
Supplement for Low-Rank Learning by Design

Anonymous Author(s)

Affiliation

Address

email

1 Appendix A Table of Notation

Symbol	Subspace	Definition
<i>Data Set Notation</i>		
m	\mathbb{N}_+	Number of Input Features
\mathbf{x}	\mathbb{R}^m	Single input Sample
N	\mathbb{N}_+	Number of Samples
\mathbf{X}	$\mathbb{R}^{N \times m}$	Data set features with N samples and m features
n	\mathbb{N}_+	Number of Target Features
\mathbf{y}	\mathbb{R}^n	Single target sample
\mathbf{Y}	$\mathbb{R}^{N \times n}$	Data set targets with N samples and n features
T	\mathbb{N}_+	Length of Data Sequence
\mathcal{X}	$\mathbb{R}^{N \times m \times T}$	Data set feature tensor: N samples, m features, sequence length T
<i>Network Architecture</i>		
L	\mathbb{N}_+	Number of Layers in Neural Network
h_i	\mathbb{N}_+	Number of Neurons at Network Layer i
\mathbf{W}_i	$\mathbb{R}^{h_{i-1} \times h_i}$	Weight matrix from layers $i - 1$ to i
\mathbf{b}_i	\mathbb{R}^{h_i}	Bias at layer i
ϕ_i	$\phi_i : \mathbb{R}^{h_i} \rightarrow \mathbb{R}^{h_i}$	Activation function at layer i
$\Phi(\{\mathbf{W}_i\}_{i=0}^L, \{\mathbf{b}_i\}_{i=0}^L, \{\phi_i\}_{i=0}^L)$	$\Phi : \mathbb{R}^m \rightarrow \mathbb{R}^n$	L -layer Neural network
$h_0 = m$	\mathbb{N}_+	Number of Neurons at Input

Table 1: Important notation used throughout the main work.

2 Appendix B Parameter Tying: Bounds on Gradients in Convolutional Layers 3 with Linear Activations

4 Our derivation of bounds on the gradient rank of convolutional layers will follow much of what was
5 derived for RNNs. The primary difference will appear in the number of steps over which gradients
6 are accumulated, and their relationship to image size, stride, kernel size, padding, etc.

7 Suppose we are working on a convolution of dimension m . If we let N denote the size of a given
8 batch, and let C_{in}, C_{out} be the input/output channels, and let $\mathbf{w}_{in} \in \mathbb{N}_+^m$ be a list of image dimensions
9 (such as Height and Width for a 2d image). Then the input image to the convolution is denoted as a
10 tensor $\mathcal{X} \in \mathbb{R}^{N \times C_{in} \times w_{in,1} \times w_{in,2} \times \dots \times w_{in,m}}$.

11 Suppose we are performing a convolution with a kernel sizes $\mathbf{k} \in \mathbb{N}_+^m$, dilations $\mathbf{d} \in \mathbb{N}_+^m$,
12 padding $\mathbf{p} \in \mathbb{N}_+^m$, and stride $\mathbf{s} \in \mathbb{N}_+^m$. The weight tensor for this convolution is denoted as
13 $\mathcal{W} \in \mathbb{R}^{C_{out} \times C_{in} \times \mathbf{k}_1 \times \mathbf{k}_2 \times \dots \times \mathbf{k}_m}$. The output of this convolution is thus computed as the tensor
14 $\mathcal{Y} \in \mathbb{R}^{N \times C_{out} \times w_{out,1} \times w_{out,2} \times \dots \times w_{out,m}}$.

$$\mathcal{Y}_{i,j,\dots} = \sum_{k=1}^{C_{in}} \mathcal{W}_{i,k,\dots} \star \mathcal{X}_{i,k,\dots}$$

15 where \star is the m -dimensional cross-correlation operator.

16 Following reverse-mode auto-differentiation for m -dimensional convolutions, the gradient of \mathcal{W} is
17 computed as a convolution between the input \mathcal{X} and the adjoint from the backward pass Δ :

$$\nabla_{\mathcal{W}} = \sum_{i=1}^N \{\mathcal{X} \star \Delta\}_i$$

18 It is clear that with linear activations, even if \mathcal{X} and Δ are low-rank, the rank of this gradient
19 is accumulated over the cross-correlation operator \star (as well as over N , in a similar way to for
20 linear layers). It becomes apparent that like for RNNs, we are accumulating over the sequence of
21 $\prod_{i=1}^d w_{out,i}$ many patches, where $w_{out,i}$ is the size of the output in the i th dimension.

22 For convolutional layers, this can be explicitly computed as

$$w_{out,i} = \left\lfloor \frac{w_{in,i} + 2p_i - d_i \times (k_i - 1) - 1}{s_i} + 1 \right\rfloor$$

23 if we let \mathcal{B} be the linear bound computed in section 2.3, then the bound on the rank of the gradient is
24 thus

$$\text{rank}(\nabla_{\mathcal{W}_i}) \leq \mathcal{B} \prod_{i=1}^m \left\lfloor \frac{w_{in,i} + 2p_i - d_i \times (k_i - 1) - 1}{s_i} + 1 \right\rfloor \quad (1)$$

25 Intuitively, this means the bound will shrink as the input image size and padding shrinks, and will
26 shrink as the stride, dilation and kernel size increase.

27 Appendix C Rank of the Leaky-ReLU Derivative

28 We can use a similar kind of analysis as in section 3.3.1 to derive a bound on the rank for the partial
29 derivative of the Leaky-ReLU activation on the output. As we mentioned in that section, the analysis
30 can extend trivially to any piecewise linear function, and indeed the derivative of a Leaky-ReLU is
31 piecewise linear.

32 Let $\mathcal{D}_\alpha(\mathbf{Z}) \in \mathbb{R}^{h \times h}$ be the matrix with entries corresponding to the linear coefficients from the
33 Leaky-Relu activation applied to internal activation $\mathbf{Z}_i = \mathbf{X}\mathbf{W}_i$. The partial derivative w.r.t to the

output can be written as the hadamard product

$$\phi'_\alpha(\mathbf{Z}_i) = \mathbf{D}_\alpha \odot \mathbf{1}^{h \times h} \quad (2)$$

where $\mathbf{1}^{h \times h}$ is an $h \times h$ matrix of ones.

From our derivation in section 3.3.1, we can say that $\mathbf{D}_\alpha \odot \mathbf{1}^{h \times h}$ will remain numerically rank-deficient according to the bound

$$\min\{c_k(\mathbf{D}_\alpha), r_k(\mathbf{D}_\alpha)\} \leq \epsilon \min\{c_1(\mathbf{D}_\alpha), r_1(\mathbf{D}_\alpha)\} \sigma_1(\mathbf{Z}_i) / \sigma_k(\mathbf{Z}_i) \quad (3)$$

We note that $\mathbf{1}^{h \times h}$ is rank-1 with $\sigma_1 = h$ and $\sigma_k \leq \epsilon h$. If we suppose this bound is saturated, the bound depends primarily on the quantity $\min\{c_1(\mathbf{D}_\alpha), r_1(\mathbf{D}_\alpha)\}$, and will loosen somewhat as the precision of σ_k shrinks it toward 0.

Appendix D Connection to ReLU-Singular Values

One initial theoretical answer to analyse particular nonlinearities involves extending the notion of singular values to certain nonlinear functions which are locally linear, such as ReLU or Leaky-ReLU activations [1]. For the nonlinear operator $\phi_\alpha(\mathbf{Z}) = \text{LeakyReLU}_\alpha(\mathbf{Z})$, where $\alpha \in [0, 1]$ controls the slope of the activation when $\mathbf{Z} < 0$. We note that when $\alpha = 0$, this is equivalent to a ReLU activation and when $\alpha = 1$ it is equivalent to a Linear activation.

Following the work in [1], for a matrix $\mathbf{Z} \in \mathbb{R}^{m \times n}$ the k th ‘‘Leaky-ReLU Singular Value’’ is defined for the operator $\phi_\alpha(\mathbf{Z})$ as

$$s_k(\phi_\alpha(\mathbf{Z})) = \min_{\text{rank } \mathbf{L} \leq k} \max_{x \in \mathcal{B}} \|\text{LeakyReLU}_\alpha(\mathbf{Z}x) - \text{LeakyReLU}_\alpha(\mathbf{L}x)\|. \quad (4)$$

In [1], the extension of the notion of ReLU singular values to Leaky-ReLUs carries naturally; however, for completeness, we have included Leaky-ReLU-specific versions of each of the proofs from that work in the supplement. Among these results, we have the following lemma:

Lemma: Let $\phi_\alpha(\mathbf{Z}) = \text{LeakyReLU}_\alpha(\mathbf{Z})$ for $\mathbf{Z} \in \mathbb{R}^{n \times m}$, then

$$s_k(\phi_\alpha(\mathbf{Z})) \leq \sigma_k(\mathbf{Z}) \quad (5)$$

In other words, the Leaky-ReLU singular values will be bounded above by the singular values of the underlying linear transformation \mathbf{Z} . It follows then that as we increase α along the interval $[0, 1]$, the analogous notion of ‘‘rank’’ (the number of non-zero values of s_k) will converge upward to the linear rank of \mathbf{Z} , and our boundaries will still hold.

Appendix E Additional Empirical Verification

Hypothesis S1: *Low-rank input/output spaces systematically bound gradient rank.* The remaining terms in the bound in equation ? show that gradient rank is bound by the rank of the inputs and the rank partial derivative of the loss function (i.e., the rank of the output space). In Figure 1, we show the computed gradient rank on a fully-connected network with three layers of 128 neurons each, trained to reconstruct a full-rank (128-dim) gaussian input (panel 1a) in contrast to a model trained to reconstruct a low-rank (16-dim) gaussian embedded in a higher-dimensional (128-dim) space (panel 1b). Indeed, in panel 1c we see that any linear model which receives the low-rank embedded input has gradients which are bounded entirely by the rank of the input space.

Appendix F Large Scale Demonstration

In this section, we include results on larger-scale networks and problem settings. Our goal is to illustrate how architectural choices can affect rank not only in small models, but also in modern architectures. These results highlight the relevance of our theoretical result to the deep learning community at large.

ImageNet with ResNet18: In figure 2, we illustrate the effect of increasing the input image size on the rank of the gradient in the ResNet18 architecture. In each panel the image size increases from top to bottom, and as expected smaller image sizes produce smaller ranks.

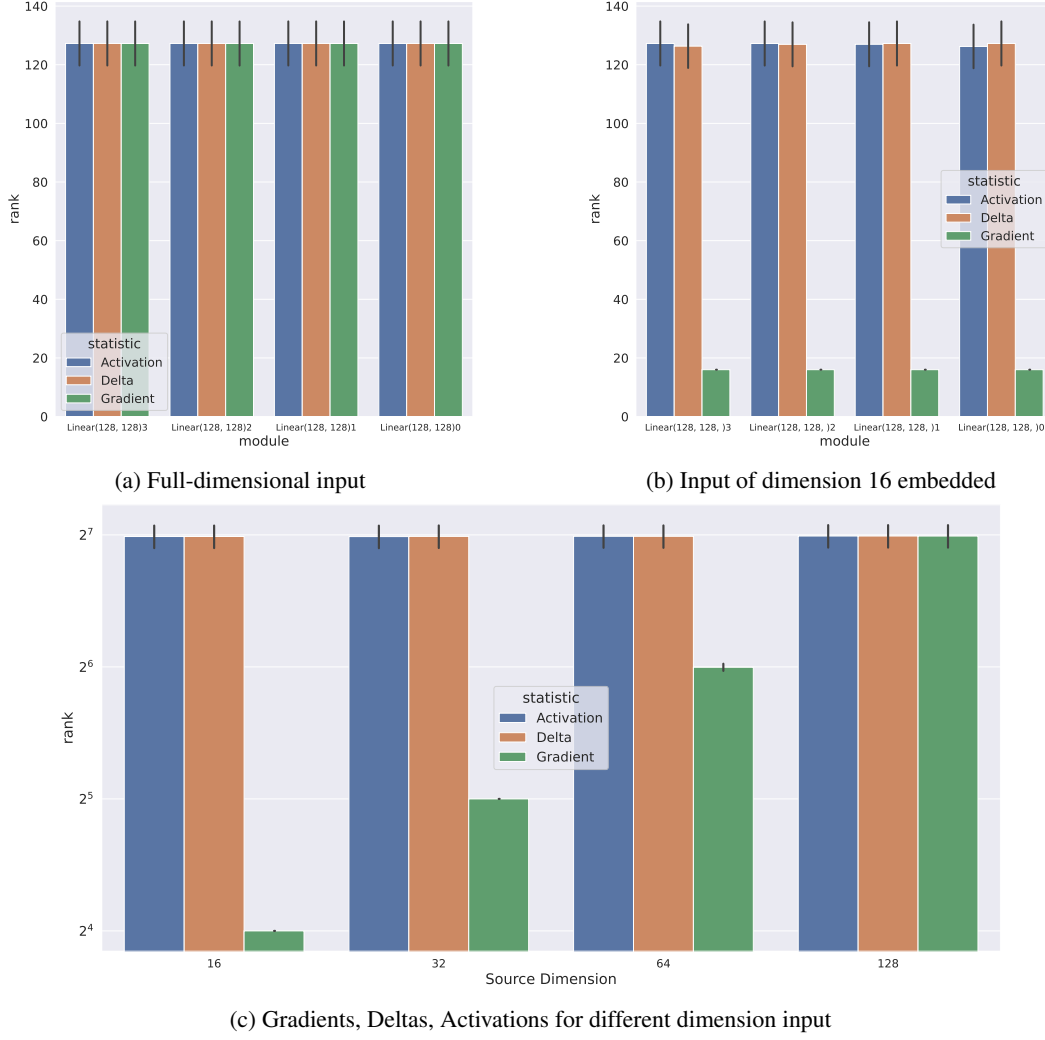
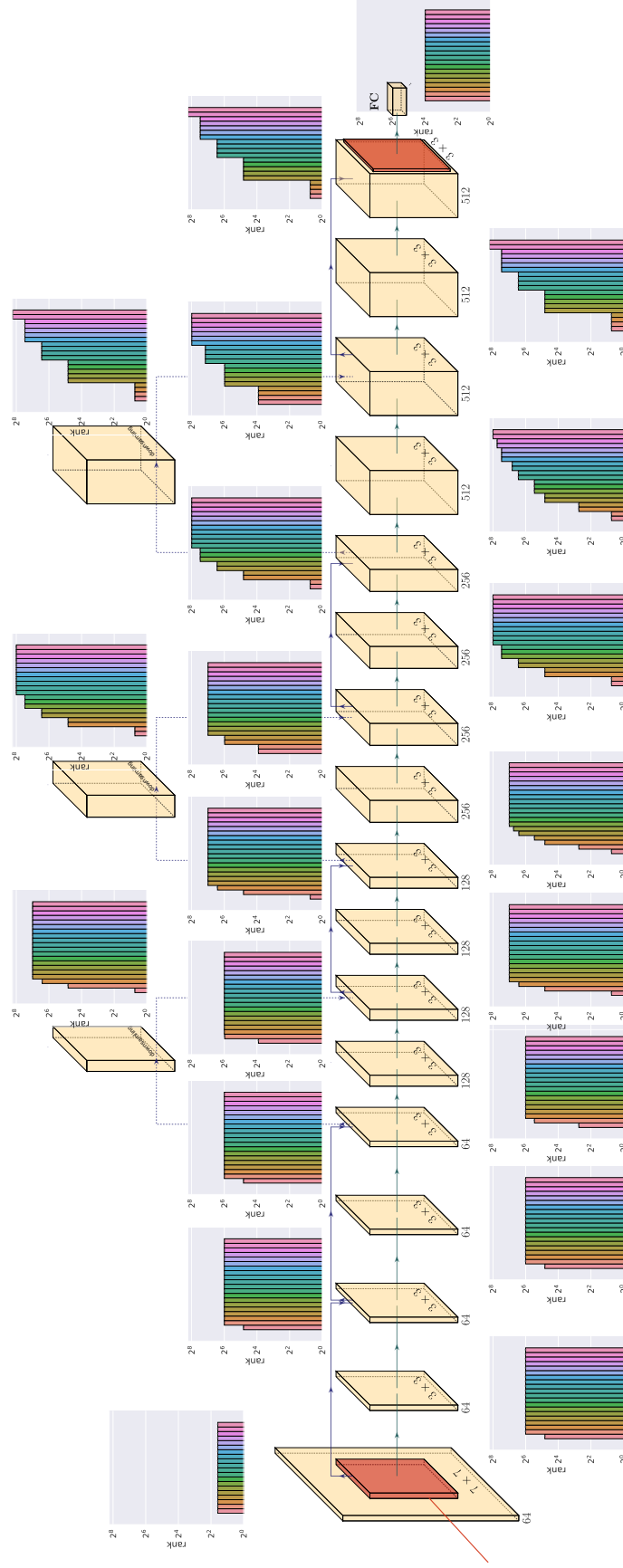


Figure 1: Low-Dimensional Input

74 Additionally, we estimated the effect of artificially introduced Leaky-ReLU activations with different
75 levels of α ; however, we observed little noticeable effect as long as the input image was large.



76 ***ImageNet with VGG11:*** In figure 3 we illustrate the effect of increasing the input image size on the
77 rank of the gradient in the ResNet18 architecture. In each panel the image size increases from top to
78 bottom, and as expected smaller image sizes produce smaller ranks.

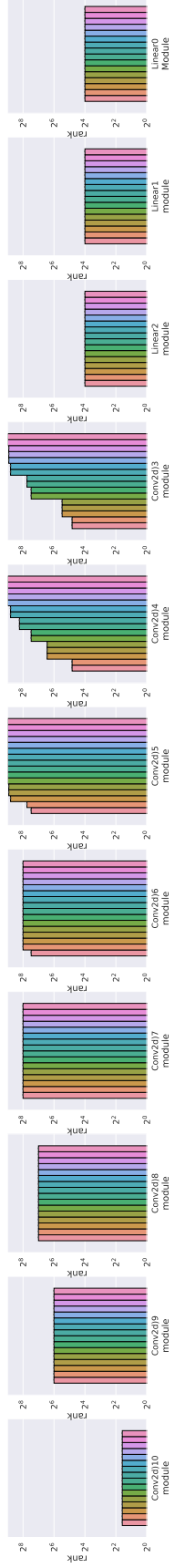


Figure 3: Illustration of the rank of the gradient at each layer in the VGG11 architecture used to classify Tiny-Imagenet. Each panel shows the effect of increasing image size (input is from top to bottom) on the rank, illustrating that larger image spaces provide more accumulation of the gradient.

79 **WikiText/M130k with BERT/XLM:** We have included a demonstration of decreasing the sequence
80 length in two large-scale NLP datasets (WikiText and Multi30K) for the BERT architecture with
81 standard pretraining and XLM pretraining. In figure 4 we include the estimated rank of the first two
82 and last two linear layers for BERT applied to the WikiText data set. The rank for all other layers and
83 for the4 XLM/Multi30k application are included in separate PDFs along with this supplement. In
84 general we do not see much variation between linear layers in each architecture, except for near the
85 input (layers 73,72), in which length 1 sequences collapse the rank of the gradients down to 1.

86 **References**

- 87 [1] DITTMER, S., KING, E. J., AND MAASS, P. Singular values for ReLU layers. *IEEE Transactions*
88 *on Neural Networks and Learning Systems* 31, 9 (2019), 3594–3605.

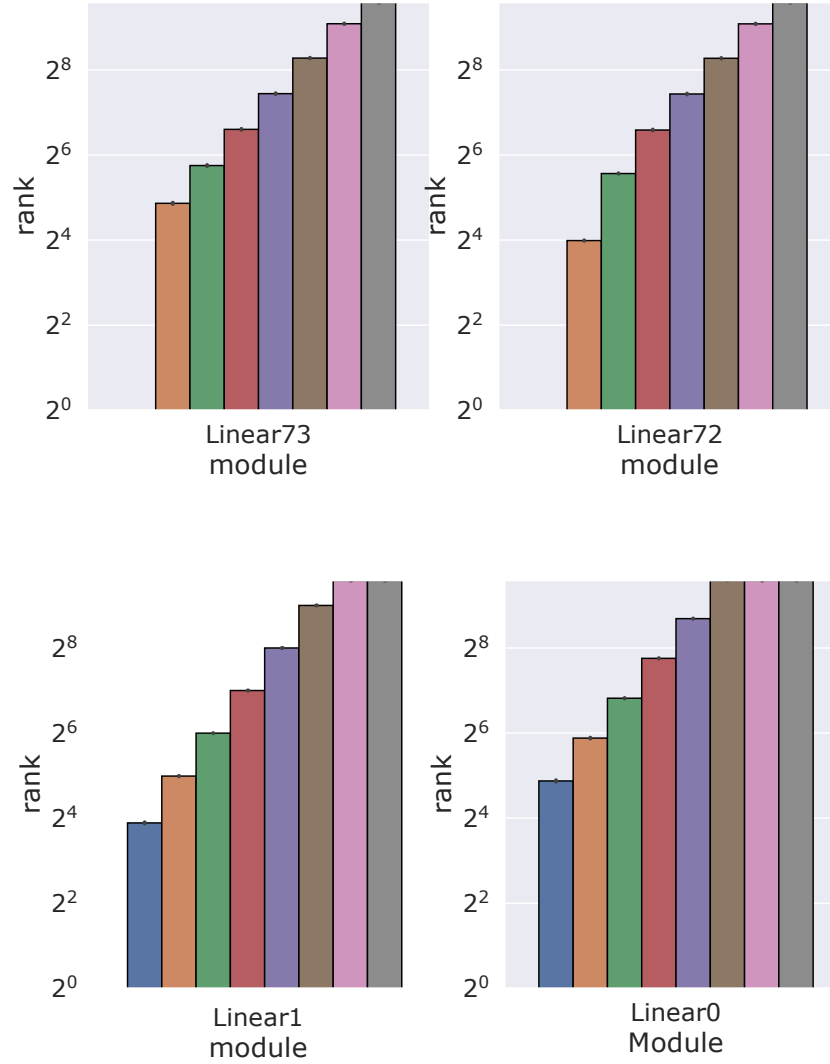


Figure 4: Illustration of the rank of the gradient at each layer in the BERT architecture used for language modeling on the WikiText2 data set. Each panel shows the effect of increasing sequence length (increasing from right to left) on the rank, illustrating that larger sequence lengths provide more accumulation of the gradient.