Tight and Efficient Upper Bound on Spectral Norm of Convolutional Layers (Appendix)

Ekaterina Grishina, Mikhail Gorbunov, and Maxim Rakhuba

HSE University
ergrishina@edu.hse.ru

A Strided convolution

A.1 Proof of Theorem 3

Proof. The proof uses techniques of the proof of Theorem 2 in [11]. In the case of convolution with zero padding, let $J_k \in \mathbb{R}^{\frac{n}{s} \times \frac{n}{s}}$ denote a matrix with ones along the k-th diagonal (if $k < 0$ the ones are below the main diagonal). For periodic convolution, $J_k = P^k$, where P is a permutation matrix:

$$P = \begin{bmatrix} 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \\ 1 & 0 & \dots & 0 \end{bmatrix}. \tag{1}$$

The Jacobian T_1 of a convolution with a stride $s = 1$ can be expressed as a summation of block doubly Toeplitz matrices:

$$T_1 = \sum_{k=-h_1}^{h_2} \sum_{p=-w_1}^{w_2} J_k \otimes J_p \otimes K_{:,:,k,p}.$$

We define a "strided Toeplitz matrix" as a Toeplitz matrix where each row is shifted by $s > 1$, resulting in a shape of $\frac{n}{s} \times n$. For example, when $n = 6$ and $s = 2$, the strided Toeplitz matrix takes the form:

$$A = \begin{bmatrix} a & b & c & d & 0 & 0 \\ e & f & a & b & c & d \\ 0 & 0 & e & f & a & b \end{bmatrix}.$$

Slicing this matrix by selecting columns at intervals determined by the stride s, such as columns $(0, s, 2 \cdot s, \dots (\frac{n}{s} - 1) \cdot s)$, results in a standard Toeplitz matrix. We can write this property as

$$A = \sum_{j=0}^{s-1} A_j (I_{\frac{n}{s}} \otimes e_j^T).$$

Let $B_i \in \mathbb{R}^{\frac{n^2}{s^2}c_{out} \times \frac{n^2}{s}c_{in}}$ be a block Toeplitz matrix. Each of its blocks $B_{ik} \in \mathbb{R}^{\frac{n}{s}c_{out} \times c_{in}n}$ is a block Toeplitz matrix with stride s. As in the previous example, we can denote slices of B_{ik} as $B_{ikj} \in \mathbb{R}^{c_{out}\frac{n}{s} \times c_{in}\frac{n}{s}}$, where each B_{ikj} is a block Toeplitz matrix with blocks of size $c_{out} \times c_{in}$. For brevity, we denote $k_1 = \lceil \frac{-h_1 - i}{s} \rceil, k_2 = \lfloor \frac{h_2 - i}{s} \rfloor, p_1 = \lceil \frac{-w_1 - i}{s} \rceil, p_2 = \lfloor \frac{w_2 - i}{s} \rfloor$.

$$B_i = \sum_{k=k_1}^{k_2} J_k \otimes B_{ik} = \sum_{k=k_1}^{k_2} J_k \otimes \sum_{j=0}^{s-1} B_{ikj}((I_{\frac{n}{s}} \otimes e_i^T) \otimes I_{c_{in}}) =$$
$$= \sum_{k=k_1}^{k_2} \sum_{j=0}^{s-1} J_k \otimes B_{ijk} \otimes e_j^T.$$

Strided linear transformation matrix may be expressed as follows

$$T_s = \sum_{i=0}^{s-1} B_i((I_{\frac{n}{s}} \otimes e_i^T) \otimes I_{c_{in}n}) = \sum_{i=0}^{s-1} B_i \otimes e_i^T =$$
$$= \sum_{i=0}^{s-1} \sum_{j=0}^{s-1} \sum_{k=k_1}^{k_2} J_k \otimes B_{ikj} \otimes e_j^T \otimes e_i^T =$$
$$= \sum_{i=0}^{s-1} \sum_{j=0}^{s-1} \sum_{k=k_1}^{k_2} \sum_{p=p_1}^{p_2} J_k \otimes J_p \otimes K_{:,:,i+ks,j+ps} \otimes e_j^T \otimes e_i^T$$
$$= \sum_{q=0}^{s^2-1} \sum_{k=\lceil \frac{-h_1 - q/s}{s} \rceil}^{\lfloor \frac{-h_2 - q/s}{s} \rfloor} \sum_{p=\lceil \frac{-w_1 - q\%s}{s} \rceil}^{\lfloor \frac{-w_2 - q\%s}{s} \rfloor} J_k \otimes J_p \otimes K_{:,:,\lfloor \frac{q}{s} \rfloor + ks, q \pmod s + ps} \otimes e_q^T =$$
$$= \sum_{k=\lceil \frac{-h_1 - s + 1}{s} \rceil}^{\lfloor \frac{h_2}{s} \rfloor} \sum_{p=\lceil \frac{-w_1 - s + 1}{s} \rceil}^{\lfloor \frac{w_2}{s} \rfloor} J_k \otimes J_p \otimes Q_{:,:,k,p}.$$

Thus, strided convolution may be viewed as a convolution with another kernel $Q \in \mathbb{R}^{c_{out} \times c_{in}s^2 \times \lceil \frac{h}{s} \rceil \times \lceil \frac{w}{s} \rceil}$, where

$$K_{c,d,a,b} = Q_{c,ds^2 + s(a \pmod s) + b \pmod s, \lfloor \frac{a}{s} \rfloor, \lfloor \frac{b}{s} \rfloor}.$$

From Theorem 1, it follows that $\|Q\|_\sigma \leq \|T_s\|_2 \leq \sqrt{\lceil \frac{h}{s} \rceil \lceil \frac{w}{s} \rceil} \|Q\|_\sigma$.

Here is an example of the code in pytorch for obtaining Q from K:

```
cout, cin, h, w = K.shape
if s != 1:
    if h % s != 0 and w % s != 0:
          p = (0, s - h % s, 0, s - w % s)
          K = F.pad(K, p, 'constant' , 0)
    Q = K.reshape(cout, cin, ceil(h/s), s, ceil(w/s), s)
    Q = Q.permute(0, 1, 3, 5, 2, 4)
```

```
    Q = Q.reshape(cout, cin*s*s, ceil(h/s), ceil(w/s))
else:
    Q = K
```

A.2 Experiments with a strided convolution

The bounds computed using Theorem 3 for a strided convolution with various kernel sizes are presented in the Table A.1. We observe that accuracy of the bounds depends on the size of the padding. Large number of zeros added to the kernel during padding reduces the accuracy.

Table A.1: Comparison of the TN bound and the $F4$ bound for a convolution with stride. The kernels are sampled from the Gaussian distribution. The exact value was computed with the power method [9] for the input size 32×32. We use 150 iterations for all the methods.

Sizes	Stride	$\|T\|_2$	$F4$	TN (Ours)	$\frac{F4}{\|T\|_2}$	$\frac{TN}{\|T\|_2}$
64, 64, 3, 3	2	38.14	53.85	46.72	1.412	**1.225**
512, 512, 3, 3	2	106.85	153.33	134.87	1.435	**1.262**
64, 64, 5, 5	2	60.9	106.91	72.69	1.756	**1.194**
512, 512, 5, 5	2	172.57	301.7	204.19	1.748	**1.183**
64, 64, 7, 7	2	86.51	175.41	100.3	2.028	**1.16**
512, 512, 7, 7	2	238.99	495.99	273.23	2.075	**1.143**
64, 64, 3, 3	4	31.93	31.93	31.93	1.0	**1.0**
512, 512, 3, 3	4	90.12	89.98	90.12	0.998	**1.0**
64, 64, 5, 5	4	50.94	80.58	79.09	1.582	**1.553**
512, 512, 5, 5	4	145.03	228.49	225.15	1.575	**1.552**
64, 64, 7, 7	4	70.67	86.77	79.13	1.228	**1.12**
512, 512, 7, 7	4	199.65	246.55	225.43	1.235	**1.129**

A.3 Spectral density function for strided convolutions

Theorem 3 shows that strided convolution with a kernel K can be expressed as a regular convolution with a kernel Q. Therefore, it admits a representation as a block doubly Toeplitz matrix with $T_{k,l}$:

$$T_{k,l} = Q_{:,:,k+h_1,l+w_1}$$

Additionally, by permuting output channel dimension of kernel Q (grouping modulo s^2), $T_{k,l}$ becomes a concatenation of s^2 matrices $T^1_{k,l}, \dots, T^{s^2}_{k,l}$, where T^i correspond to some convolutions with possibly different kernel sizes (up to

padding). Since the padding of kernels does not affect the spectral density function, spectral density of such a matrix is a concatenation of s^2 spectral density function with possibly different kernel sizes:

$$F(\omega_1, \omega_2) = \left(F^1(\omega_1, \omega_2) \dots F^{s^2}(\omega_1, \omega_2)\right)$$

Contrary to the more sophisticated spectral density function proposed in Section VI.A in [13], Lemma 2 also holds true since Q is a convolution kernel. This shows that bounding the spectral norm of T can be alternatively done on F, which can allow for further analysis of strided convolutions.

B Real and complex rank-1 approximation of tensors

In this section, we will present an example of a real tensor, for which real and complex best rank-1 approximations do not coincide with each other. Let us use notation $\|K\|_{\sigma,\mathbb{C}}$ in the case when supremum in the definition of the spectral norm is taken over complex vectors, and $\|K\|_{\sigma,\mathbb{R}}$ in the case of real vectors. The $\sqrt{hw}\|K\|_{\sigma,\mathbb{C}}$ bounds from above the spectral norm of a convolution for any input size, while this is not the case for $\sqrt{hw}\|K\|_{\sigma,\mathbb{R}}$ as we show in this section. Our example was inspired by [2, 4], see also examples in [7, 8].

Let a tensor $K \in \mathbb{R}^{2\times2\times2\times2}$ be $K = (e_1 + ie_2)^{\circ 4} + (e_1 - ie_2)^{\circ 4}$, where $e_1 = (1, 0)^T, e_2 = (0, 1)^T$, $\circ$ denotes tensor product. K is a real-valued tensor with an unfolding

$$K.reshape(2, 8, \texttt{order='c'}) = \begin{bmatrix} 2 & 0 & 0 & -2 & 0 & -2 & -2 & 0 \\ 0 & -2 & -2 & 0 & -2 & 0 & 0 & 2 \end{bmatrix}.$$

K is a supersymmetric tensor, which implies that it has a symmetric real best rank-1 approximation [6]. Let x be

$$x = \underset{\|x\|_2=1,\, x\in\mathbb{R}^2}{\arg\max} |[\![K, x, x, x, x]\!]|.$$

Thus, we can use the parametrization $x = (\cos\alpha, \sin\alpha)^T$.

$$\begin{aligned} |[\![K, x, x, x, x]\!]| &= |(x^T(e_1 + ie_2))^4 + (x^T(e_1 - ie_2))^4| = \\ &= |(\cos\alpha + i\sin\alpha)^4 + (\cos\alpha - i\sin\alpha)^4| = \\ &= |\cos(4\alpha) + i\sin(4\alpha) + \cos(4\alpha) - i\sin(4\alpha)| = \\ &= 2|\cos(4\alpha)| \le 2 \end{aligned}$$

Let $\alpha = \frac{\pi}{4}$ and $x = \left(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}\right)$, then the best real-valued rank-1 approximation of K is a tensor $2x^{\circ 4}$ and the corresponding singular value is $\|K\|_{\sigma,\mathbb{R}} = 2$. Hence, the bound is $\sqrt{hw}\|K\|_{\sigma,\mathbb{R}} = 4$.

The spectral norm of the above unfolding is $\|K.reshape(2, -1)\|_2 = 4$. From Lemma 1, we conclude that $\|K\|_{\sigma,\mathbb{C}} \le \|K.reshape(2, -1)\|_2 = 4$. The complex

rank-1 approximation of K is a tensor $4x^{\circ 4}$, where $x = \left(\frac{1}{2} + \frac{1}{2}i, -\frac{1}{2} + \frac{1}{2}i\right)$, the corresponding singular value is $\|K\|_{\sigma,\mathbb{C}} = 4$ (which coincides with singular value of the unfolding) and the bound is $\sqrt{hw}\|K\|_{\sigma,\mathbb{C}} = 8$.

We have computed the spectral norm of the convolution with circular padding exactly for different input sizes using [10]. For input size 4×4 (and sizes $4n \times 4n$) the spectral norm of the circular convolution is 8, which equals to the bound $\sqrt{hw}\|K\|_{\sigma,\mathbb{C}}$ and is, therefore, larger than $\sqrt{hw}\|K\|_{\sigma,\mathbb{R}} = 4$.

This example illustrates the fact that in order to compute the bound $\sqrt{hw}\|K\|_{\sigma,\mathbb{C}}$ correctly (so that it upper bounds spectral norm of the convolution), we need to look for the best rank-1 approximation using the complex version of HOPM.

C Gradient computation

Tensors $P_{real}, P_{im} \in \mathbb{R}^{2\times2\times2\times2}$ consist of $-1, 0, 1$. P_{real} is equal to the real part of the tensor $(e_1 + ie_2)^{\otimes 4}$, and P_{im} is equal to its imaginary part. Here $e_1 = (1, 0)^T, e_2 = (0, 1)^T$. The unfoldings of these tensors are as follows:

$$P_{real}.reshape(2, 8, \texttt{order='c'}) = \begin{bmatrix} 1 & 0 & 0 & -1 & 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

$$P_{im}.reshape(2, 8, \texttt{order='c'}) = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & -1 \\ 1 & 0 & 0 & -1 & 0 & -1 & -1 & 0 \end{bmatrix}$$

Tensors P_{real} and P_{im} are fixed, so we can precompute them beforehand and use later. For a faster computation of the gradient, we only need to compute the gradients $\nabla_K real, \nabla_K im$ according to the (9), because we already have the values $real$ and im from the forward pass. One can also use automatic differentiation instead of the derived formulas.

D Higher Dimensional Convolution

Proof of Theorem 2

Proof. An unfolding $R = K_{(1,23\ldots d+2)}$ of the kernel $K \in \mathbb{R}^{c_{out}\times c_{in}\times h_1\times\ldots\times h_d}$ is a submatrix of a multi-level Toeplitz matrix T, hence, $\|R\|_2 \leq \|T\|_2$. As is shown in Lemma 1, the spectral norm of an unfolding upper bounds tensor norm, so we can write the lower bound:

$$\|K\|_\sigma \leq \|R\|_2 \leq \|T\|_2$$

To prove the upper bound, we first need to state that inequality $\|T\|_2 \leq \|F\|_2$ holds true for the multidimensional convolution. For the readers' convenience, here we present the proof of this fact, which directly follows [13, Lemma 4] and [13, Section VI.B].

Following [13, Section VI.B], the Jacobian T can be written as

$$T = \sum_{[k_1]<n_1} \cdots \sum_{[k_d]<n_d} [J_{n_1}^{k_1} \otimes \cdots \otimes J_{n_d}^{k_d}] \otimes T_k,$$

$$T_k = \frac{1}{(2\pi)^d} \int_\Omega F(\tau) e^{-i\langle k,\tau\rangle} d\tau,$$

where J_n^k is a matrix of size $n \times n$ with ones along the k^{th} diagonal and zeros elsewhere in the case of zero padding or $J_n^k = P^k$ in the case of circular padding (P is a permutation matrix defined in (1)). $\Omega = [-\pi, \pi]^d$, $k = (k_1, \dots, k_d), \tau = (\tau_1, \dots, \tau_d)$. We can write the generating function as

$$F(\tau) = \sum_k T_k e^{i\langle k,\tau\rangle}.$$

Let $u \in \mathbb{R}^{c_{out} n^d}, v \in \mathbb{R}^{c_{in} n^d}$ be the singular vectors of T. Let us divide u and v into n^d subvectors of size c_{out} and c_{in}. Let $u_k \in \mathbb{R}^{c_{out}}, v_m \in \mathbb{R}^{c_{in}}$ be the k^{th} and m^{th} subvectors corresponding to $(T)_{k,m} = T_{k-m}$, $1 \le m, k \le n^d$, so we can write:

$$\|T\|_2 = u^T T v = \sum_k \sum_m u_k^T T_{k-m} v_m = \sum_k \sum_m \frac{1}{(2\pi)^d} \int_\Omega u_k^T F(\tau) e^{-i\langle k-m,\tau\rangle} v_m d\tau =$$
$$= \frac{1}{(2\pi)^d} \int_\Omega u(\tau)^T F(\tau) v(\tau) d\tau,$$

where

$$u(\tau) = \sum_k u_k e^{-i\langle k,\tau\rangle}, \quad v(\tau) = \sum_m v_m e^{i\langle m,\tau\rangle}.$$

Following [13, Lemma 4]:

$$\|T\|_2 \le \frac{1}{(2\pi)^d} \int_\Omega \|F\|_2 \|u(\tau)\|_2 \|v(\tau)\|_2 d\tau \le$$
$$\le \|F\|_2 \frac{1}{(2\pi)^d} \sqrt{\int_\Omega \|u(\tau)\|_2^2 d\tau} \sqrt{\int_\Omega \|v(\tau)\|_2^2 d\tau} = \|F\|_2 \|u\|_2 \|v\|_2 = \|F\|_2.$$

Analogously to Theorem 1,

$$\|F\|_2 = \sup_{u_1,u_2} |[\![K; u_1, u_2, z_1 \dots z_d]\!]| = \sqrt{h_1 \dots h_d} \sup_{u_1,u_2} |[\![K; u_1, u_2, \frac{z_1}{\sqrt{h_1}} \dots \frac{z_d}{\sqrt{h_d}}]\!]| \le$$
$$\le \sqrt{h_1 \dots h_d} \|K\|_\sigma.$$

Thus, $\|T\|_2 \le \|F\|_2 \le \|K\|_\sigma$, which completes the proof.

E Spectral norm regularization

Tab. E.2 presents accuracy for ResNet18 on CIFAR100 and ResNet34 on ImageNet with different regularizers and regularization coefficients β.

Table E.2: Test accuracy with different regularizers.

Method	β	Acc. w/o wd	Acc. w/ wd
Baseline	0	73.10	73.84
$F4$	0.0016	73.55	74.83
$F4$	0.0018	73.69	74.44
$F4$	0.0022	**73.96**	74.91
TN (Ours)	0.0016	73.77	74.76
TN (Ours)	0.0018	73.63	74.77
TN (Ours)	0.0022	**73.96**	**74.99**

(a) ResNet18 trained on CIFAR100

Method	β	Acc@1	Acc@5
Baseline	0	73.368	**91.438**
$F4$	2e-3	73.034	91.136
$F4$	1e-4	73.388	91.300
$F4$	5e-4	73.322	91.258
TN (Ours)	2e-3	73.086	91.150
TN (Ours)	1e-4	**73.510**	91.420
TN (Ours)	5e-4	73.372	91.326

(b) ResNet34 trained on ImageNet

F Spectral norm of convolutional layers of CNNs

Figure F.1 demonstrates the behaviour of the bounds during the training of ResNet-18, showing that our bound remains an accurate approximation of the exact spectral norm throughout the training process. We reinitialize the singular vectors every epoch from random approximation and use one iteration per training step to update the singular vectors. We observe that this strategy is good enough to maintain the precision during the training process.

To constrain the Lipschitz constant of an entire network, we should take into account spectral norm of each layer. Convolutional layers in CNNs are usually followed by batch normalization. Their concatenation forms a linear transformation for which we can estimate the spectral norm. In our experiments, we apply regularization only to the convolutional layers. Fig. F.2 demonstrates that training with regularization noticeably decreases the spectral norm of convolutions. In addition, our TN bound becomes more accurate when regularization is applied. Although we do not regularize BatchNorm layers, Fig. F.3 shows that the composition's spectral norm decreases when regularization is applied to convolutions. Similarly to [3], in Figure F.4, we compare our method with the alternative approaches for different configurations of kernel tensors.

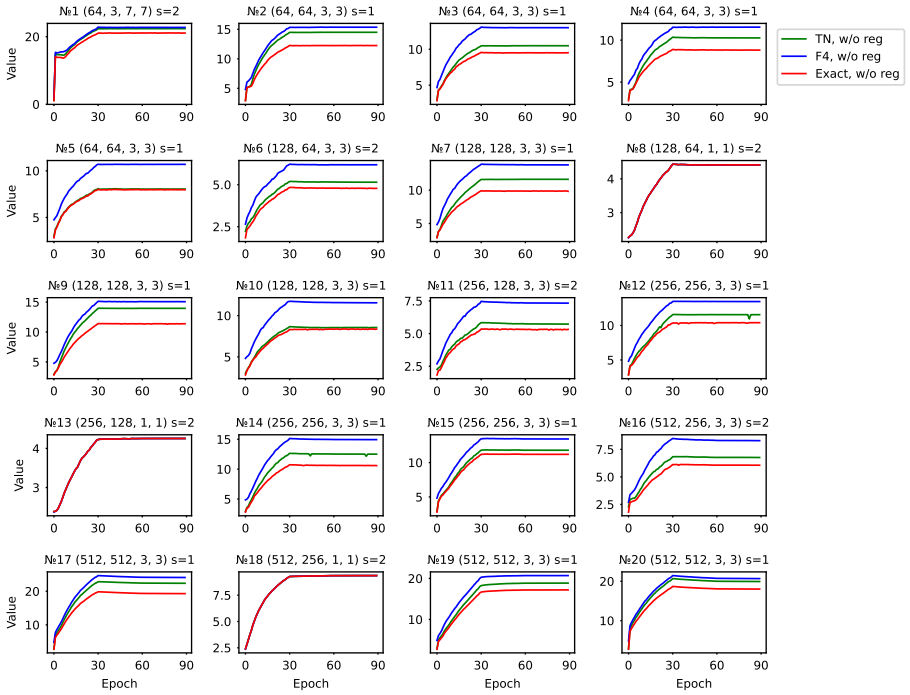

Fig. F.1: The plot compares our TN bound with the $F4$ bound for convolutional layers of ResNet18 trained on CIFAR100. We do not use any regularization or weight decay in this experiment.

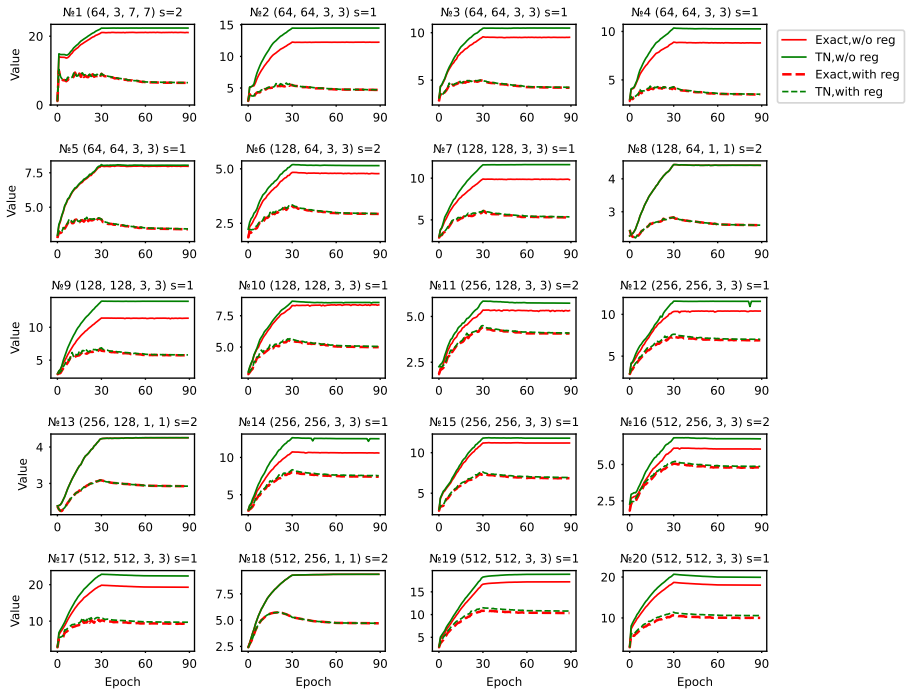

Fig. F.2: Effect of regularization with TN bound on the spectral norm of convolutional layers of ResNet18 trained on CIFAR100.

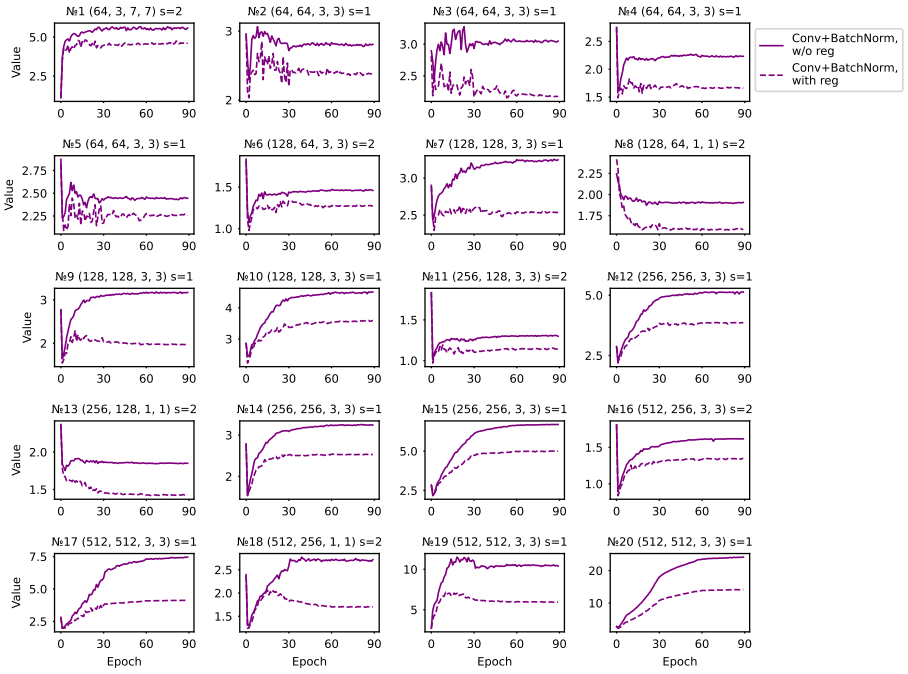

Fig. F.3: The behaviour of the spectral norm of composition of convolution and subsequent BatchNorm layers for ResNet18 trained on CIFAR100 with and without TN regularization.

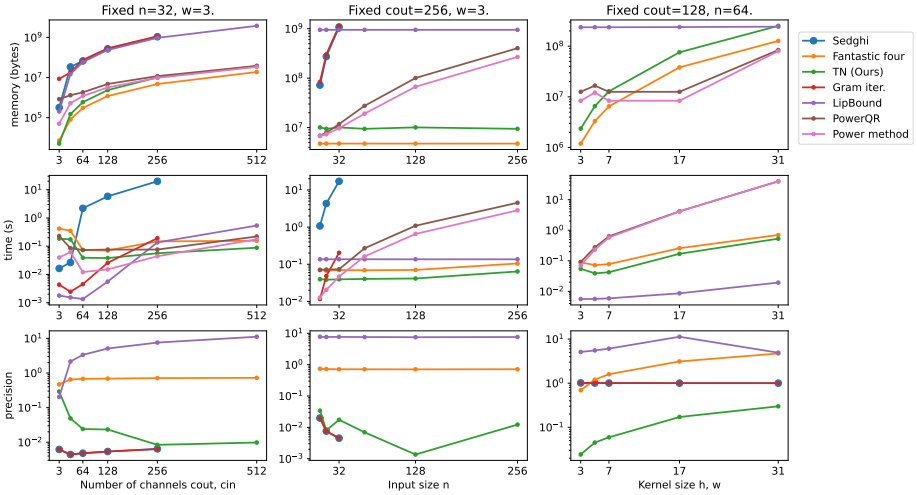

Fig. F.4: Comparison of existing methods in terms of memory consumption, time efficiency and precision for convolution with zero padding and kernels with entries sampled from $\mathcal{N}(0,1)$. We measure the precision as $|\sigma_{method} - \sigma_{ref}|/\sigma_{ref}$, where σ_{ref} is a highly accurate reference value obtained using the power method. We do not plot precision of PowerQR [5] as it gives the exact value. The power method and PowerQR [5, 9] are accurate, but their time complexity noticeably depends on n and c_{out}. LipBound [1] produces errors larger than the other methods. Gram iteration [3] is fast, but consumes as much memory as the method by Sedghi *et al.* [10] and is inapplicable for large c_{out}, c_{in} and n. Our method is memory efficient and provides a trade-off between speed and accuracy, improving the Fantastic four bound [12].

References

1. Araujo, A., Negrevergne, B., Chevaleyre, Y., Atif, J.: On lipschitz regularization of convolutional layers using toeplitz matrix theory. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 6661–6669 (2021)
2. De Silva, V., Lim, L.H.: Tensor rank and the ill-posedness of the best low-rank approximation problem. SIAM Journal on Matrix Analysis and Applications **30**(3), 1084–1127 (2008)
3. Delattre, B., Barthélemy, Q., Araujo, A., Allauzen, A.: Efficient bound of lipschitz constant for convolutional layers by gram iteration. In: International Conference on Machine Learning. pp. 7513–7532. PMLR (2023)
4. Draisma, J., Ottaviani, G., Tocino, A.: Best rank-k approximations for tensors: generalizing eckart–young. Research in the Mathematical Sciences **5**(2), 27 (2018)
5. Ebrahimpour-Boroojeny, A., Telgarsky, M., Sundaram, H.: Spectrum extraction and clipping for implicitly linear layers. In: NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning (2023)
6. Friedland, S.: Best rank one approximation of real symmetric tensors can be chosen symmetric. Frontiers of Mathematics in China **8**, 19–40 (2013)
7. Friedland, S., Lim, L.H.: Nuclear norm of higher-order tensors. Mathematics of Computation **87**(311), 1255–1281 (2018)
8. Friedland, S., Wang, L.: Spectral norm of a symmetric tensor and its computation. Mathematics of Computation **89**(325), 2175–2215 (2020)
9. Ryu, E., Liu, J., Wang, S., Chen, X., Wang, Z., Yin, W.: Plug-and-play methods provably converge with properly trained denoisers. In: International Conference on Machine Learning. pp. 5546–5557. PMLR (2019)
10. Sedghi, H., Gupta, V., Long, P.M.: The singular values of convolutional layers. In: International Conference on Learning Representations (2018)
11. Senderovich, A., Bulatova, E., Obukhov, A., Rakhuba, M.: Towards practical control of singular values of convolutional layers. Advances in Neural Information Processing Systems **35**, 10918–10930 (2022)
12. Singla, S., Feizi, S.: Fantastic four: Differentiable and efficient bounds on singular values of convolution layers. In: International Conference on Learning Representations (2020)
13. Yi, X.: Asymptotic spectral representation of linear convolutional layers. IEEE Transactions on Signal Processing **70**, 566–581 (2022)