Learning Block-Sparse Neural Networks

Huanru Henry Mao, Shuyang Li, Yiting Ethan Li, Jonathan Margoliash

Roadmap

1. Motivation
2. Problem Formulation
3. Experiments
4. Sparsity and Efficiency
5. Conclusions

Motivation

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." *Advances in neural information processing systems*. 2012.

Deep Neural Nets

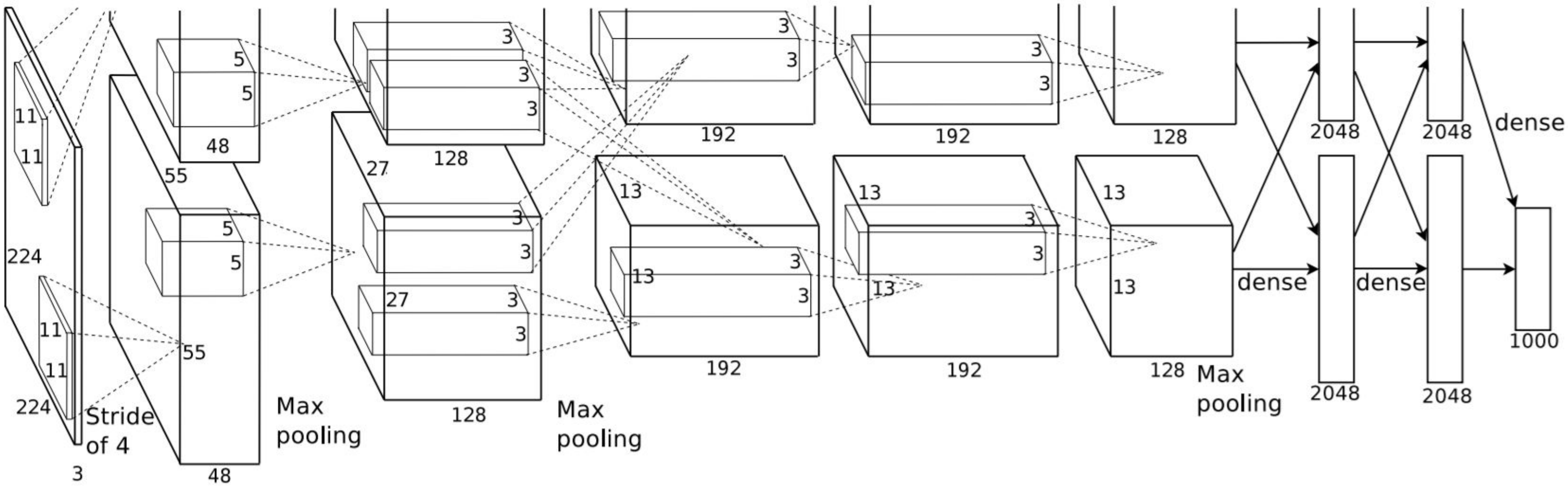
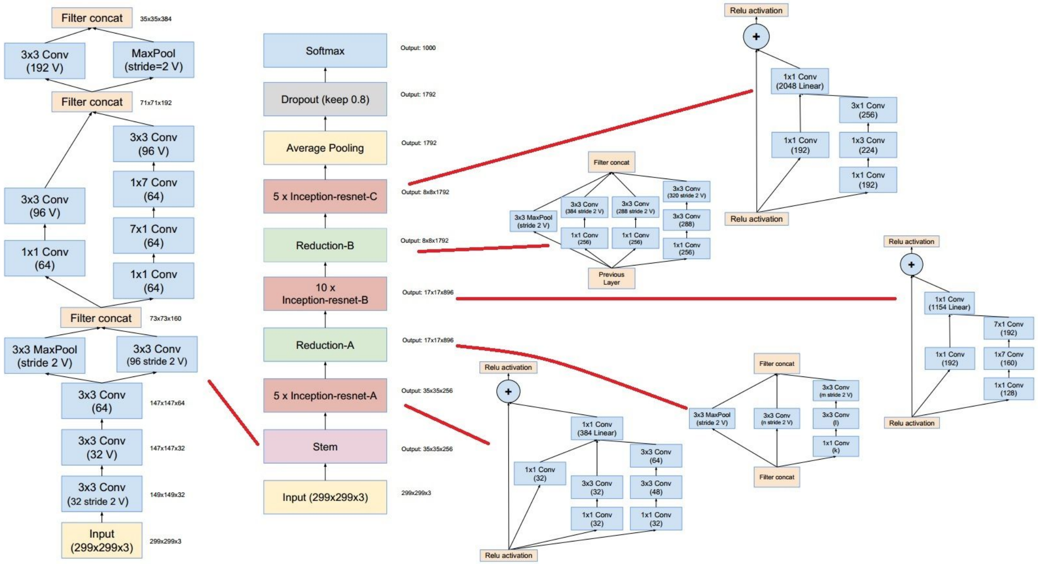


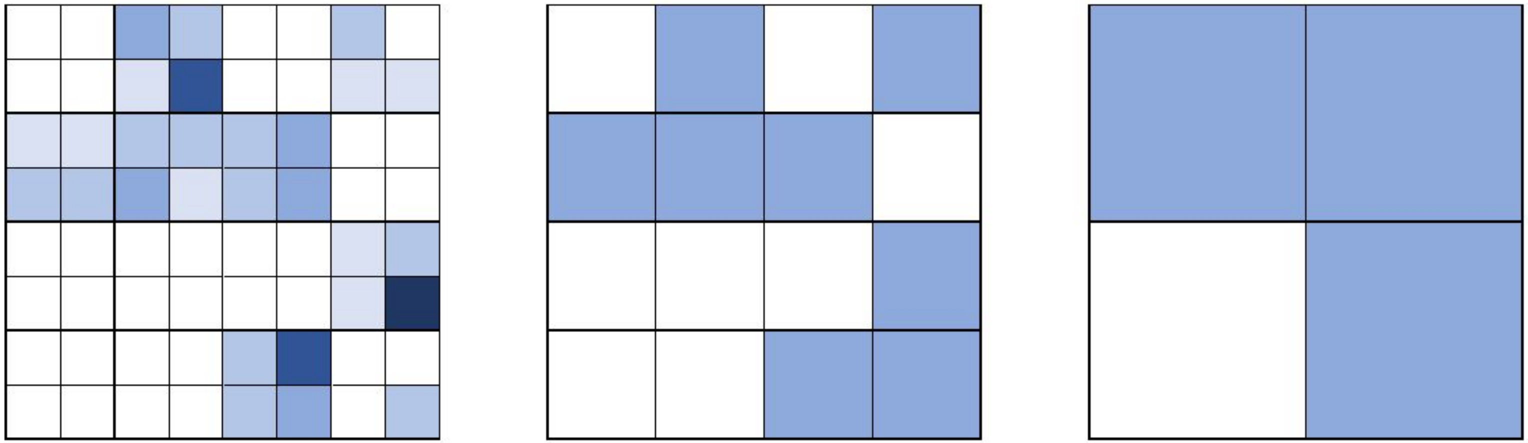
Image source: http://yeephycho.github.io/2016/08/31/A-reminder-of-algorithms-in-Convolutional-Neural-Networks-and-their-influences-III/

Cost of Complexity



- ~10 million parameters
- Time(Training) ~ # params
- Time(Use) ~ # params

Block Sparsity



Ozcan, Ahmet S. "Filopodia: a rapid structural plasticity substrate for fast learning." *Frontiers in synaptic neuroscience* 9 (2017): 12.

Brain Development

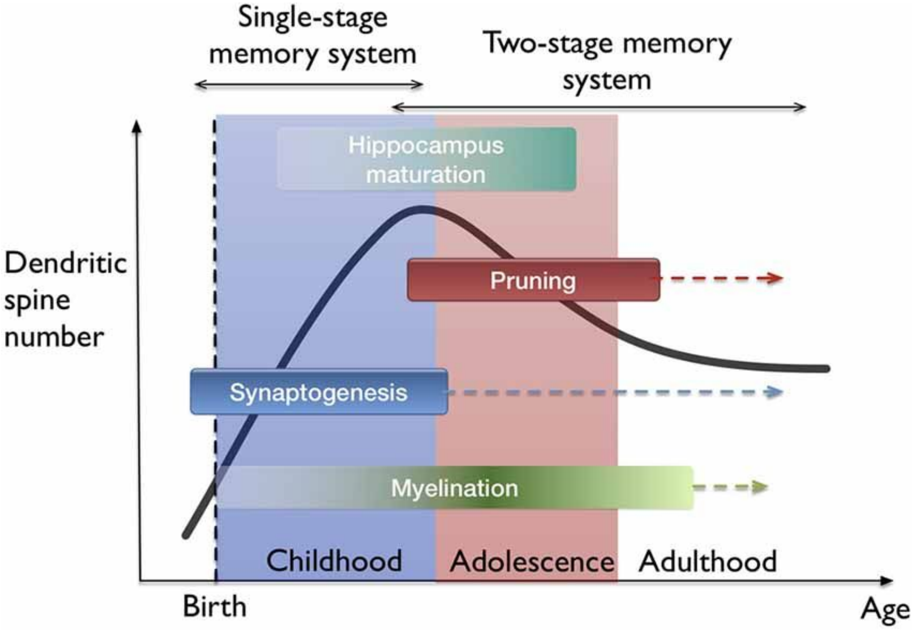
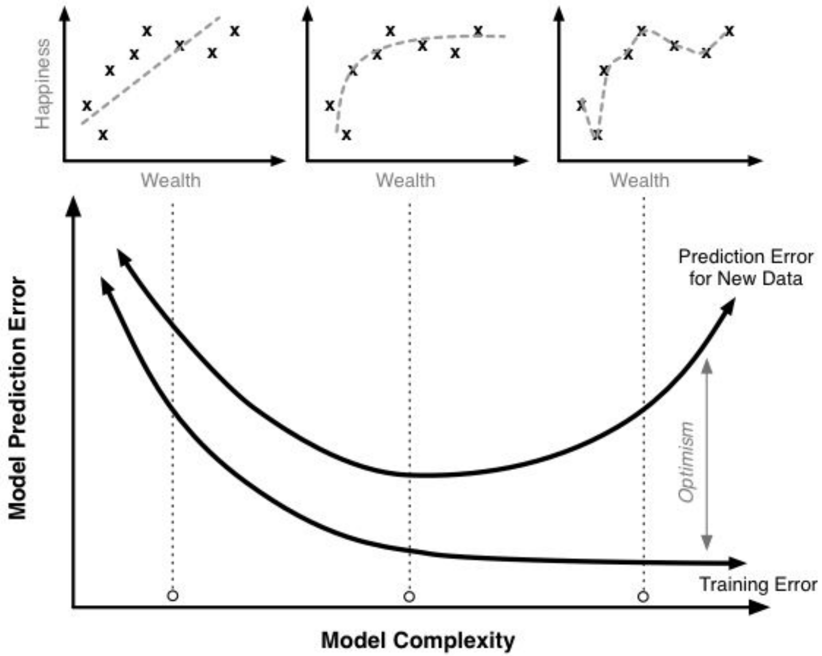


Image source: http://scott.fortmann-roe.com/docs/MeasuringError.html

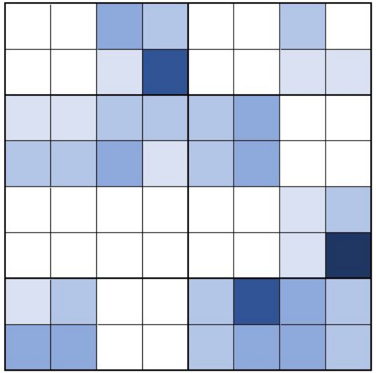
Excessive Model Complexity



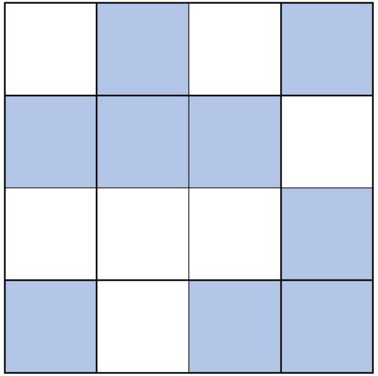
Problem Formulation

“Optimize our loss function with a network containing at most k blocks with nonzero weights”

Primal Problem



$$\hat{W} = \{\|W_i\|_0, \quad i = 1, \ldots b\}$$



Primal Problem

$$\begin{aligned} \min_{W} \quad & f(W) \\ \text{s. t.} \quad & \|\hat{W}\|_0 \le k \end{aligned}$$

$$\boxed{\hat{W} = \{\|W_i\|_0, \quad i = 1, \ldots b\}}$$

Primal Relaxation

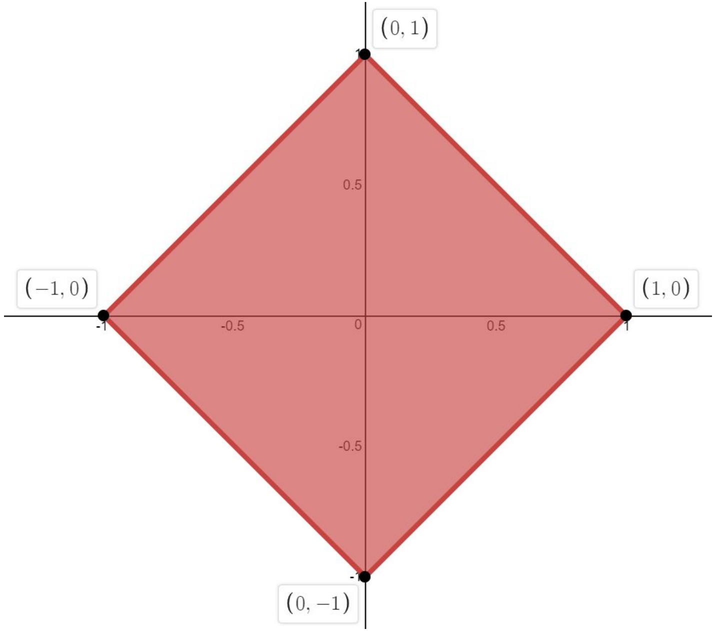
$$\min_{W} \quad f(W)$$
$$\text{s. t.} \quad \|\hat{W}\|_0 \le k$$

$$\|\hat{W}\|_1$$

$$\hat{W} = \{\|W_i\|_0, \quad i = 1, \ldots b\}$$

$$\|W_i\|_2$$

L1 Norm as L0 Norm Relaxation



Primal Relaxation (Group Lasso)

$$\min_{W} \quad f(W)$$

$$\text{s. t.} \quad \sum_{i=1}^{b} \|W_i\|_2 \leq k$$

Dual Problem

$$\max_{\lambda} \quad g(\lambda) = \inf_{W} \mathcal{L}(W, \lambda)$$

$$= \inf_{W} \{ f(W) + \lambda \sum_{i=1}^{b} \|W_i\|_2 \} - k\lambda$$

$$\text{s. t.} \quad \lambda \geq 0$$

Empirical Risk Minimization

$$W^* = \arg\min_W \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i, W)) + \lambda(\sum_{i=1}^{b} \|W_i\|_2 - k)$$

$$W^* = \arg\min_W \frac{1}{N}\sum_{i=1}^{N} L(y_i, f(x_i, W)) + \lambda \max(\sum_{i=1}^{b} \|W_i\|_2 - k, 0)$$

Sparsity and Efficiency

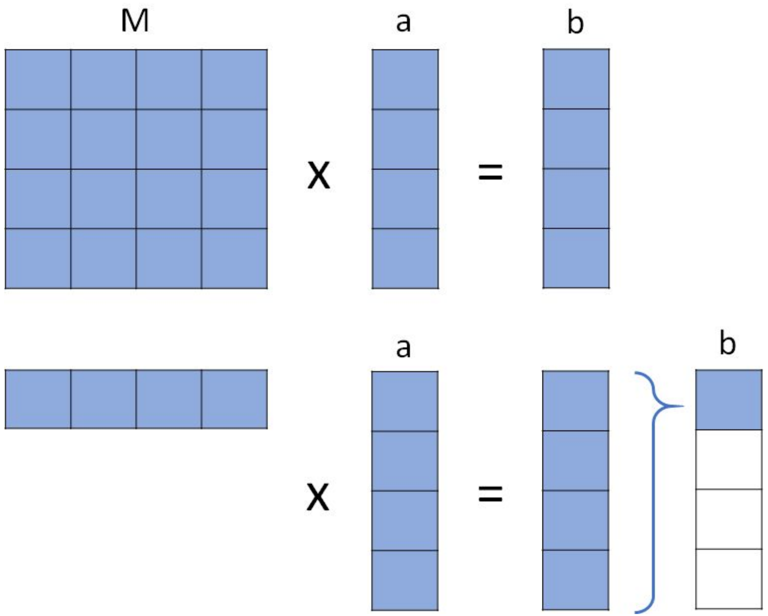
Sparsity

$$\rho = \frac{K}{n^2} \quad \text{Sparsity}$$

$$K = \#\ \text{empty blocks}$$

$$K_i = \#\ \text{empty blocks in row } i$$

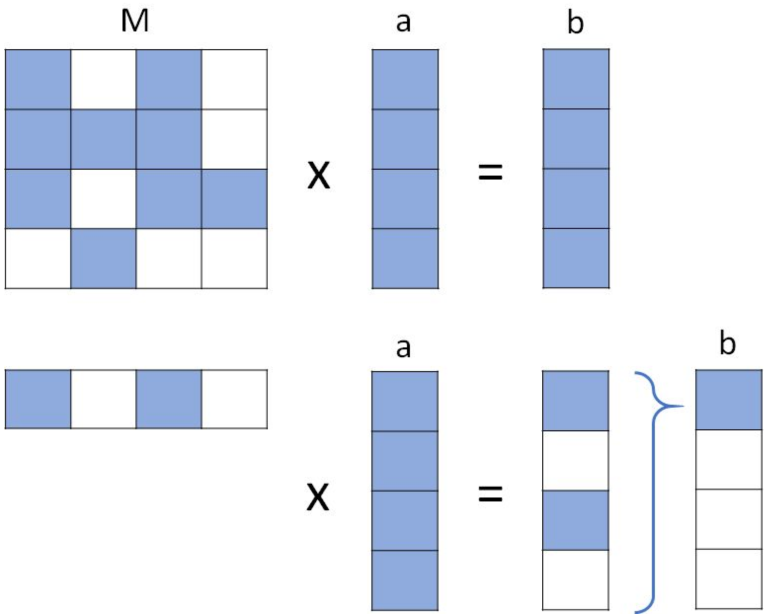
Matrix Multiplication



$$b_i = \sum_{j=0}^{n} M_{ij} a_i$$

$$O(dense) = \sum_{i}^{n} n = n^2$$

Sparse Matrix Multiplication

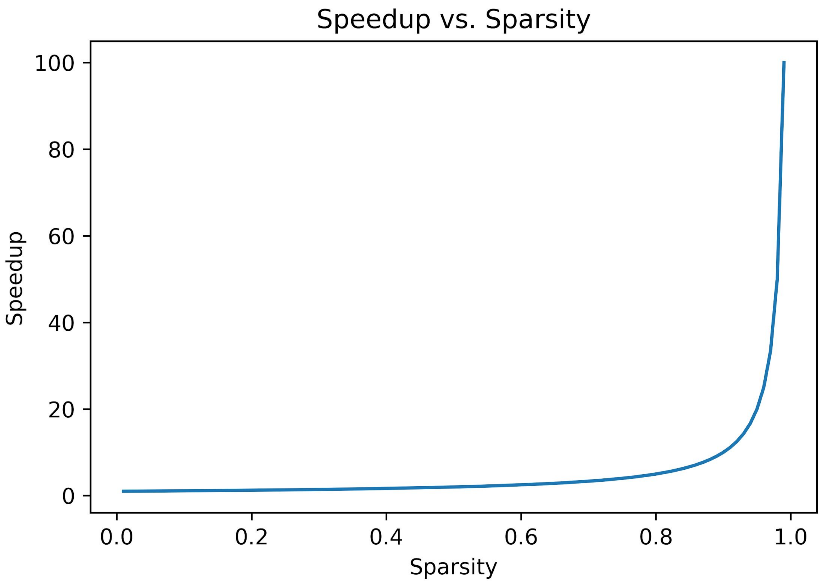


$$b_i = \sum_{j=0}^{n} \mathbb{I}_{\{M_{ij} \neq 0\}} M_{ij} a_i$$

$$O(sparse) = \sum_{i}^{n} (n - K_i) = n^2 - \sum_{i}^{n} K_i$$
$$= n^2 - K$$

Training Speedup

$$Speedup = \frac{O(dense)}{O(sparse)} = \frac{n^2}{n^2 - K}$$
$$= (1 - \frac{K}{n^2})^{-1} = \frac{1}{1-\rho}$$



Experiments

Task

- Language Modeling using Recurrent Neural Networks
- Important application in a variety of downstream natural language processing tasks

Chain Rule Factorization:

$$p(x_1, ..., x_T) = \prod_{t=1}^{T} p(x_t \mid x_{t-1}, ..., x_1).$$

Language Modeling Example

Reviewers	were	satisfied	with	the	smaller	Super	Mario	Bros.
Reviewers	were	satisfied	with	the	smaller	Super	Mario	Bros.
Reviewers	were	satisfied	with	the	smaller	Super	Mario	Bros.

...

Reviewers	were	satisfied	with	the	smaller	Super	Mario	Bros.
Reviewers	were	satisfied	with	the	smaller	Super	Mario	Bros.

Objective Function

Standard Cross Entropy Loss over the vocabulary (all possible words) at each time step.

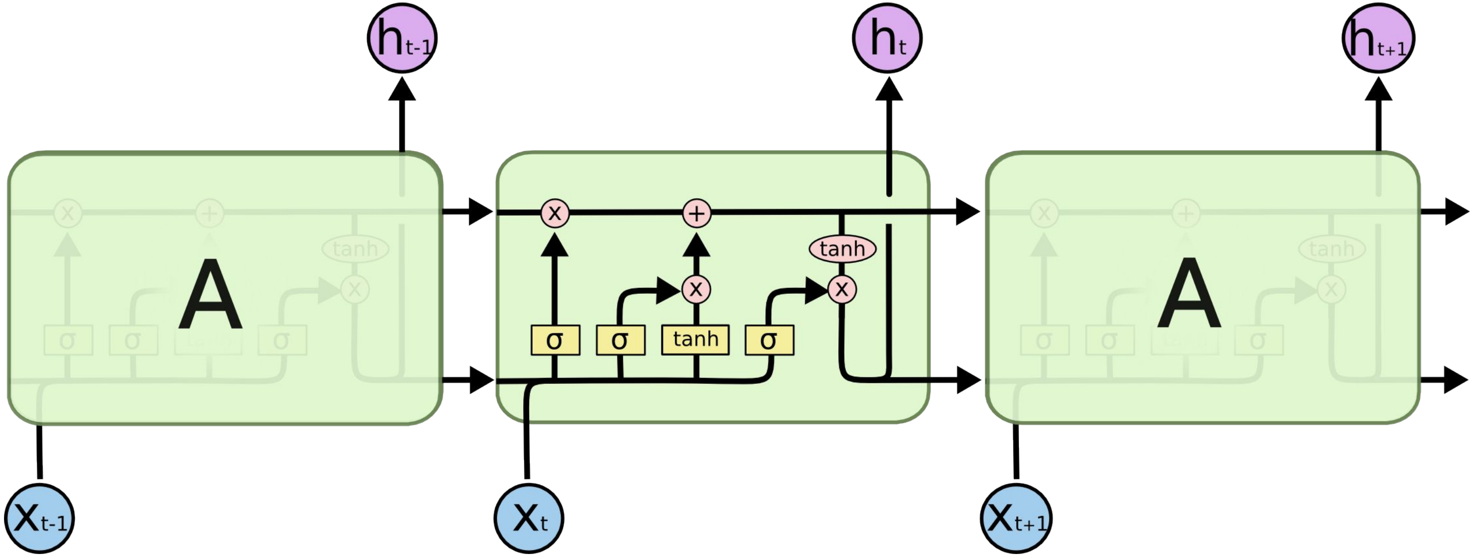
$$-\sum_{c=1}^{M} y_{o,c} \log(p_{o,c})$$

Dataset

- WikiText-2: 100+ million tokens extracted from Wikipedia
- Standard language modeling benchmark dataset

	WikiText-2		
	Train	**Valid**	**Test**
Articles	600	60	60
Tokens	2,088,628	217,646	245,569
Vocab	33,278		
OoV	2.6%		

Long Short Term Memory (LSTM)



Block-Sparse LSTM

$$
\begin{aligned}
f_t &= \sigma_g(\boxed{W_f} x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(\boxed{W_i} x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(\boxed{W_o} x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(\boxed{W_c} x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}
$$

Block-Sparse LSTM

$$\boxed{\begin{matrix} W_f \\ W_i \\ W_o \end{matrix}}$$

$$\boxed{W_c}$$

Pruning While Training

Problem with **constraint relaxation**: there are rarely true zeros that can be skipped in matrix multiplication. Training cannot benefit from block sparsity speedup.

Idea: heuristically set blocks with small weights to true zeros.

Algorithm:

1. Train model on the objective function with block lasso loss
2. Set and freeze lowest K% of blocks (by a block's L2 norm) to zero
3. Gradually increase K until it reaches target sparsity
4. Repeat step 1

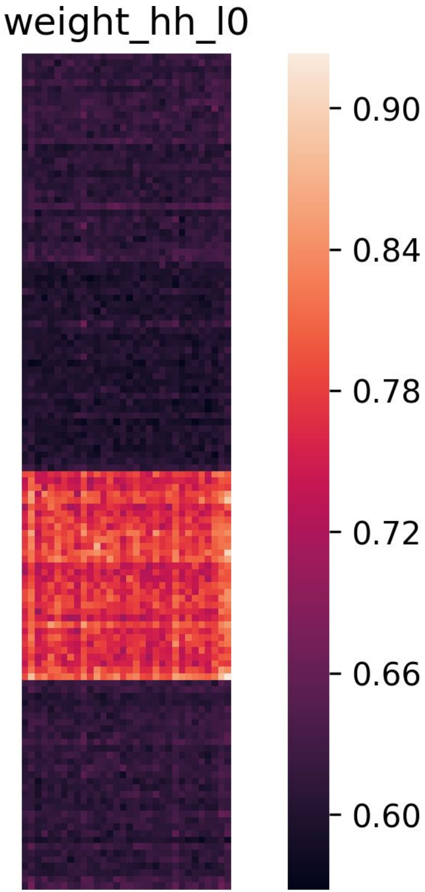
Model Experiments

- AWD LSTM Baseline (Unconstrained problem)
- LSTM with Block Lasso (Constrained problem)
- LSTM with Block Lasso and Gradual Pruning (Constrained problem)
- LSTM with Block Lasso and Random Pre-pruning (Constrained problem)

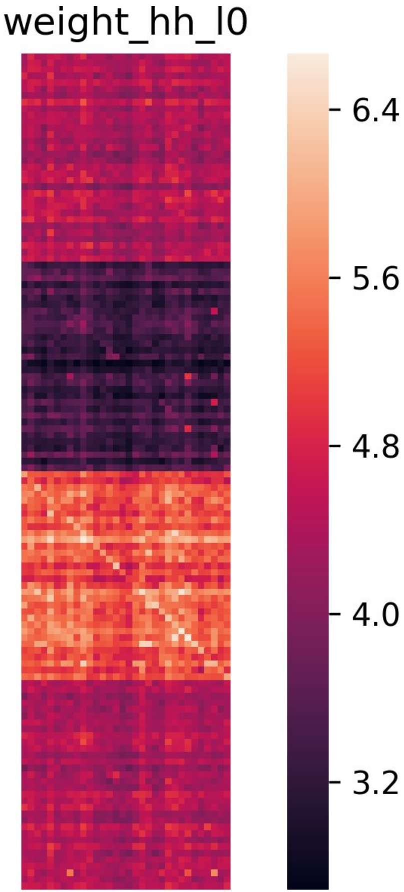
Experiments

	PPL (Train)	PPL (Val)	PPL (Test)	Sparsity	Speedup
AWD LSTM Baseline	79.13	86.81	81.76	0%	1x
+ 1e-4 Block Lasso	132.52	115.43	108.06	Target 80%	1x
+ 1e-4 Block Lasso + Gradual Linear Pruning	189.62	151.05	140.87	80%	1.9x
+ 1e-4 Block Lasso + Pre-Pruning	201.72	158.71	148.37	80%	5x

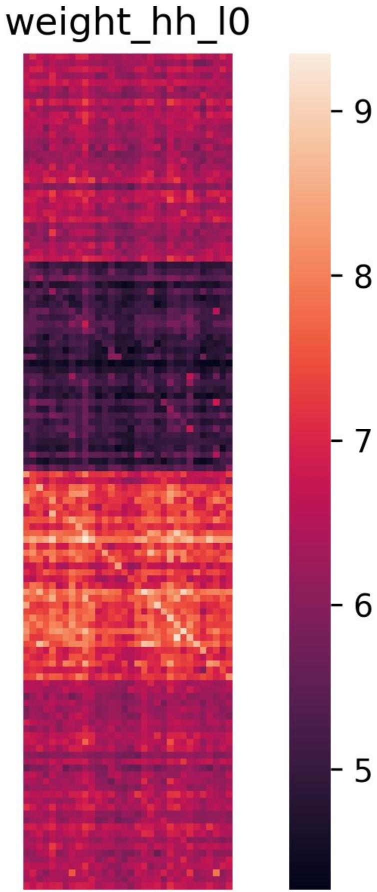
WikiText-2 Dataset Language Modeling with 100 epochs of training



Epoch 0

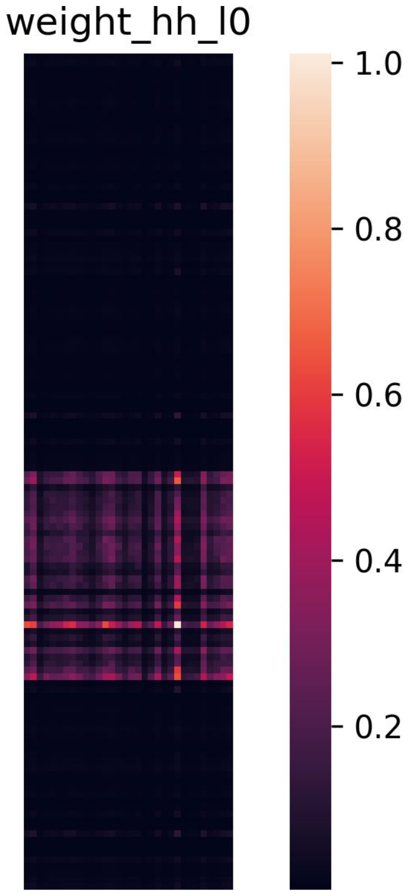


Epoch 50

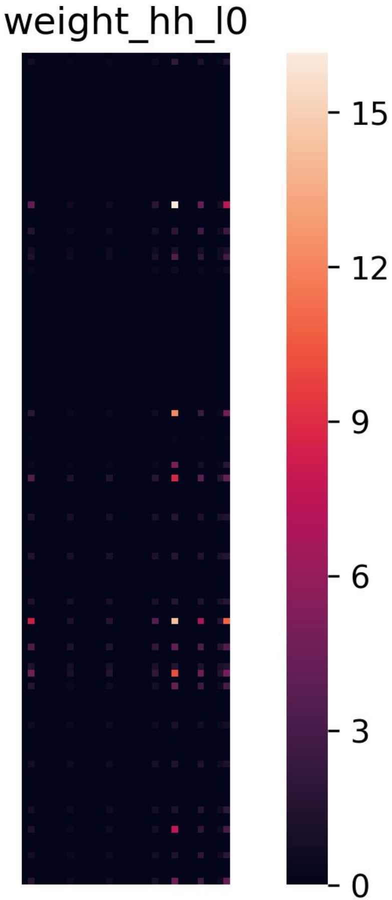


Epoch 100

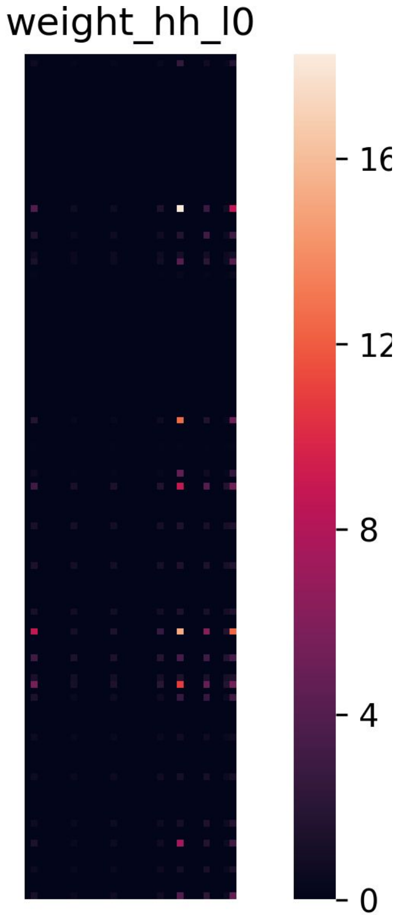
LSTM Baseline



Epoch 0

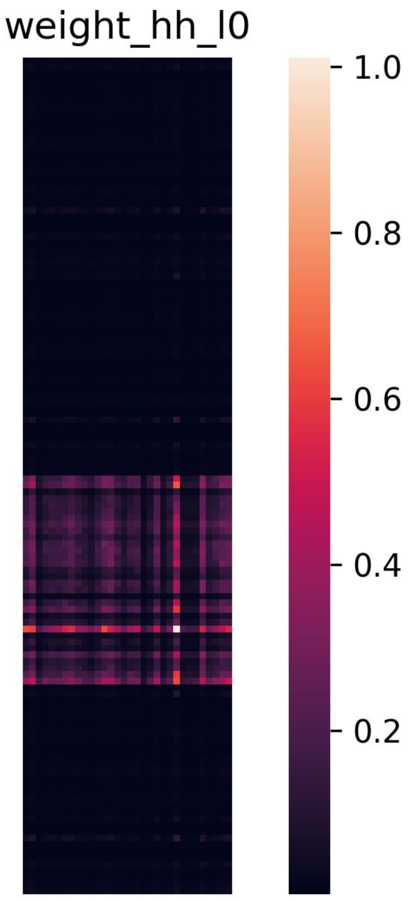


Epoch 50

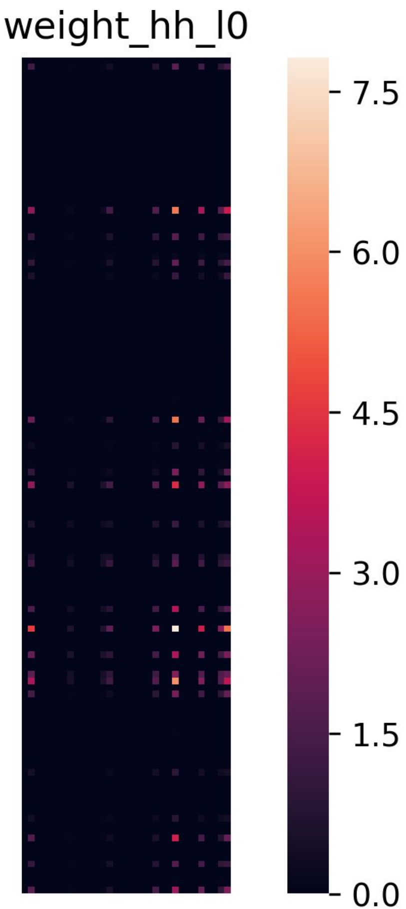


Epoch 100

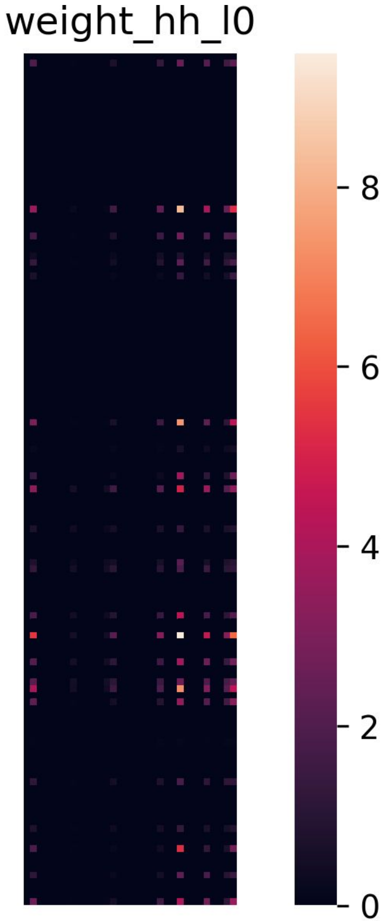
LSTM Block Lasso



Epoch 0

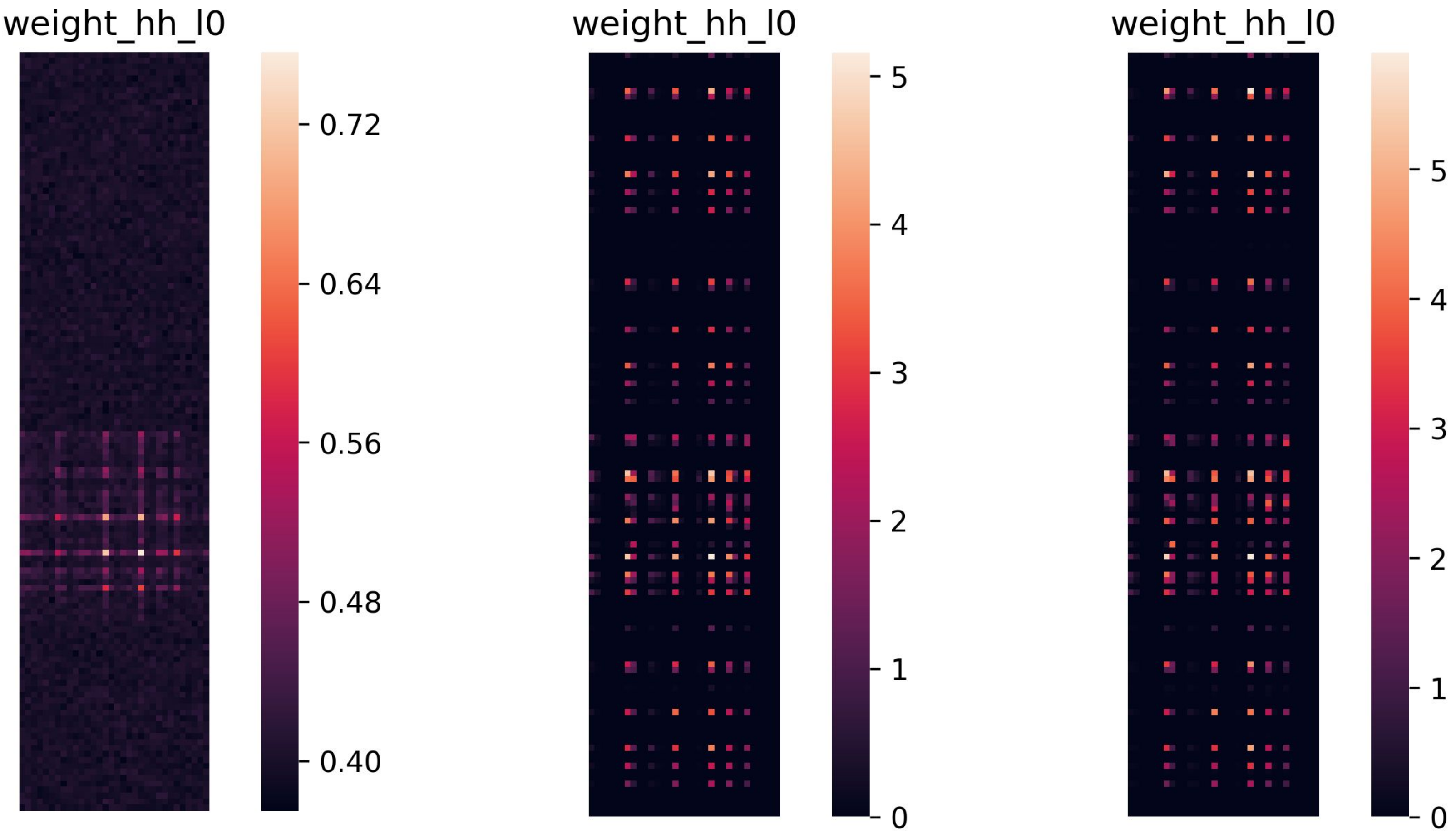


Epoch 50



Epoch 100

LSTM Prune Lasso



Epoch 0

Epoch 50

Epoch 100

LSTM Pre-pruned

Conclusion

- Toy experiment shows that our network is underfitting when block lasso is applied
  - Block lasso serves as additional regularization
  - Future experiments should be done by tuning the regularization and lambda for block sparsity
- Learning true zero sparsity causes performance loss, but it is slightly better than having random fixed sparsity
- We can achieve decent 2x training speed up with gradual pruning to 80% sparsity
- We can achieve 5x inference speed up with 80% sparsity

Thank you!

APPENDIX

Speedup (Gradual Pruning)

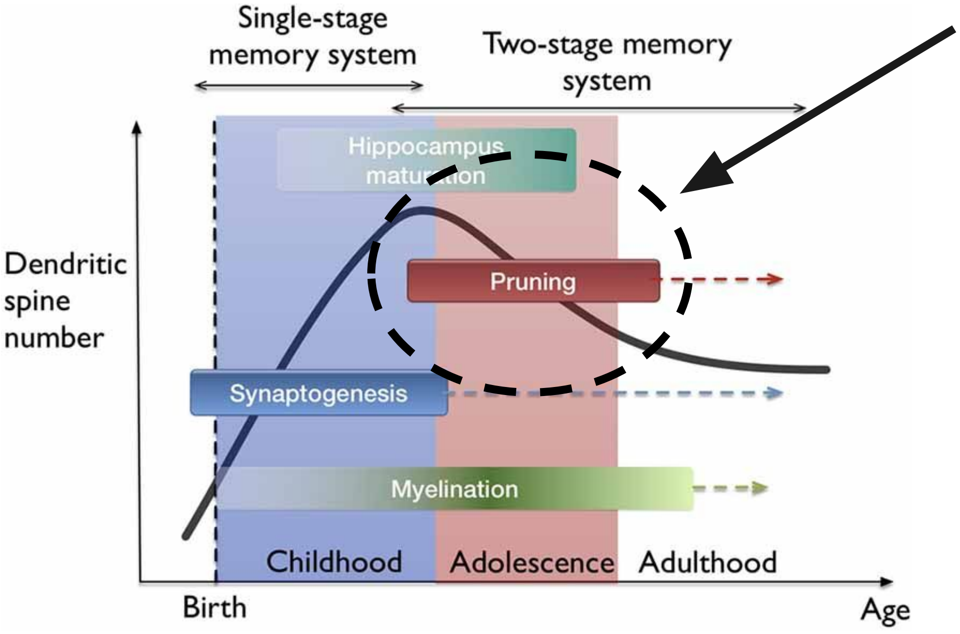
$$\rho_t = \frac{K_t}{n^2}$$

$$K_t = \#\ \text{empty blocks after epoch } t$$

$$\rho_t = f(t) = a(t-1) + b$$

Ozcan, Ahmet S. "Filopodia: a rapid structural plasticity substrate for fast learning." *Frontiers in synaptic neuroscience* 9 (2017): 12.

Brain Development



Speedup (Gradual Pruning)

$$O(dense) = \sum_{t=1}^{T_D} n^2 = n^2 T_D$$

$$O(sparse) = \sum_{t=1}^{T_S} (n^2 - K_t) = n^2 T_S - \sum_{t=1}^{T_S} K_t$$

$$= n^2 (T_S - \sum_{t=1}^{T_S} \rho_t)$$