LEARNING IMPLICITLY RECURRENT CNNS THROUGH PARAMETER SHARING

Pedro Savarese
TTI-Chicago

Michael Maire
University of Chicago

TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO

THE UNIVERSITY OF CHICAGO

Approach:

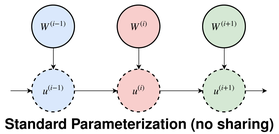
- Share parameters across layers in a neural network
- Soft sharing: relaxation of discrete optimization problem of selecting layers from a pool

Consequences:

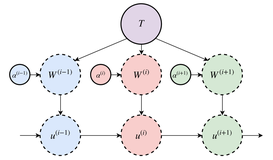
- Decouples parameter count from network depth
- Imposes inductive bias similar to RNNs
- Benefits optimization: accelerates learning of layers with similar functionality (redundant parameters → shared parameters)

Experimental Results:

- Classification: better accuracy *and* smaller CNNs
- Learning may discover explicitly recurrent architectures (new form of architecture search)
- Synthetic task: improved extrapolation ability when learning to emulate algorithmic procedure



Standard Parameterization (no sharing)



Parameter Sharing (re-parameterization)

SOFT PARAMETER SHARING

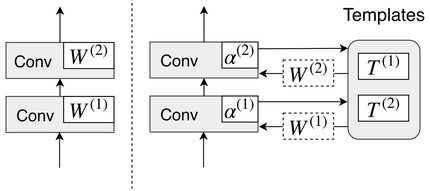
- Trained networks present excessive redundancy
- Possible cause: different layers cannot re-use parameters
- Proposed method: layers learn to share parameters, alternatively seen as re-parameterization

$$W^{(i)} = \sum_{j=1}^{k} \alpha_j^{(i)} T^{(j)} \qquad u^{(i)} = W^{(i)} * u^{(i-1)} = \sum_{j=1}^{k} \alpha_j^{(i)} \left(T^{(j)} * u^{(i-1)} \right)$$

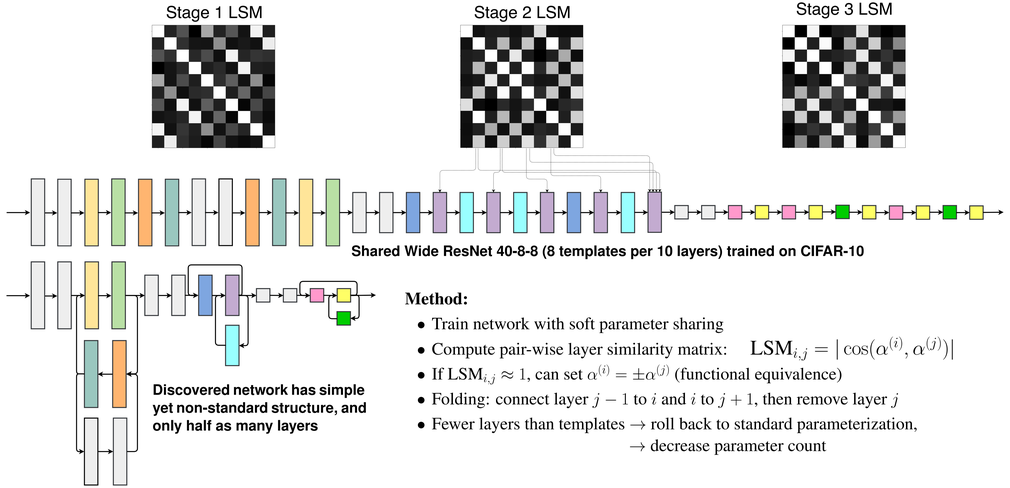
- $T^{(j)} \in \mathbb{R}^d$: j'th parameter template, shared among layers, same size as $W^{(i)} \in \mathbb{R}^d$
- $\alpha^{(i)} \in \mathbb{R}^k$: bound to layer i, learnable mixing coefficients
- Fully differentiable: trained end-to-end with SGD

$$\nabla_{T^{(j)}} \mathcal{L} = \sum_{i=1}^{L} \alpha_j^{(i)} \nabla_{W^{(i)}} \mathcal{L} \qquad \nabla_{\alpha_j^{(i)}} \mathcal{L} = \langle T^{(j)}, \nabla_{W^{(i)}} \mathcal{L} \rangle$$

- Number of parameters: $k(L+d)$ instead of $L \cdot d$
- α initialization: sparse, orthogonal



FOLDING LEARNED NETWORKS INTO EXPLICITLY RECURRENT FORMS



Method:

- Train network with soft parameter sharing
- Compute pair-wise layer similarity matrix: $\text{LSM}_{i,j} = \left| \cos(\alpha^{(i)}, \alpha^{(j)}) \right|$
- If $\text{LSM}_{i,j} \approx 1$, can set $\alpha^{(i)} = \pm\alpha^{(j)}$ (functional equivalence)
- Folding: connect layer $j-1$ to i and i to $j+1$, then remove layer j
- Fewer layers than templates → roll back to standard parameterization, → decrease parameter count

IMPROVED PARAMETER EFFICIENCY

Baseline Models:

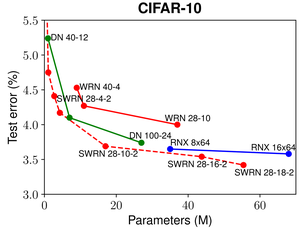
- Wide Residual Networks (WRNs) [7]
- WRN L-w
  - L layers
  - widen factor w
- ResNet, ResNeXt [5], DenseNet [1] all serve as additional comparison points

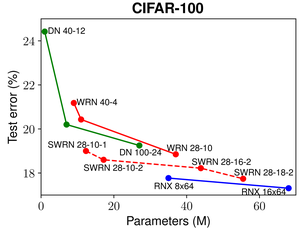
Our Models:

- Shared Wide Residual Networks (SWRNs)
- SWRN L-w-k
  - k parameter templates per group of layers
  - Groups consist of layers with the same shape parameter tensor
  - Setting $k = \frac{L-4}{3} - 2$ yields one template per layer; denote such model as SWRN-L-w

Results:

- Performance boost at one template per layer (*i.e.* without any parameter reduction)
- 2 templates per 6 layers (SWRN 28-XX-2): even better on CIFAR-10, competitive on CIFAR-100





CIFAR	Params ($\times 10^6$)	Error (%) C-10+	C-100+
ResNeXt-29 16x64	68	3.58	17.31
DenseNet 100-24	27	3.74	19.25
DenseNet 190-40	26	3.46	17.18
WRN 28-10	36	4.0	19.25
WRN 28-10*	36	3.89	18.85
SWRN 28-10*	36	3.88	18.43
SWRN 28-10-2*	17	3.75	18.66
SWRN 28-14*	71	3.67	18.25
SWRN 28-14-2*	33	3.69	18.37
SWRN 28-18*	118	3.48	**17.43**
SWRN 28-18-2*	55	**3.43**	17.75

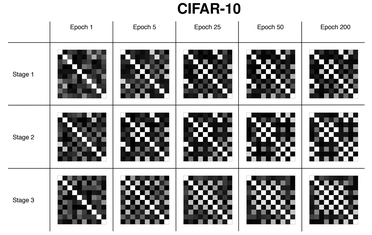
Note: models marked with * are trained with dropout 0.3

ImageNet	Params ($\times 10^6$)	Error (%) Top-1	Top-5
WRN 50-2	69	22.0	6.05
DenseNet-264	33	22.15	6.12
ResNet-200	65	21.66	5.79
SWRN 50-2	69	21.74	5.95

EVOLUTION OF SHARING PATTERNS

- Structure emerges quickly in the first epochs of CIFAR-10 training
- Folding possible after only 2% of total training iterations
- Faster evolution & stronger patterns via regularizing the LSM:

LSM evolution for 100-layer CNN trained on synthetic task



COMPARISON TO NEURAL ARCHITECTURE SEARCH

- Faster training than recent gradient-based NAS, but …
- … less parameter efficient
- Non-NAS techniques (*e.g.* cutout, auxiliary towers, bi-level optimization) contribute to gains reported in many NAS results
- Cutout regularization boosts baseline WRN model …
- … and our SWRN models (see Table to the right)
- Our search over connection patterns is orthogonal to traditional NAS space

CIFAR-10	Params ($\times 10^6$)	Training Time (GPU Days)	Test Error (%)
NASNet-A [8]	3.3	1800	2.65
NASNet-A [8]	27.6	1800	2.4
AmoebaNet-B [4]	2.8	3150	2.55
AmoebaNet-B [4]	13.7	3150	2.31
AmoebaNet-B [4]	34.9	3150	2.13
DARTS [2]	3.4	4	2.83
SNAS [6]	2.8	1.5	2.85
ENAS [3]	4.6	0.45	2.89
WRN 28-10 (baseline with cutout)	36.4	0.4	3.08
SWRN 28-6-2 (ours)	6.1	0.25	3.0
SWRN 28-10-2 (ours)	17.1	0.4	2.69
SWRN 28-14-2 (ours)	33.5	0.7	2.53

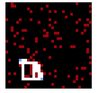
EXTRAPOLATION BIAS

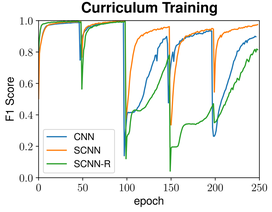
Challenge - Learn Shortest-Paths Algorithm:

- $32 \times 32 \times 2$ input images, 2D grid graph
- Two query points (blue) & many obstacles (red)
- Task: predict every pixel which lies on some shortest path between query points
- Curriculum: 50 epochs per phase, distance between query points increases

Results:

- Faster adaptation to harder examples
- Better overall accuracy





[1] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely Connected Convolutional Networks. *CVPR*, 2017.

[2] H. Liu, K. Simonyan, and Y. Yang. DARTS: Differentiable Architecture Search. *ICLR*, 2019.

[3] H. Pham, M. Guan, B. Zoph, Q. V. Le, and J. Dean. Efficient Neural Architecture Search via Parameters Sharing. *ICML*, 2018.

[4] E. Real, A. Aggarwal, Y. Huang, and Q. V. Le. Regularized Evolution for Image Classifier Architecture Search. *AAAI*, 2019.

[5] S. Xie, R. B. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated Residual Transformations for Deep Neural Networks. *CVPR*, 2017.

[6] S. Xie, H. Zheng, C. Liu, and L. Lin. SNAS: Stochastic Neural Architecture Search. *ICLR*, 2019.

[7] S. Zagoruyko and N. Komodakis. Wide Residual Networks. *BMVC*, 2016.

[8] B. Zoph and Q. V. Le. Neural Architecture Search with Reinforcement Learning. *ICLR*, 2017.