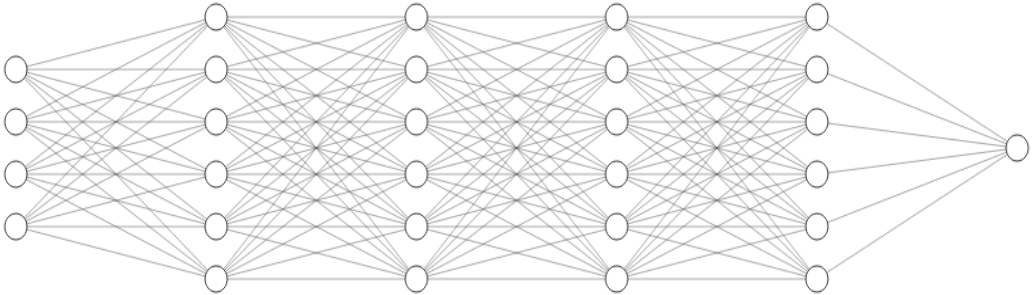
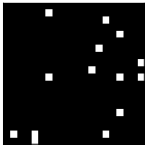
Low-rank lottery tickets: finding efficient low-rank neural networks via matrix differential equations

Steffen Schotthöfer, Emanuele Zangrando, Jonas Kusch, Gianluca Ceruti, Francesco Tudisco

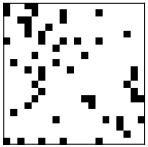
KIT, GERMANY — GSSI, ITALY — EPFL, SWITZERLAND — UNI INNSBRUCK, AUSTRIA



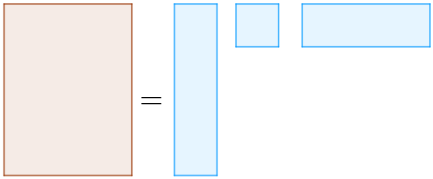
Model compression - an overview



(a) full-rank representation



(b) sparse representation



(c) low-rank representation

We want to compress neural networks **during** training using low-rank factorization.
Goals of **D**ynamical **L**ow-**R**ank **T**raining (DLRT):

- Low training and inference memory footprint
- Fast training and inference
- High accuracy of the compressed network

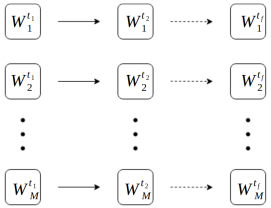
DLRT: Idea

Interpret the gradient update

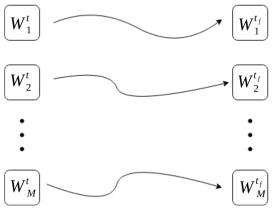
$$W^{t+1} = W^t - \eta \nabla_W \mathcal{L}, \qquad t = 1, 2, \dots \tag{1}$$

as a continuous time gradient flow. I.e.,

$$\dot{W}(t) = -\nabla_W \mathcal{L}, \qquad W(t_0) = W_0 \tag{2}$$



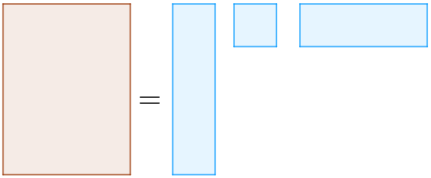
(a) discrete gradient update



(b) time-continous gradient flow

The continous time gradient flow system is **low-rank** for neural networks!

Dynamical Low-Rank Updates



$$W = U \quad S \quad V^\top$$

Find update rules for the low-rank factorization.

- Update the low-rank basis
- Expand the low-rank manifold
- Update the low-rank coefficients
- Reduce the rank of the low-rank manifold

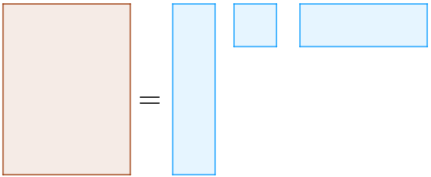
Three low-rank updates instead of one full-rank update!

Dynamical Low-Rank Updates

Find update rules for the low-rank factorization.

- Update the low-rank basis
- Expand the low-rank manifold
- Update the low-rank coefficients
- Reduce the rank of the low-rank manifold

Three low-rank updates instead of one full-rank update!



$$W = U \quad S \quad V^\top$$

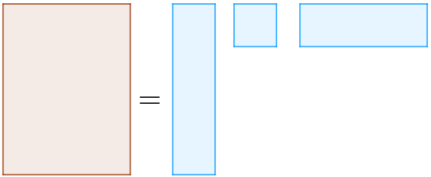
$$U^t \to U^{t+1} \qquad V^{t,\top} \to V^{t+1,\top}$$

Dynamical Low-Rank Updates

Find update rules for the low-rank factorization.

- Update the low-rank basis
- Expand the low-rank manifold
- Update the low-rank coefficients
- Reduce the rank of the low-rank manifold

Three low-rank updates instead of one full-rank update!



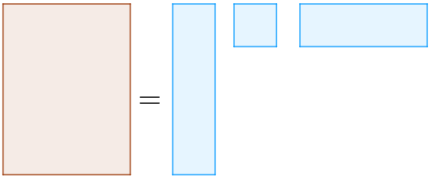
$$W = U \quad S \quad V^\top$$

$$S^t \to \widetilde{S}^t$$

Dynamical Low-Rank Updates

Find update rules for the low-rank factorization.

- Update the low-rank basis
- Expand the low-rank manifold
- Update the low-rank coefficients
- Reduce the rank of the low-rank manifold



$$W = U \quad S \quad V^\top$$

Three low-rank updates instead of one full-rank update!

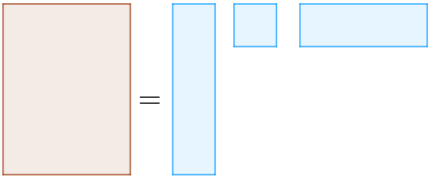
$$\widetilde{S}^t \to \widetilde{S}^{t+1}$$

Dynamical Low-Rank Updates

Find update rules for the low-rank factorization.

- Update the low-rank basis
- Expand the low-rank manifold
- Update the low-rank coefficients
- Reduce the rank of the low-rank manifold

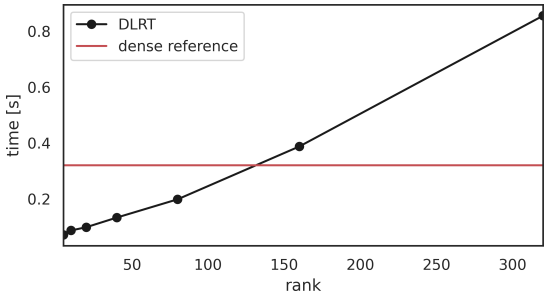
Three low-rank updates instead of one full-rank update!



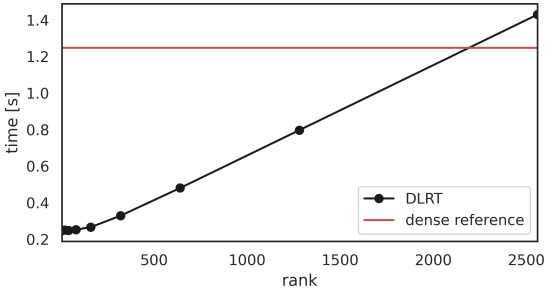
$$W = U \quad S \quad V^\top$$

$$\widetilde{S}^{t+1} \to S^{t+1}$$

DLRT efficiency



(a) Training timings

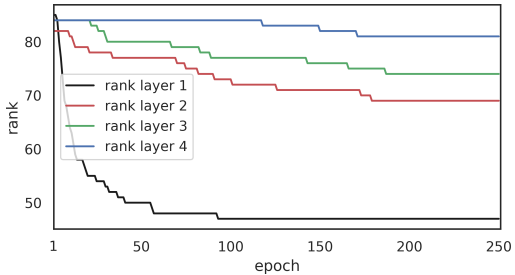


(b) Prediction timings

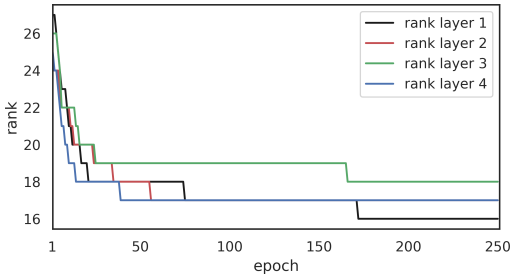
- DLRT **training is one order of magnitude faster** than the dense training for $r \ll M, N$
- DLRT **inference timing is one order of magnitude faster**

DLRT rank evolution

DLRT compression of a 5-layer, rank 500 network on MNIST.



(a) $\tau = 0.05$



(b) $\tau = 0.15$

- The only hyperparameter of DLRT is the compression rate τ
- τ determines, how aggressive we cut off singular values of S
- DLRT enables **fast initial compression** and convergence of the solution rank

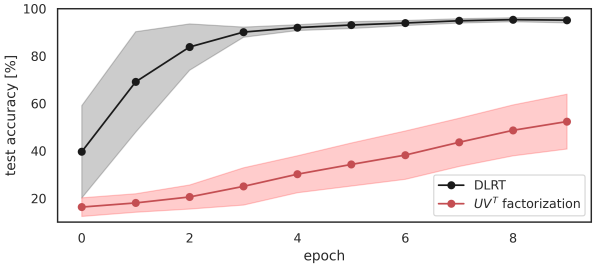
Compression results

- DLRT achieves SOTA compression during inference
- DLRT is able to compress networks during training

	ImageNet1k			
		test acc.[%]	compression rate	
	method	(to baseline)	eval[%]	train[%]
ResNet-50	**DLRT**	**−0.56**	**54.1**	**14.2**
	PP-2[SKVRN19]	−0.8	52.2	< 0
	PP-1[SKVRN19]	−0.2	44.2	< 0
	CP[HZS17]	−1.4	50.0	< 0
	SFP[HKD$^+$18]	−0.2	41.8	< 0
	ThiNet[LWL17]	−1.5	36.9	< 0
VGG16	**DLRT**	**−2.19**	**86**	**78.4**
	PP-1[SKVRN19]	−0.19	80.2	< 0
	CP[HZS17]	−1.80	80.0	< 0
	ThiNet[LWL17]	−0.47	69.04	< 0
	RNP(3X)[LRLZ17]	−2.43	66.67	< 0

DLRT stability

Training on low-rank manifolds is a non-trivial task.



(a) Training performance, using $W(t_0)$ with exponentially decaying singular values

- **DLRT is robust** with respect to weight initializations
- Vanilla factorization $W = UV^\top$ with direct gradient descent is **not robust** with respect to weight initializations

Thank you for your attention!

We welcome you to our poster session!

Contact
steffen.schotthoefer@kit.edu
emanuele.zangrando@gssi.it

jonas.kusch1@gmail.com
gianluca.ceruti@epfl.ch
francesco.tudisco@gssi.it

References I

- Yang He, Guoliang Kang, Xuanyi Dong, Yanwei Fu, and Yi Yang, *Soft filter pruning for accelerating deep convolutional neural networks*, 2018.
- Yihui He, Xiangyu Zhang, and Jian Sun, *Channel pruning for accelerating very deep neural networks*, 2017.
- Ji Lin, Yongming Rao, Jiwen Lu, and Jie Zhou, *Runtime neural pruning*, Advances in Neural Information Processing Systems (I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, eds.), vol. 30, Curran Associates, Inc., 2017.
- Jian-Hao Luo, Jianxin Wu, and Weiyao Lin, *Thinet: A filter level pruning method for deep neural network compression*, 2017.

References II

Pravendra Singh, Vinay Kumar Verma, Piyush Rai, and Vinay P. Namboodiri, *Play and prune: Adaptive filter pruning for deep model compression*, Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19, International Joint Conferences on Artificial Intelligence Organization, 7 2019, pp. 3460–3466.