`http://papers.nips.cc/paper/7875-visualizing-the-loss-landscape-of-neural-nets.pdf`

Batch normalization (Ioffe et al. 15')

Bulk normalization hyperparameters

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: $\gamma, \beta$
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

    $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$                          // mini-batch mean
    $\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2$     // mini-batch variance
    $\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}}$                    // normalize
    $y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i)$           // 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

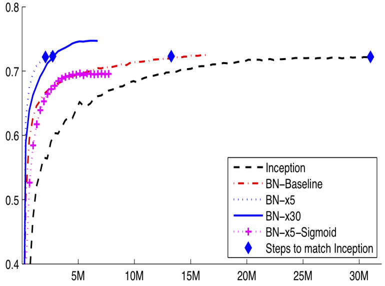


Figure 2: *Single crop validation accuracy of Inception and its batch-normalized variants, vs. the number of training steps.*

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

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.

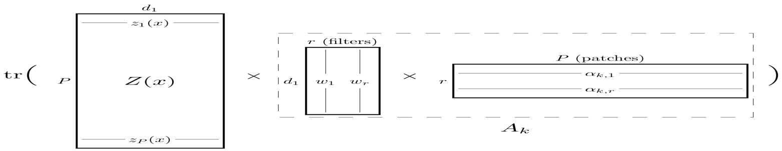
`https://arxiv.org/pdf/1609.01000.pdf`

Convexifying CNN parameters pt. 2 (Zhang et al. 16')

Low-rank non-convexity of CNNs

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

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

	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

	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.

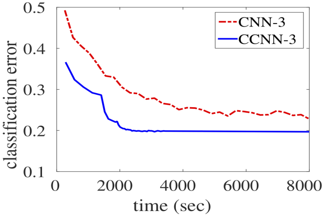


Figure 4: The convergence of CNN-3 and CCNN-3 on the CIFAR-10 dataset.

	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`

Improving the loss landscape summary

Influence of network architecture and training choices

- Larger training batch size narrows the loss function while weight decay (adding $\|\theta\|$ to the loss, broadens the loss function.
- Adding skip connections through residual networks can greatly smooth the loss landscape.
- Batch normalization can help train parameters in bulk and in so doing improve the training rate; though superfluous for expressivity.
- CNNs can even be convexified which may limit overall accuracy, but ensures ease of training regardless of initialization.