Model Compression: Weights Pruning for RNNs

Luyu Wang
University of Waterloo & RBC Research Institute
July 17, 2017

Outlines

- Introduction to Model Compression
- RNN Model Compression via Weights Pruning
- Experimental Results
- Conclusion

The Need for Model Compression

- **Success of Deep Neural Network Models**
- **Vast datasets, GPU for training**
- Production environment
- Deep neural nets need lots of computational to make inferences

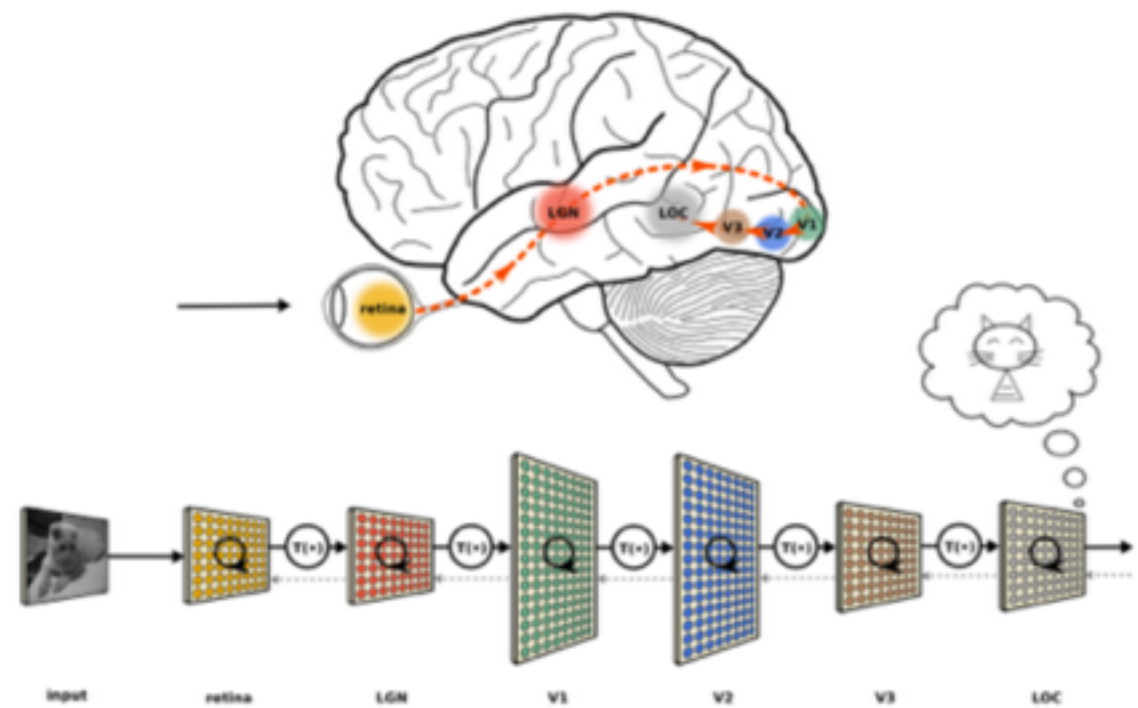


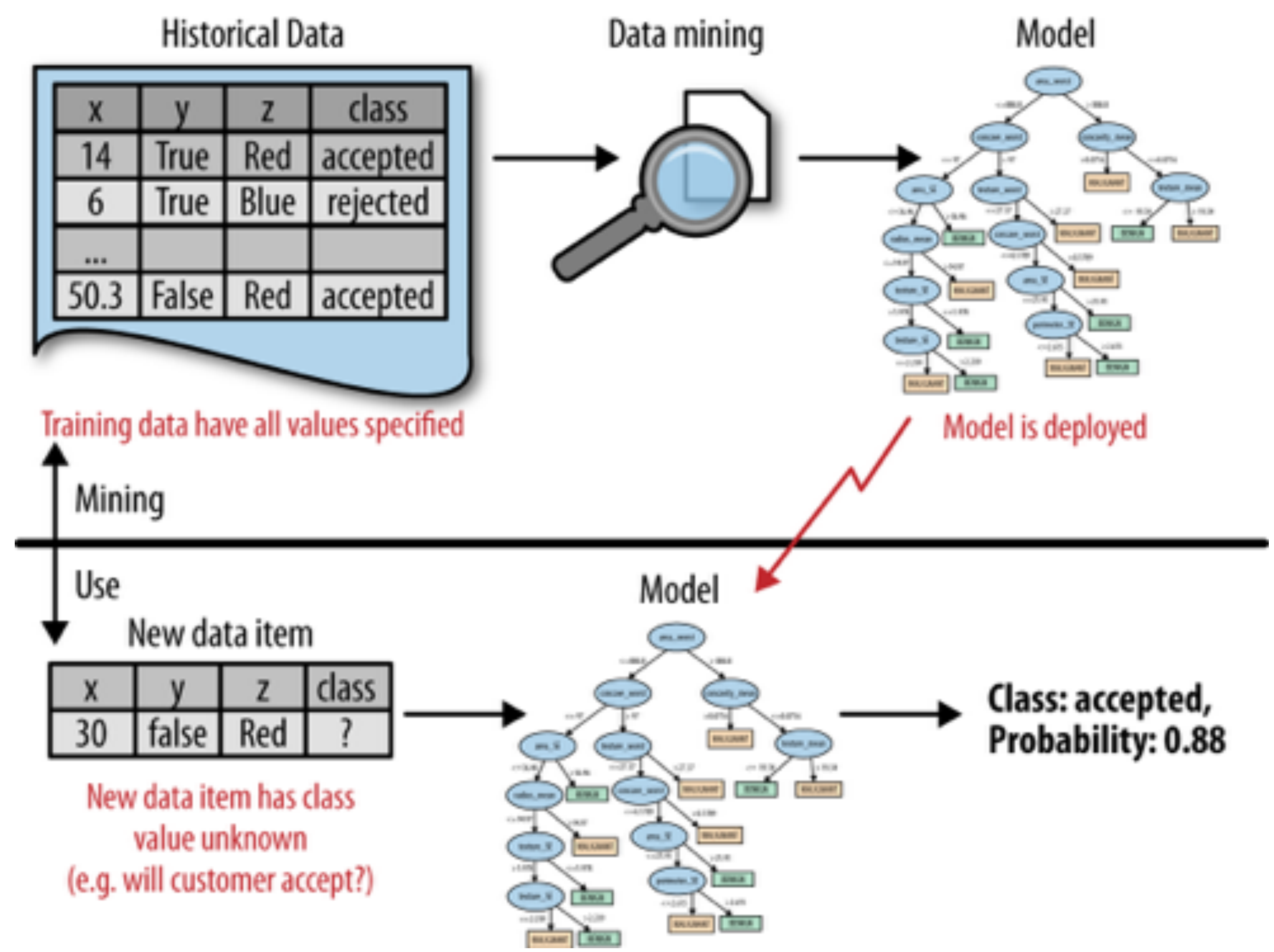
Image Recognition

Speech Recognition

Natural Language Processing

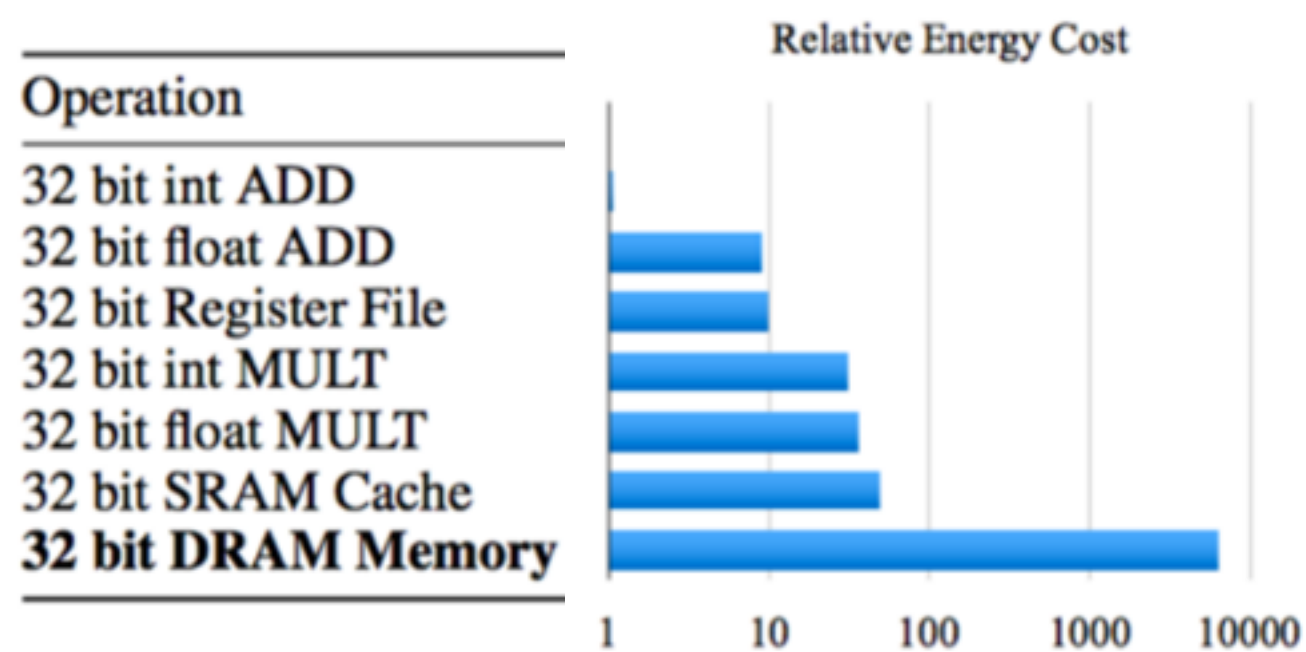
The Need for Model Compression

- Success of Deep Neural Network Models
- Vast datasets, GPU for training
- **Production environment**
- Deep neural nets need lots of computational to make inferences



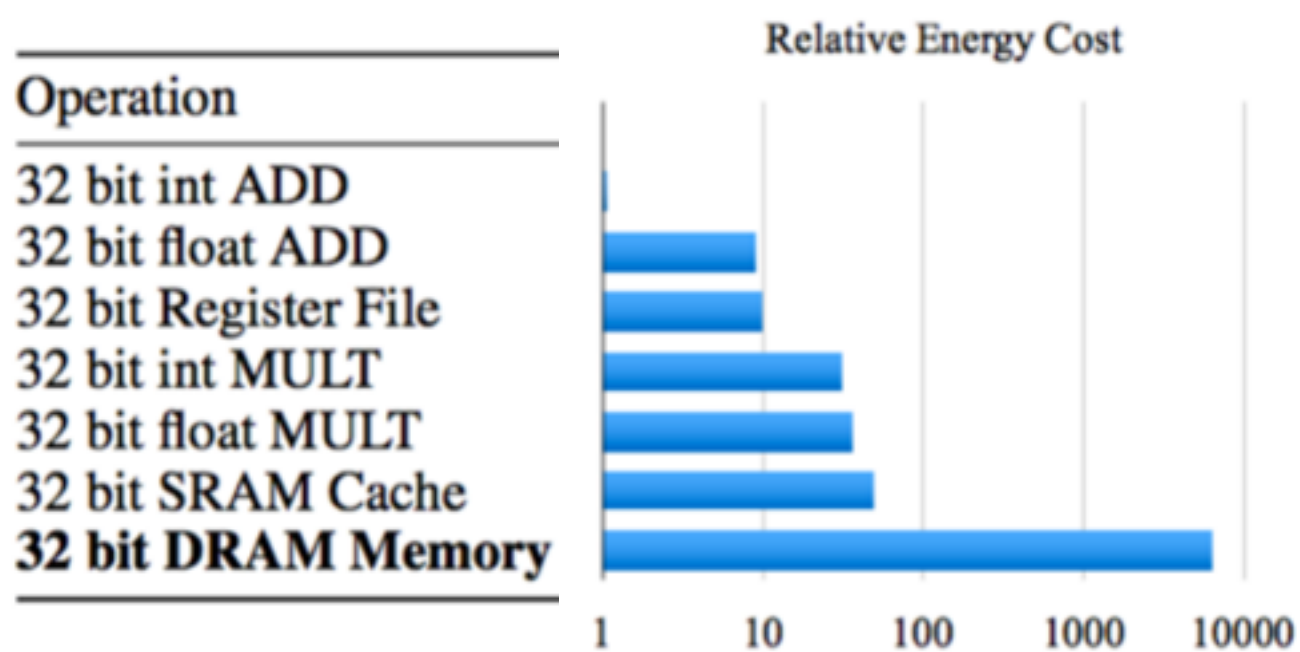
The Need for Model Compression

- Success of Deep Neural Network Models
- Vast datasets, GPU for training
- Production environment
- **Deep neural nets need lots of computational to make inferences**



The Need for Model Compression

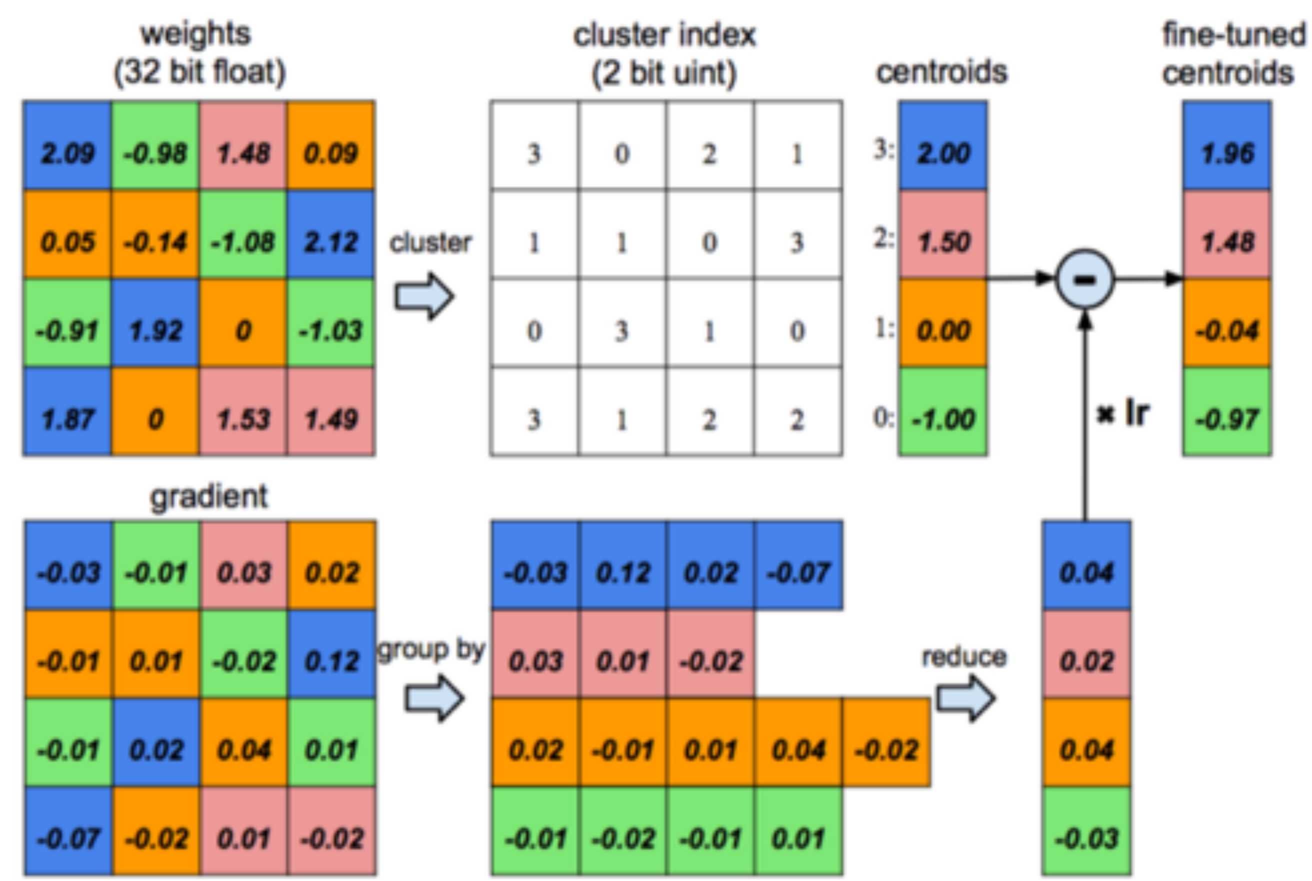
- Success of Deep Neural Network Models
- Vast datasets, GPU for training
- Production environment
- **Deep neural nets need lots of computational to make inferences**



Question: less cost for deployed models?

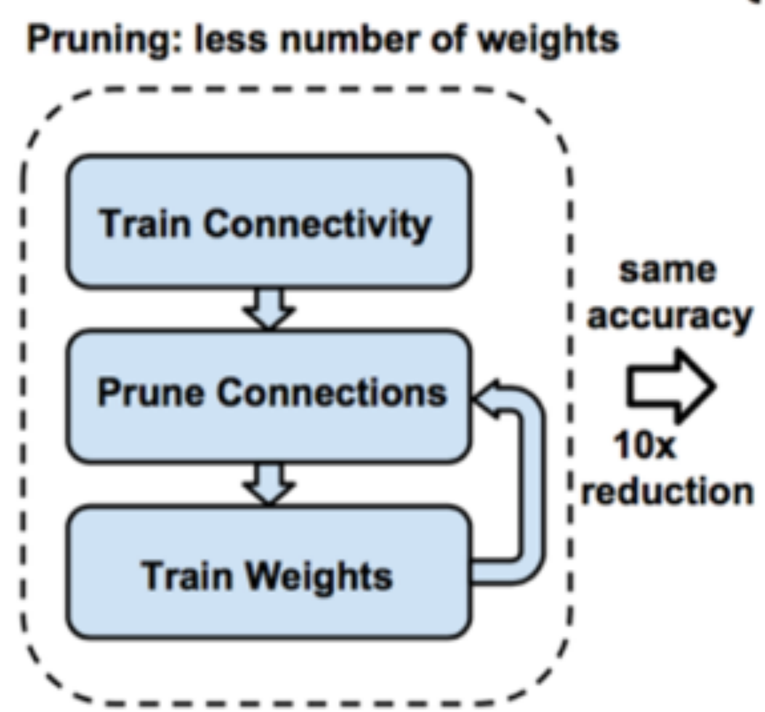
Methods for Model Compression

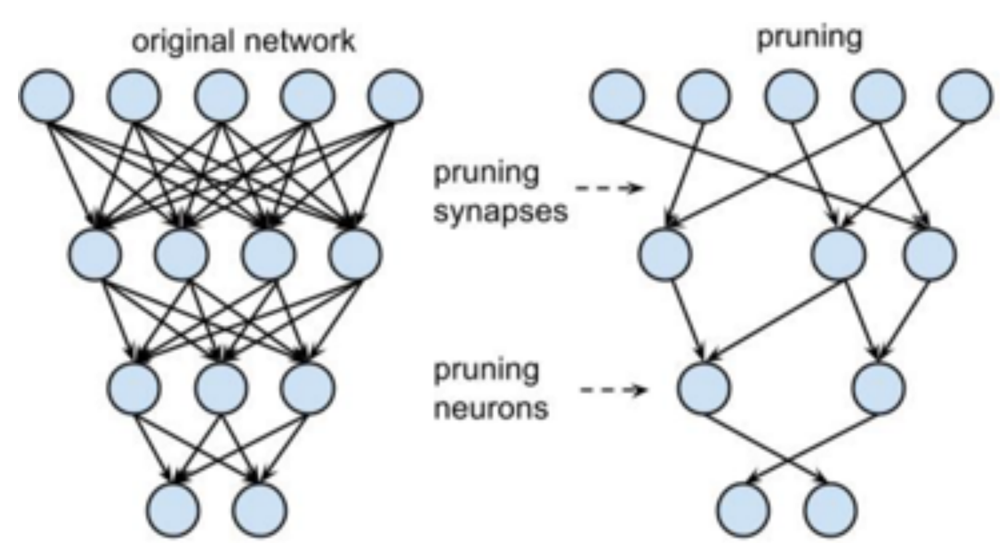
- Weights quantization (or even binarization)
- Model distillation
- Weights pruning



Weights Pruning

- Learn the connectivity via normal network training
- Prune the low-weight connections
- Retrain the sparse network





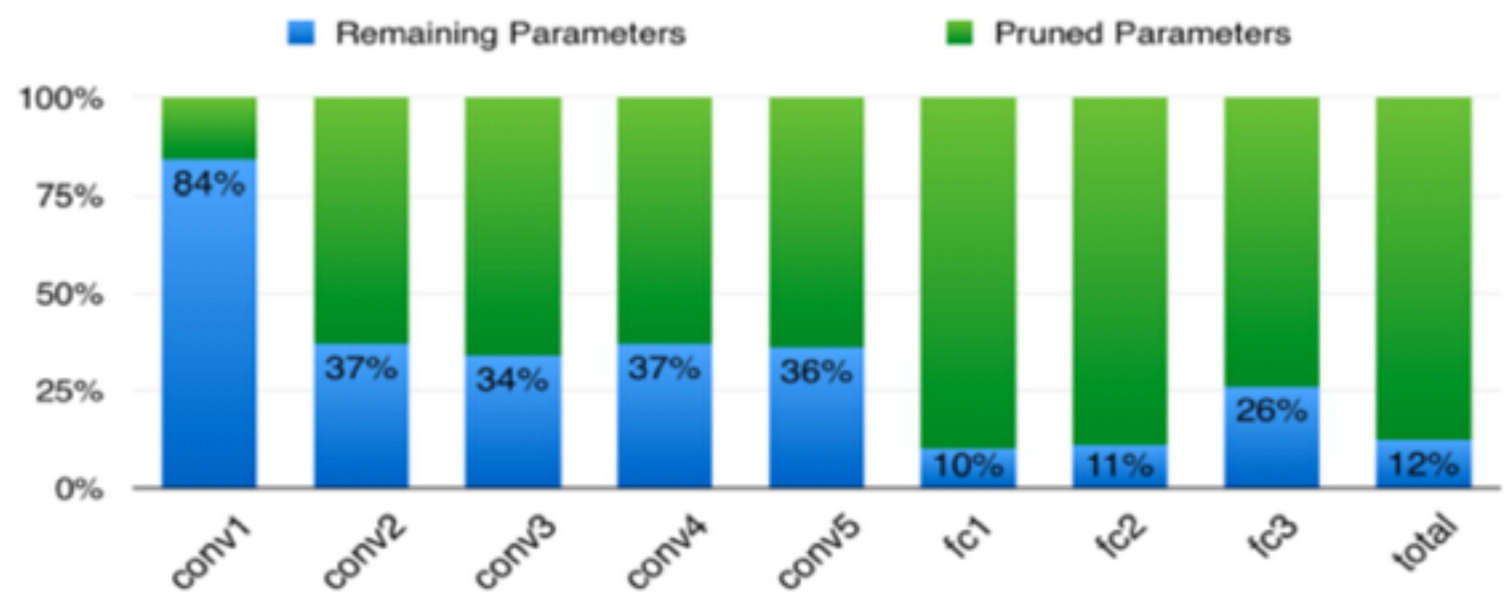
Han, Song, et al. "Learning both weights and connections for efficient neural network." Advances in Neural Information Processing Systems. 2015.

Weights Pruning

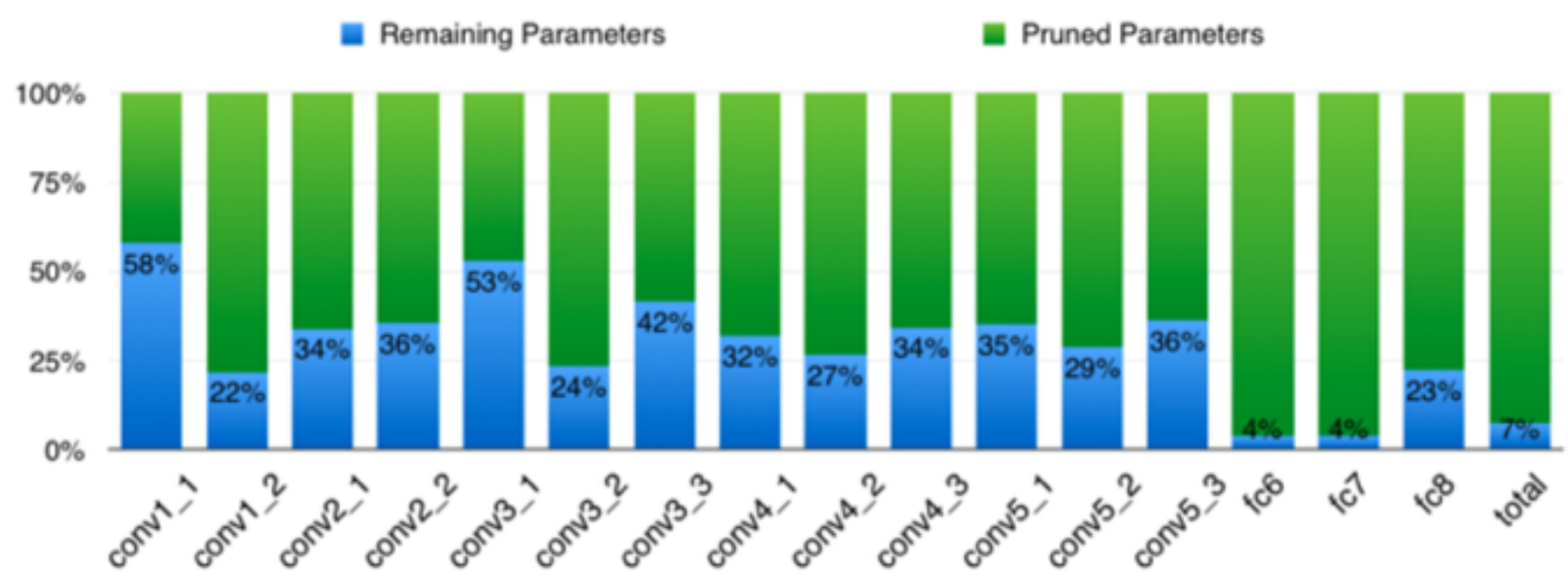
Network	Top-1 Error	Top-5 Error	Parameters	Compression Rate
LeNet-300-100 Ref	1.64%	-	267K	
LeNet-300-100 Pruned	1.59%	-	**22K**	**12**×
LeNet-5 Ref	0.80%	-	431K	
LeNet-5 Pruned	0.77%	-	**36K**	**12**×
AlexNet Ref	42.78%	19.73%	61M	
AlexNet Pruned	42.77%	19.67%	**6.7M**	**9**×
VGG16 Ref	31.50%	11.32%	138M	
VGG16 Pruned	31.34%	10.88%	**10.3M**	**13**×

Weights Pruning

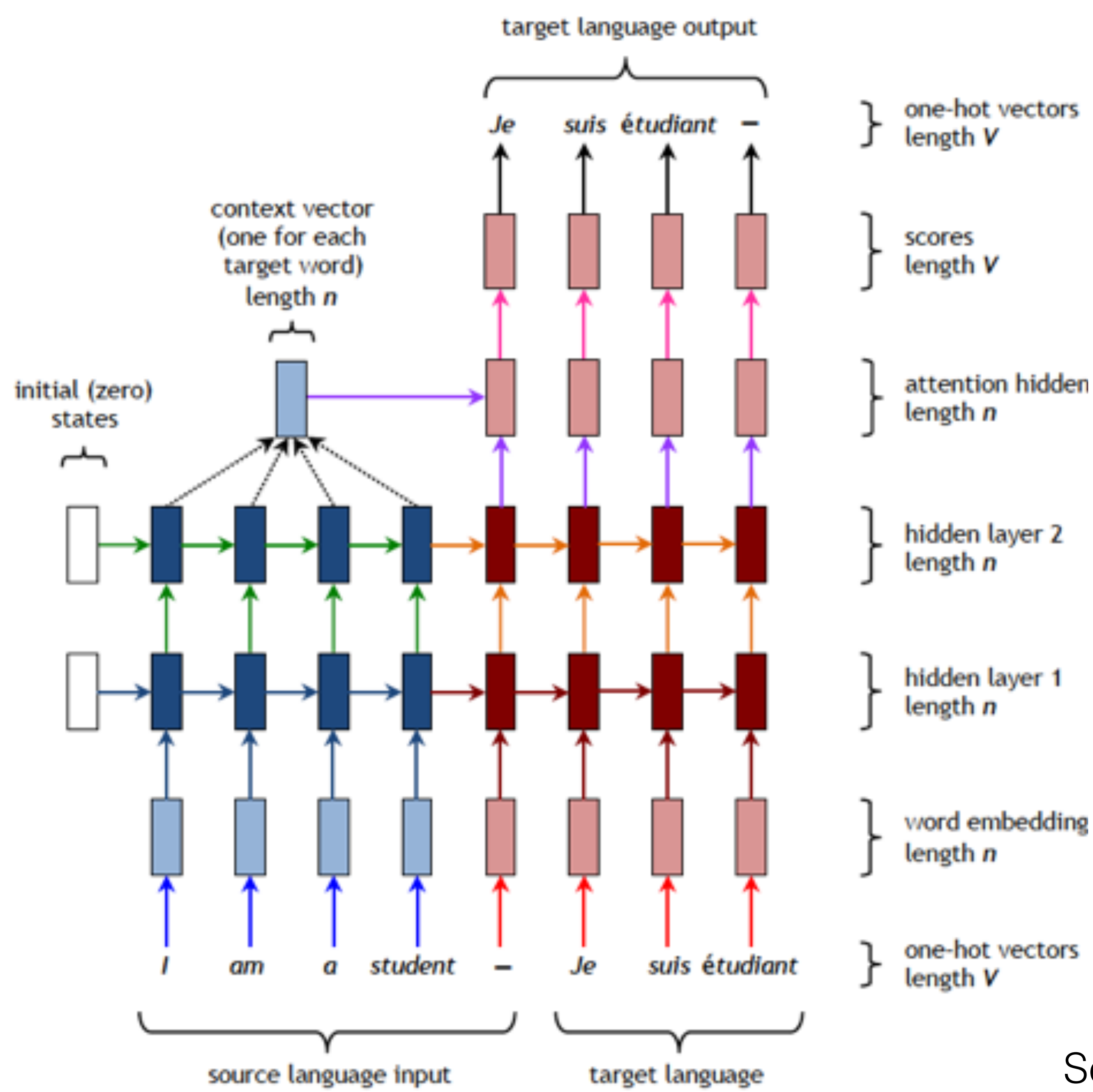
AlexNet

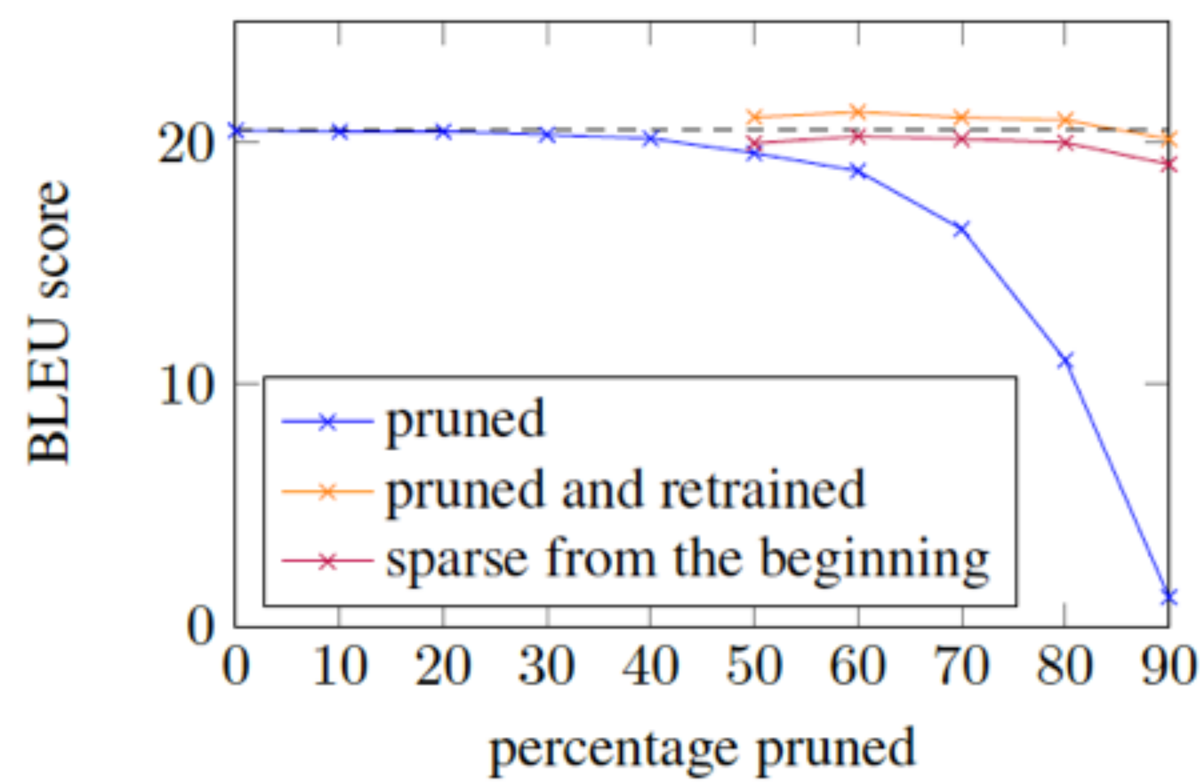


VGG



Neural Machine Translation Models via Pruning





See, Abigail, Minh-Thang Luong, and Christopher D. Manning. "Compression of Neural Machine Translation Models via Pruning." CoNLL 2016 (2016): 291.

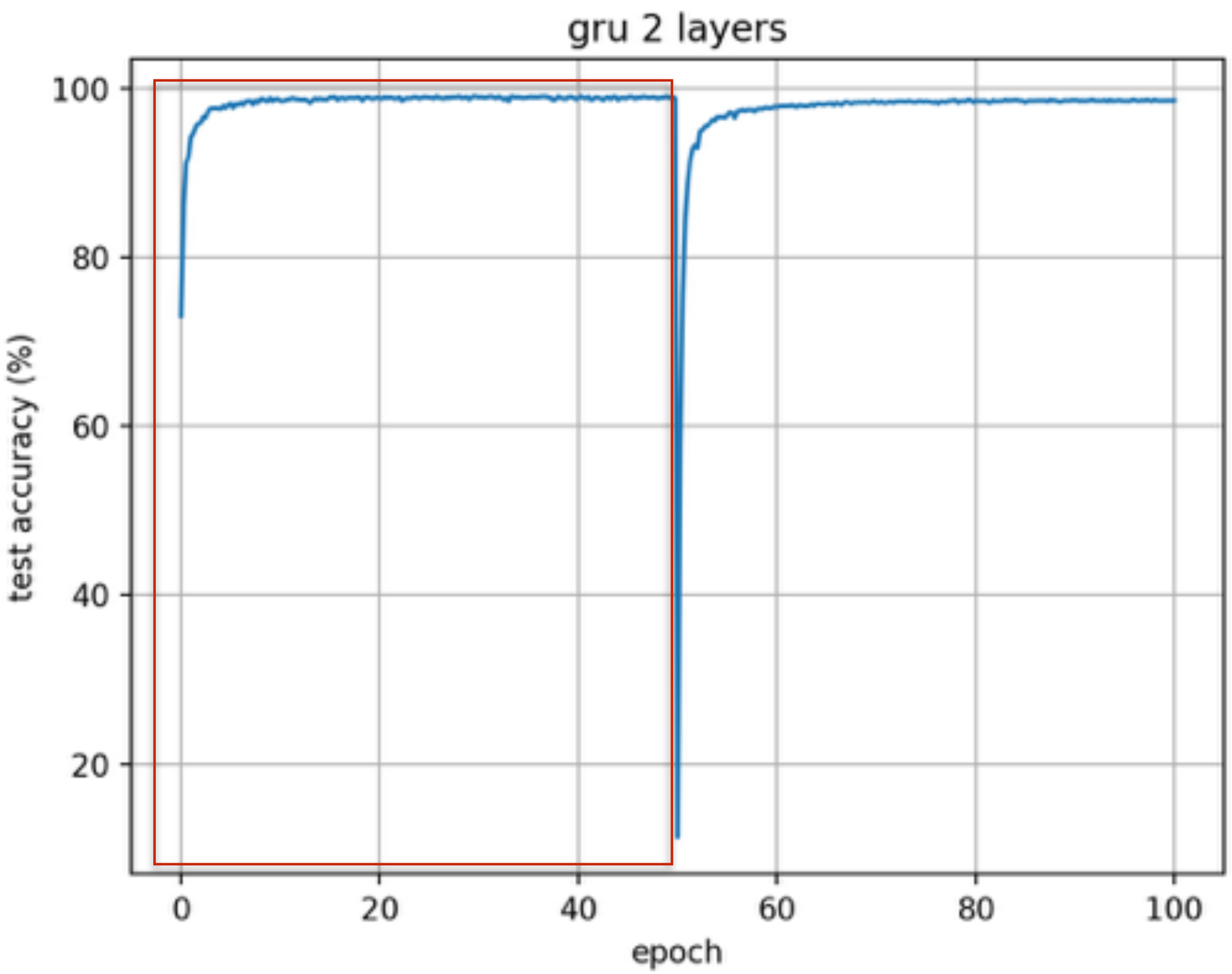
Pruning RNNs

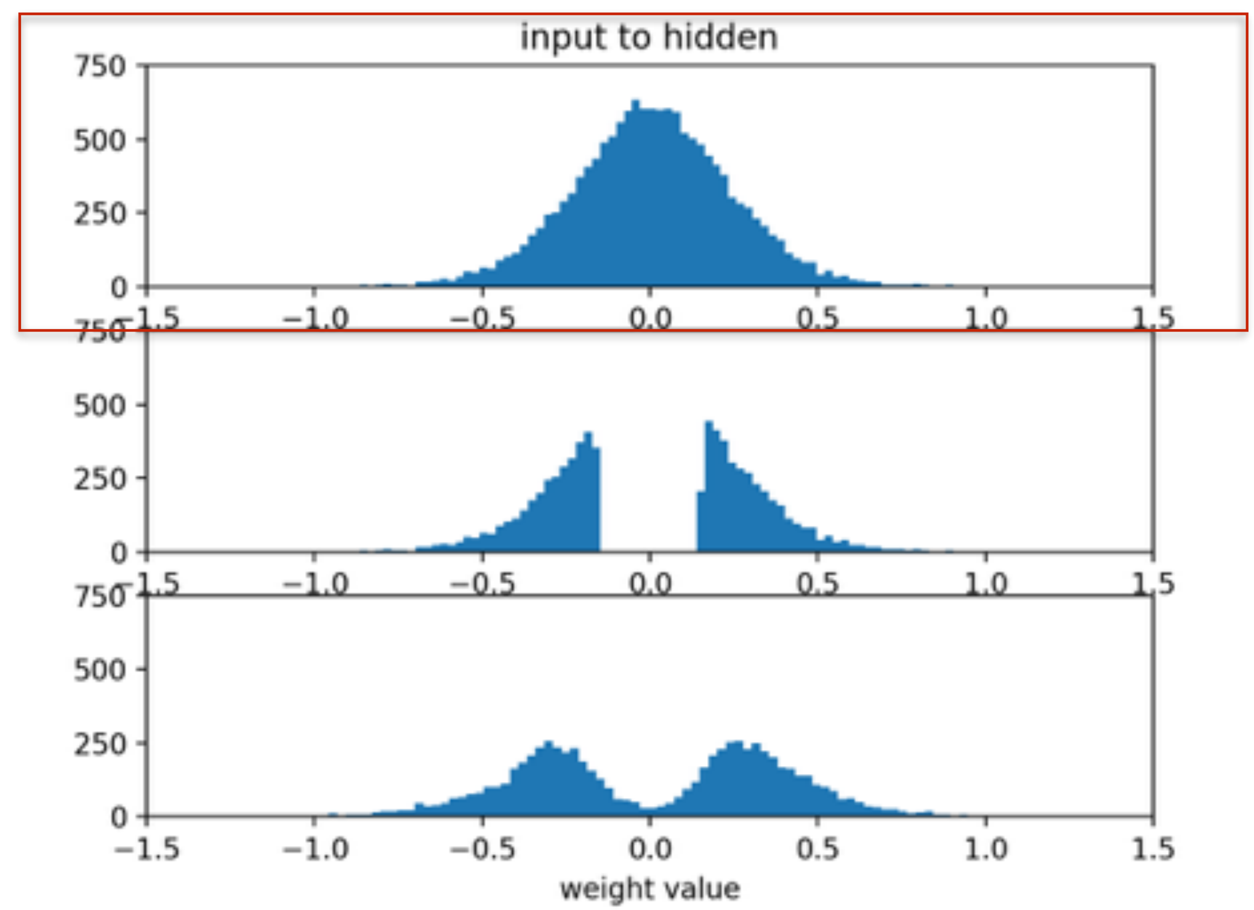
- Vanilla RNN
- LSTM
- GRU

- MNIST Dataset 28x28
- Hidden states = 128
- RMSprop with fixed learning rate 0.001
- PyTorch on AWS GPU server

Pruning RNNs

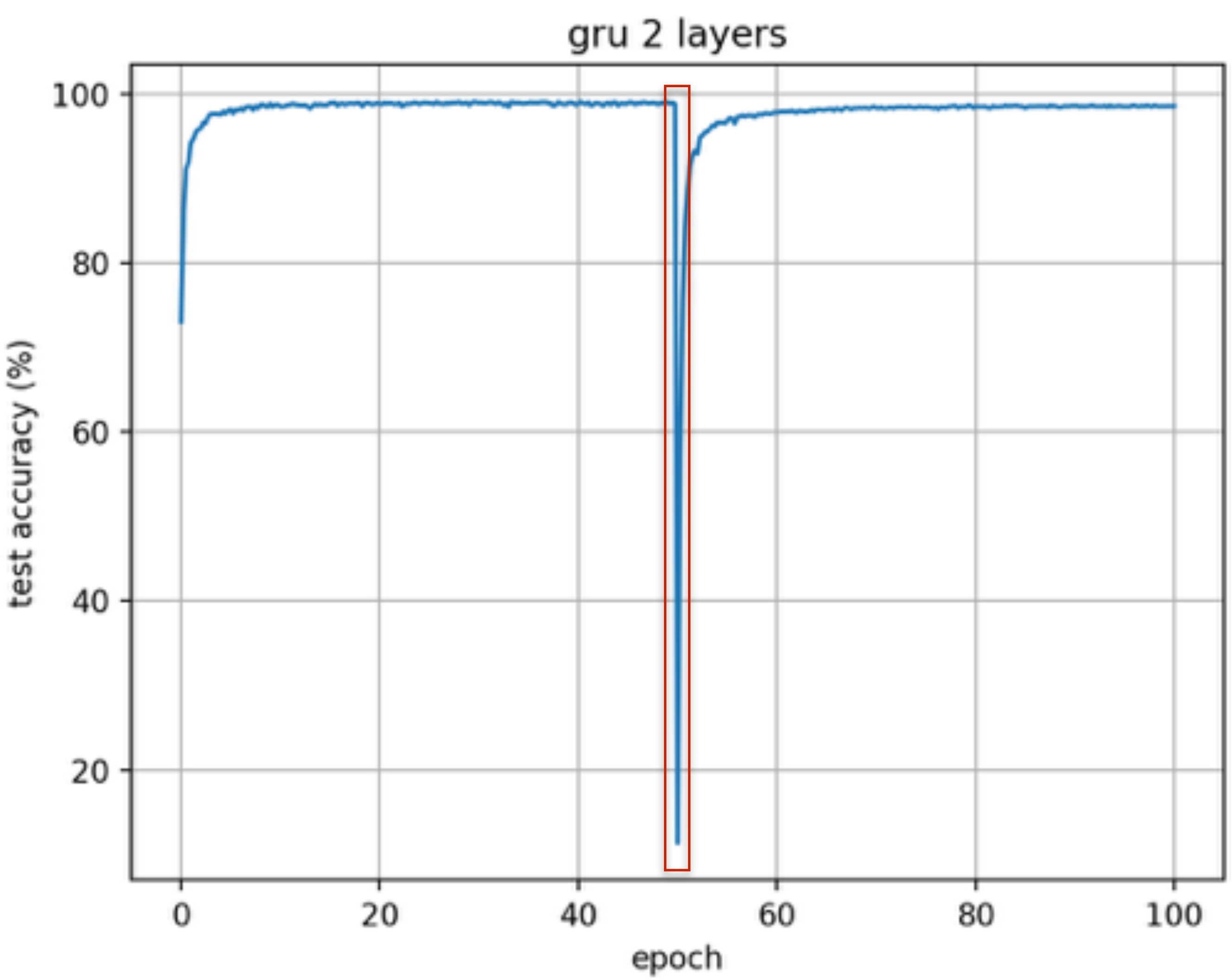
Pre-train

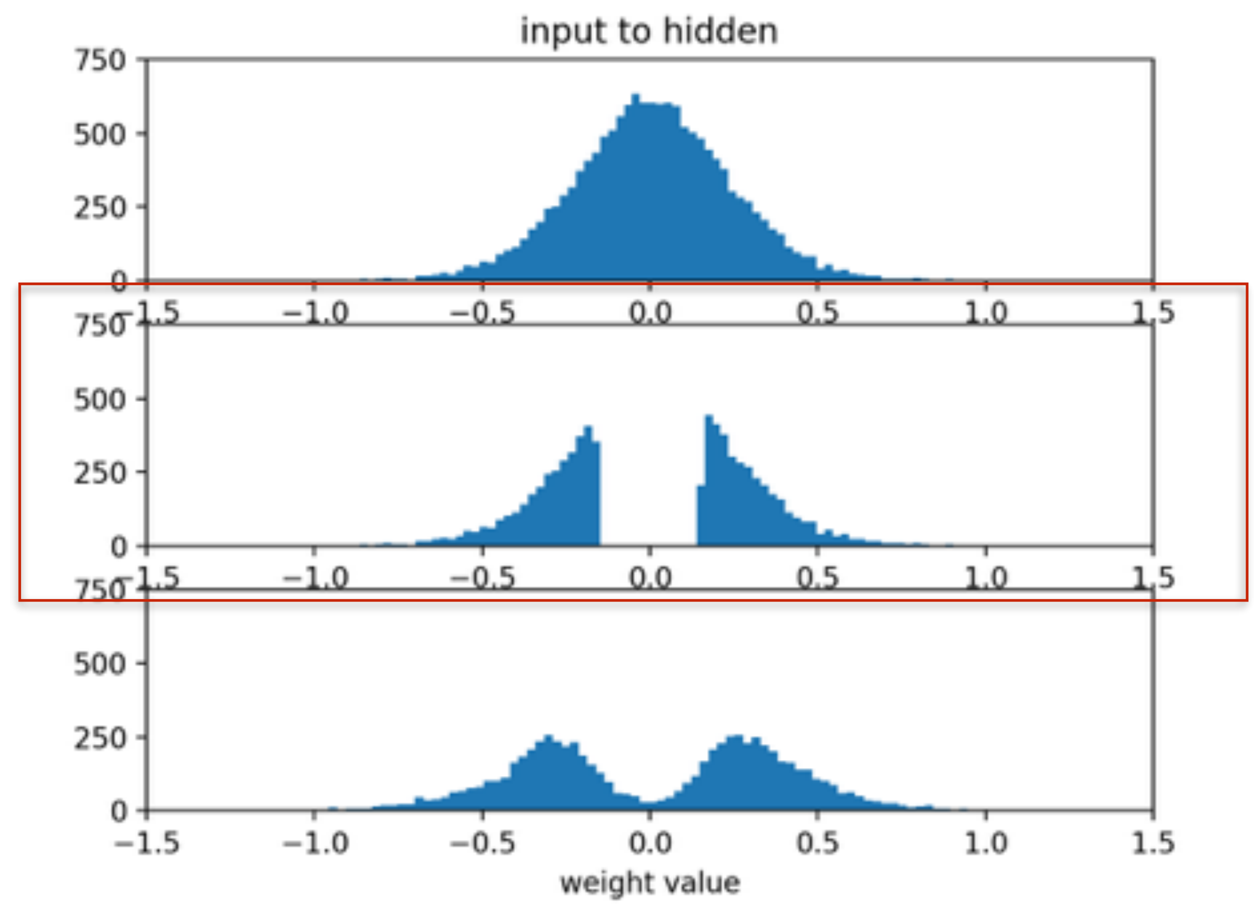




Pruning RNNs

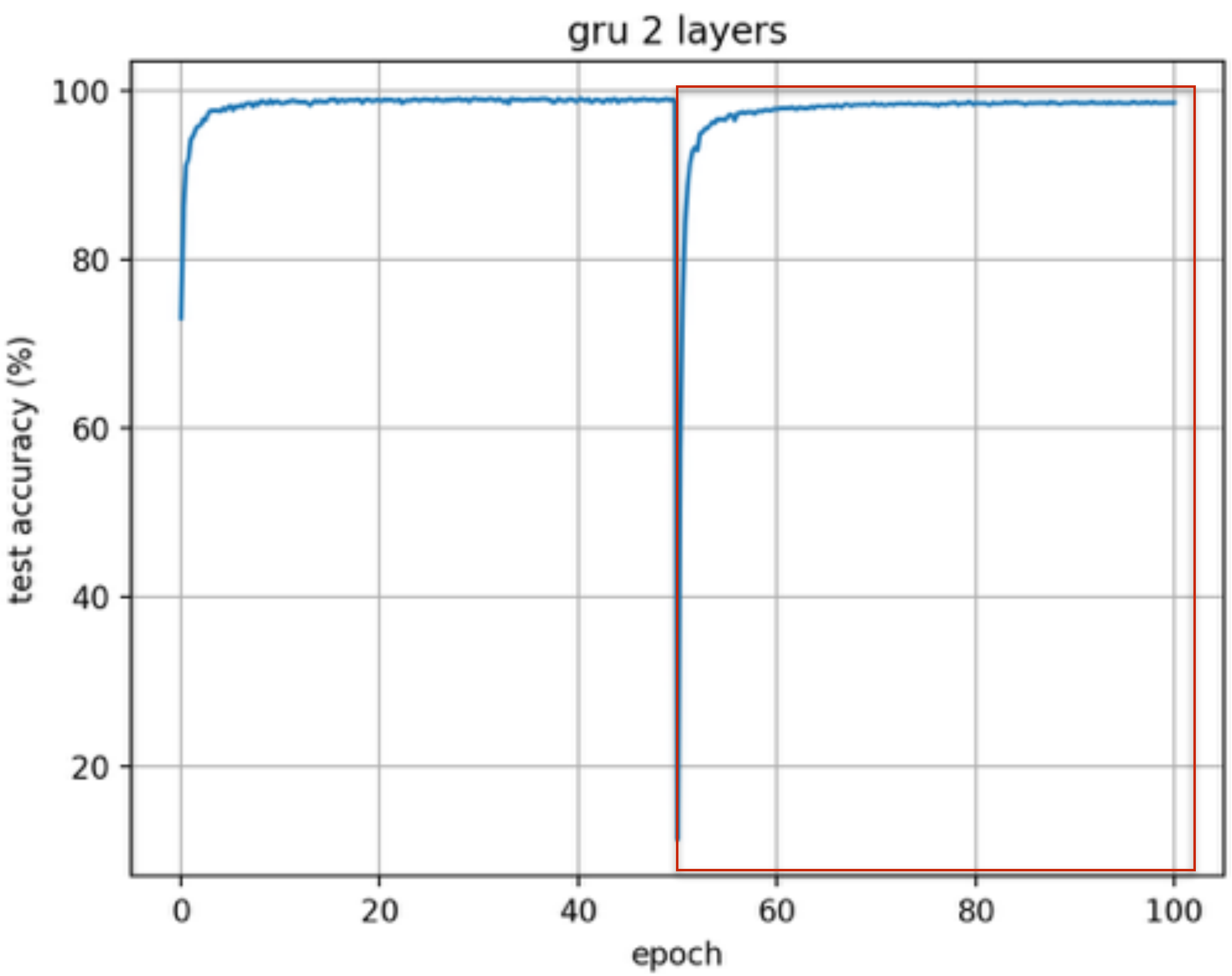
Pruning

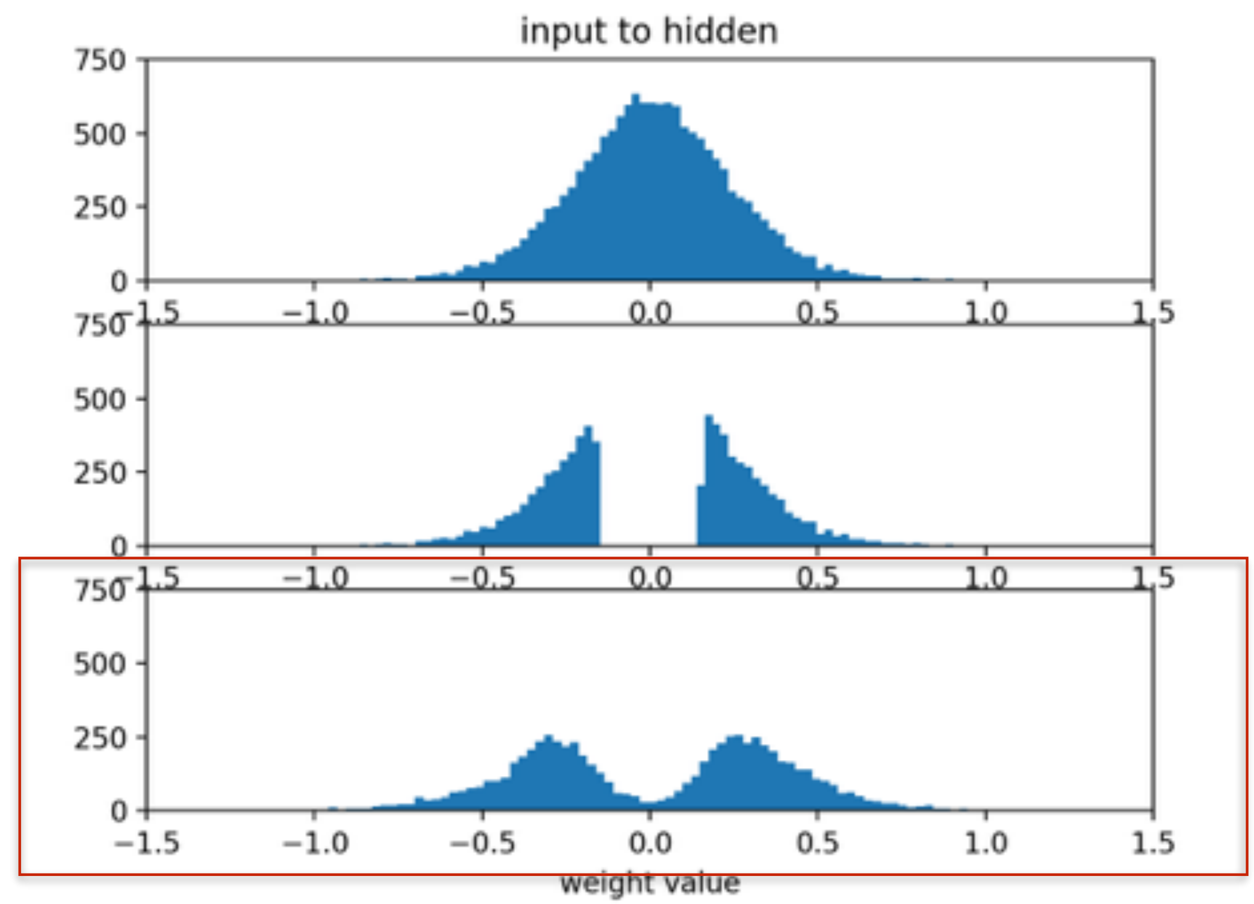




Pruning RNNs

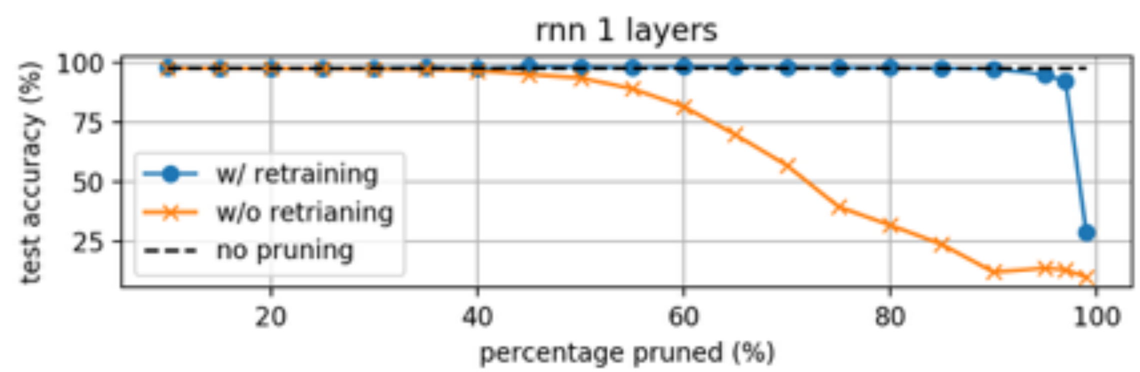
Re-train

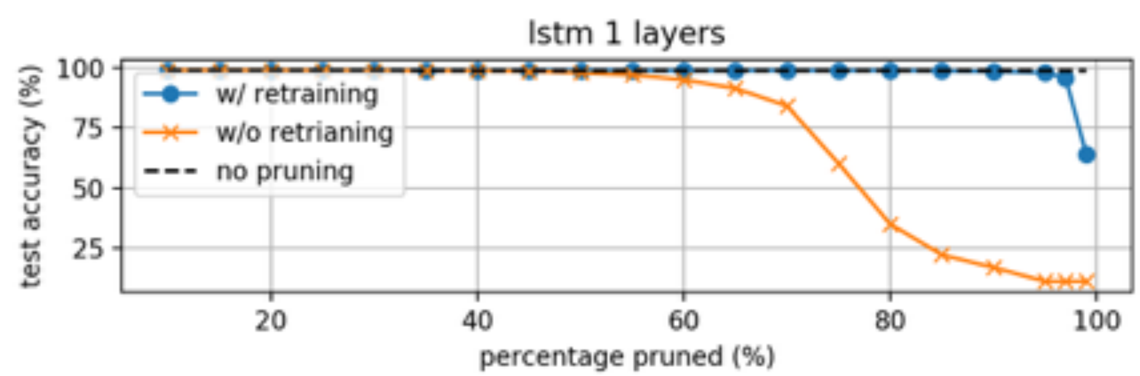


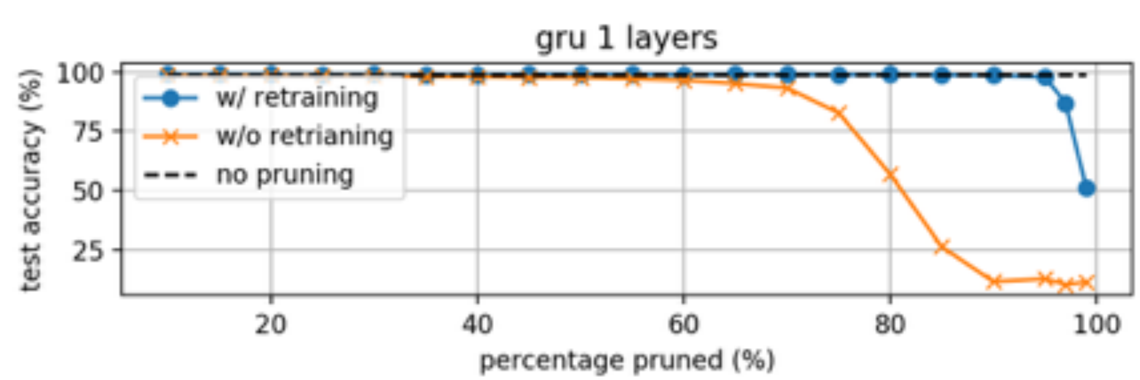


Pruning RNNs

- Retraining closes the gap
- 95% pruned - accuracy loss smaller than 1%
- LSTM and GRU more resilient to pruning - additional redundancy in gates

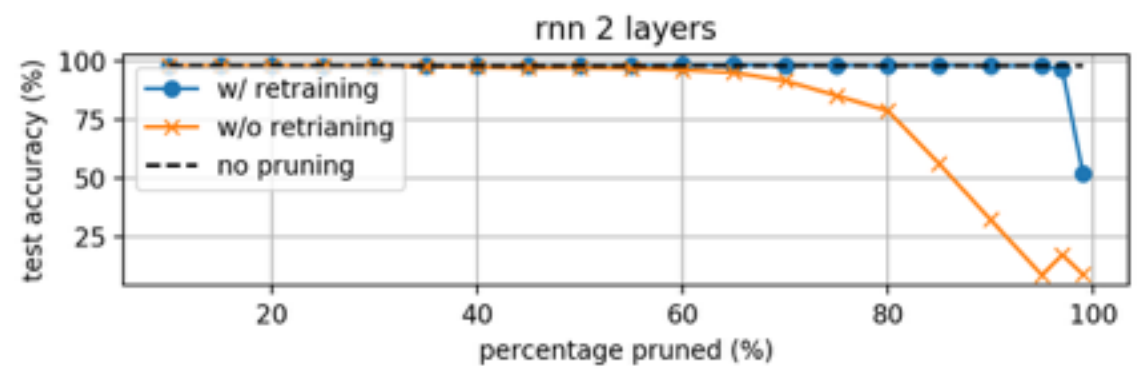


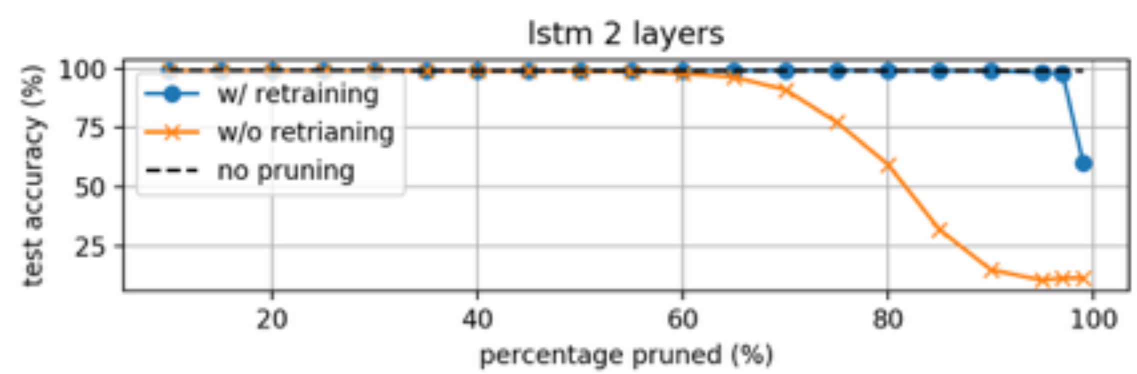


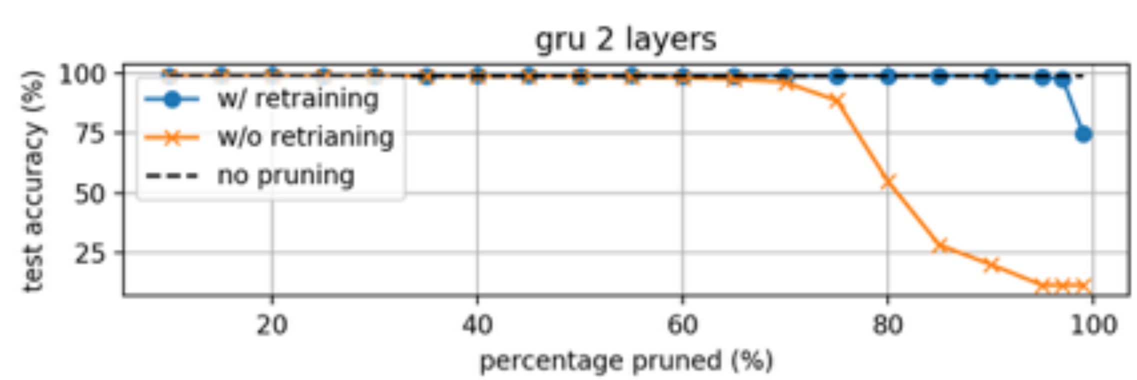


Pruning RNNs

- Retraining closes the gap
- 95% pruned - accuracy loss smaller than 1%
- LSTM and GRU more resilient to pruning - additional redundancy in gates

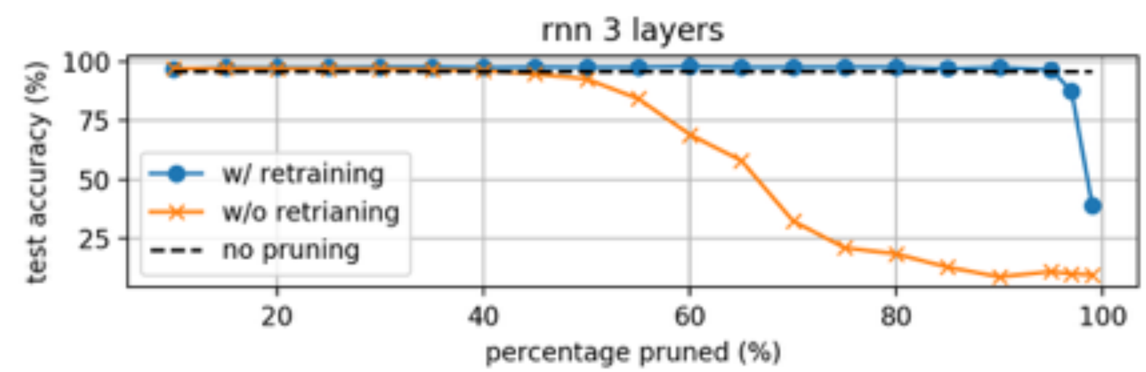


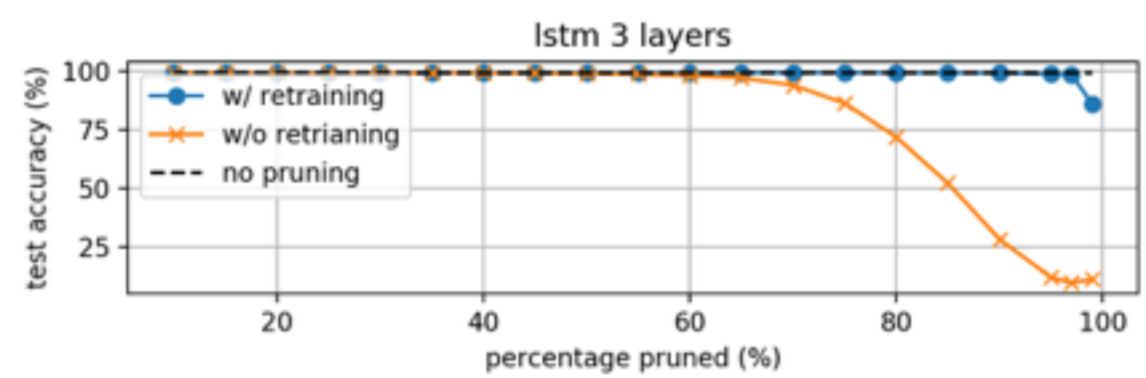


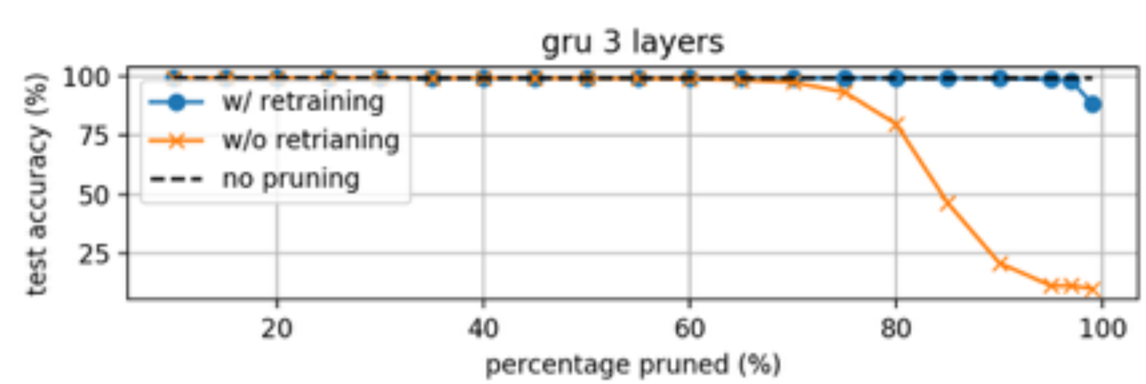


Pruning RNNs

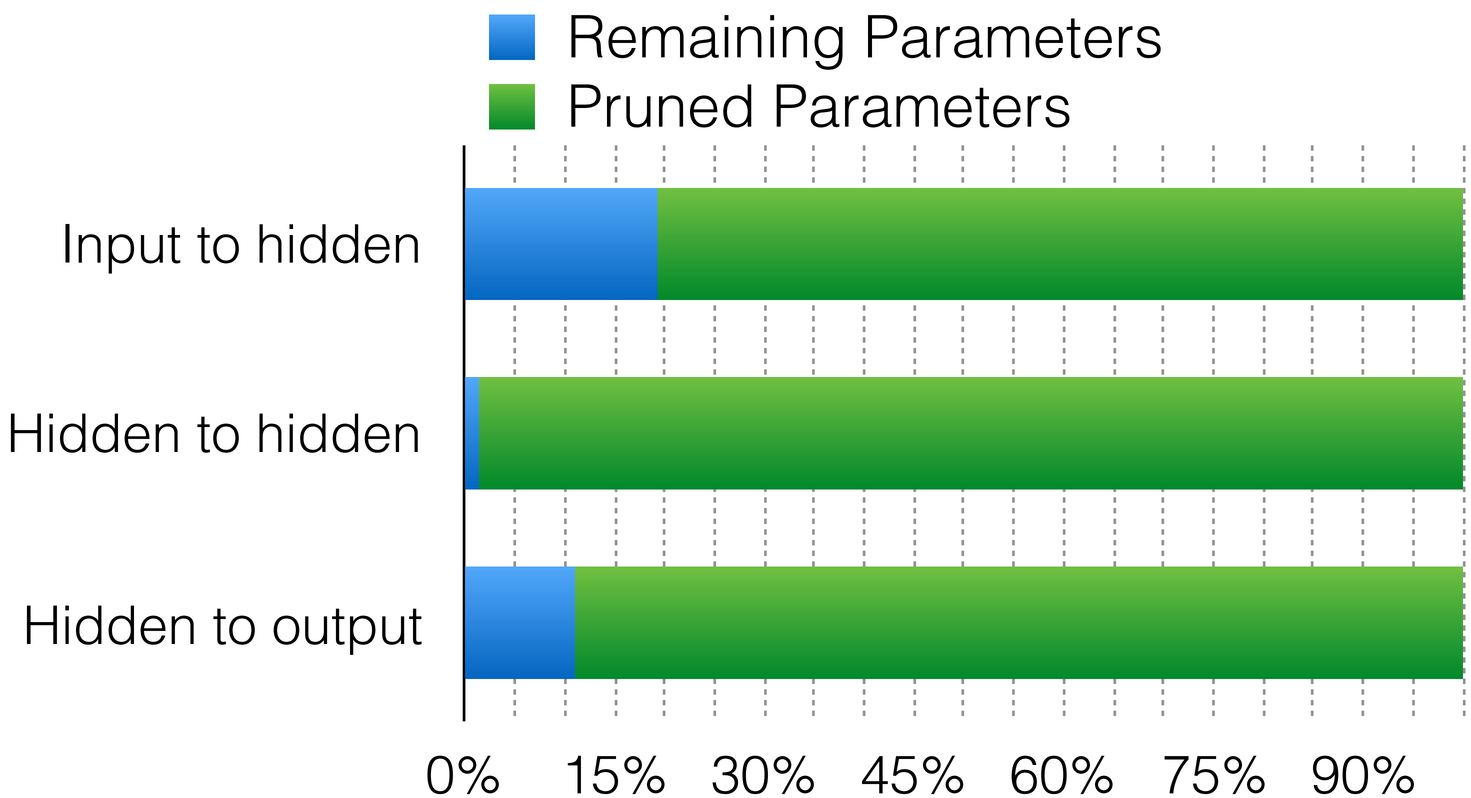
- Retraining closes the gap
- 95% pruned - accuracy loss smaller than 1%
- LSTM and GRU more resilient to pruning - additional redundancy in gates
- More layers - more can be pruned for LSTM & GRU (RNN?)







1 Layer RNN - 95% Pruned



Hidden to hidden more redundancy

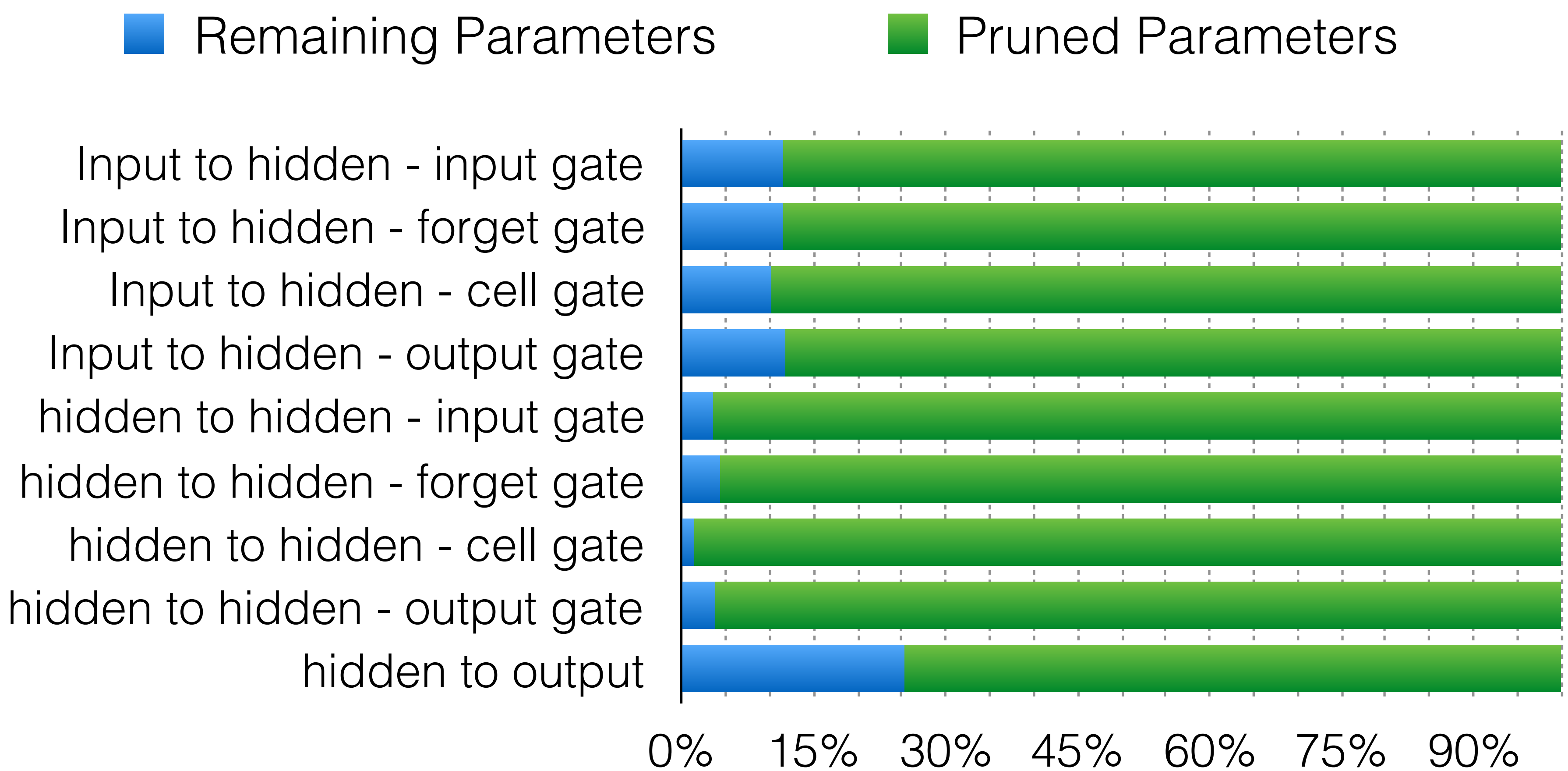
$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}$$

$$\boldsymbol{h}^{(t)} = \psi(\boldsymbol{a}^{(t)})$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}$$

$$\hat{\boldsymbol{y}}^{(t)} = \phi(\boldsymbol{o}^{(t)})$$

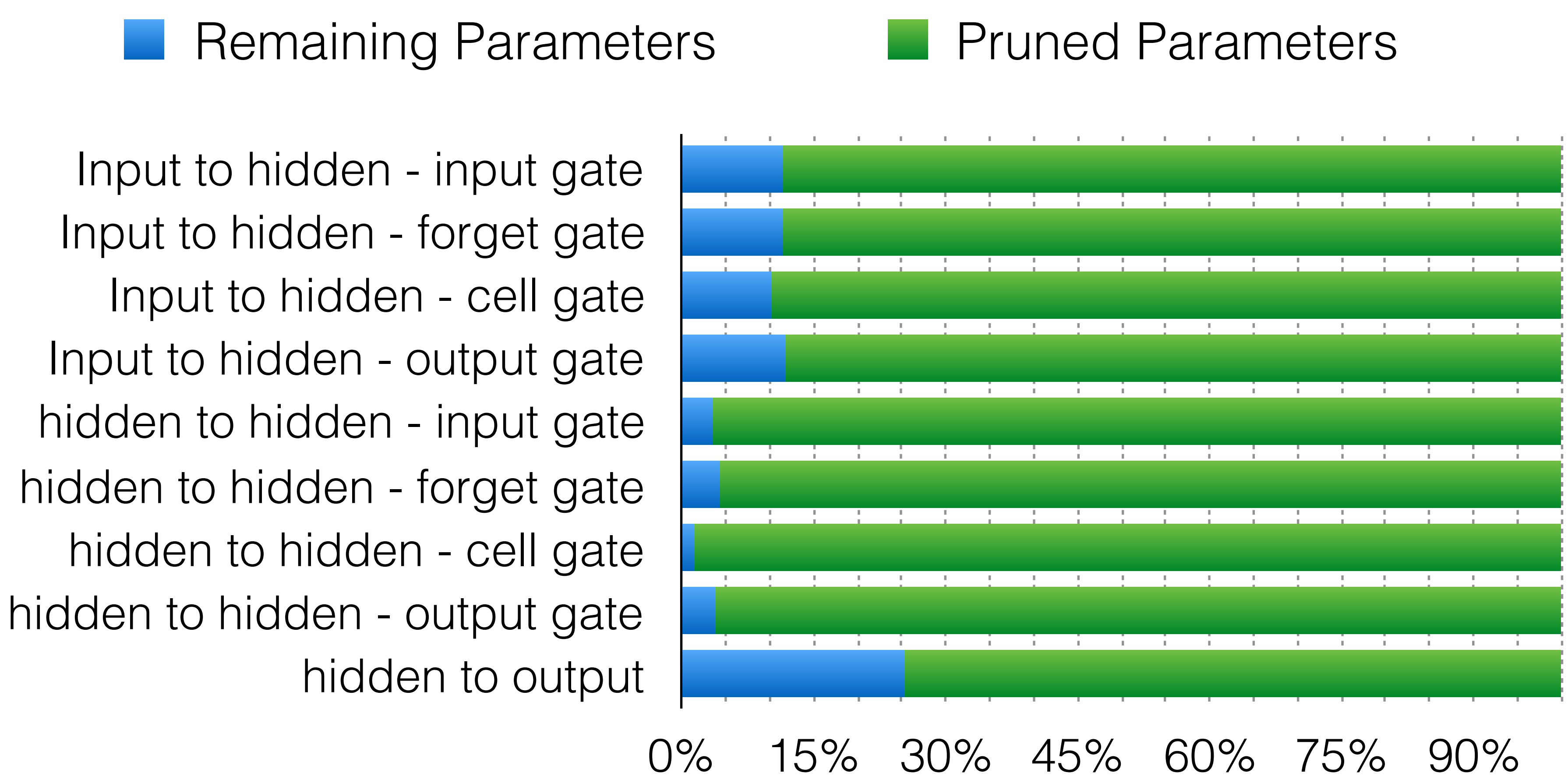
1 Layer LSTM - 95% Pruned Input Gate



Hidden to hidden more redundancy

$$
\begin{aligned}
\boldsymbol{i}^{(t)} &= \sigma\left(\boldsymbol{b}^{i} + \boldsymbol{W}^{i}\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right) \\
\boldsymbol{g}^{(t)} &= \tanh\left(\boldsymbol{b}^{g} + \boldsymbol{W}^{g}\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right) \\
\boldsymbol{f}^{(t)} &= \sigma\left(\boldsymbol{b}^{f} + \boldsymbol{W}^{f}\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right) \\
\boldsymbol{c}^{(t)} &= \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{g}^{(t)} \\
\boldsymbol{o}^{(t)} &= \sigma\left(\boldsymbol{b}^{o} + \boldsymbol{W}^{o}\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right) \\
\boldsymbol{h}^{(t)} &= \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})
\end{aligned}
$$

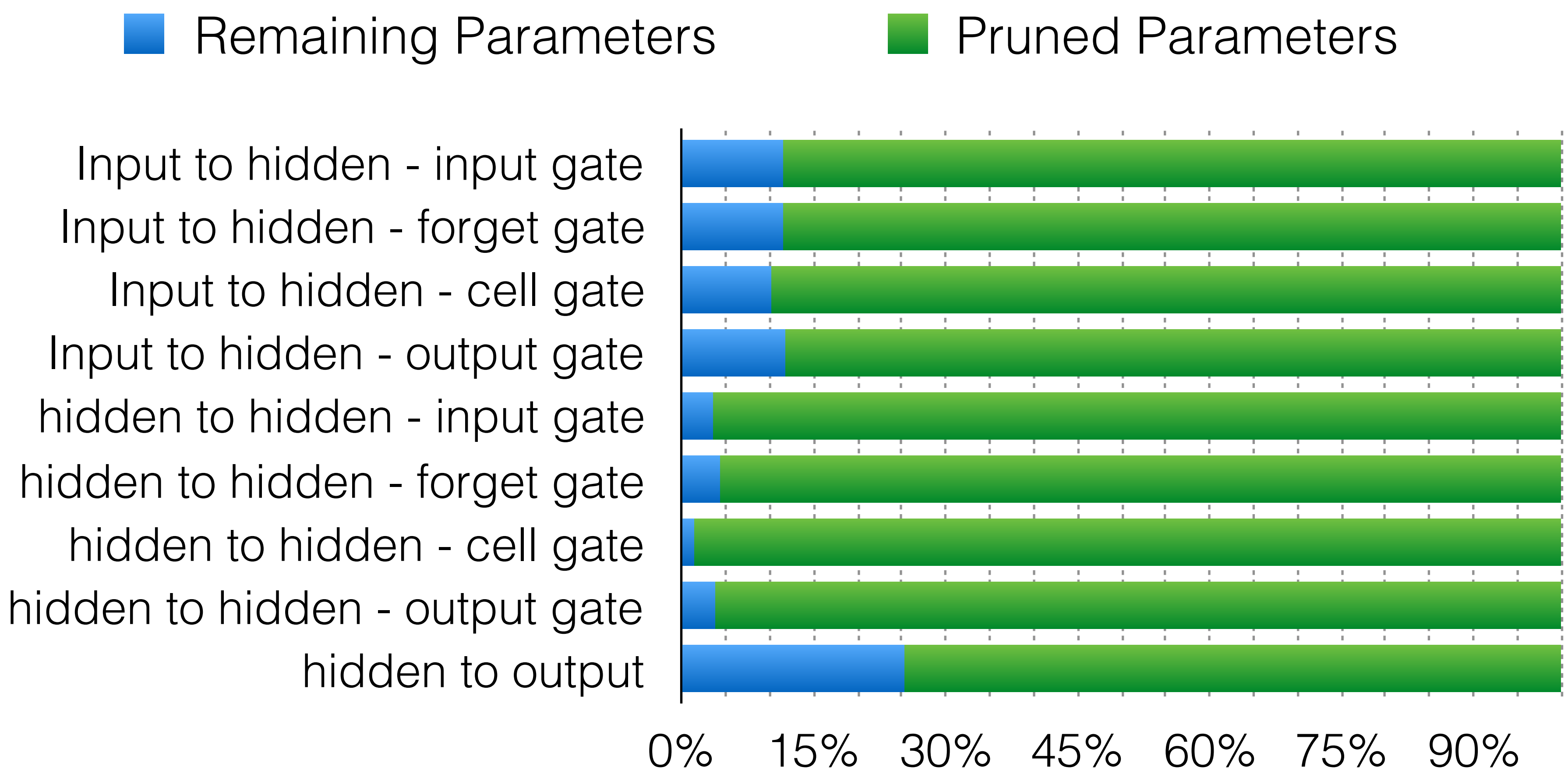
1 Layer LSTM - 95% Pruned Cell Gate



Hidden to hidden more redundancy

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{b}^i + \boldsymbol{W}^i\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$
$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$
$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{b}^f + \boldsymbol{W}^f\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$
$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{g}^{(t)}$$
$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{b}^o + \boldsymbol{W}^o\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$
$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

1 Layer LSTM - 95% Pruned Forget Gate



Hidden to hidden more redundancy

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{b}^i + \boldsymbol{W}^i\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

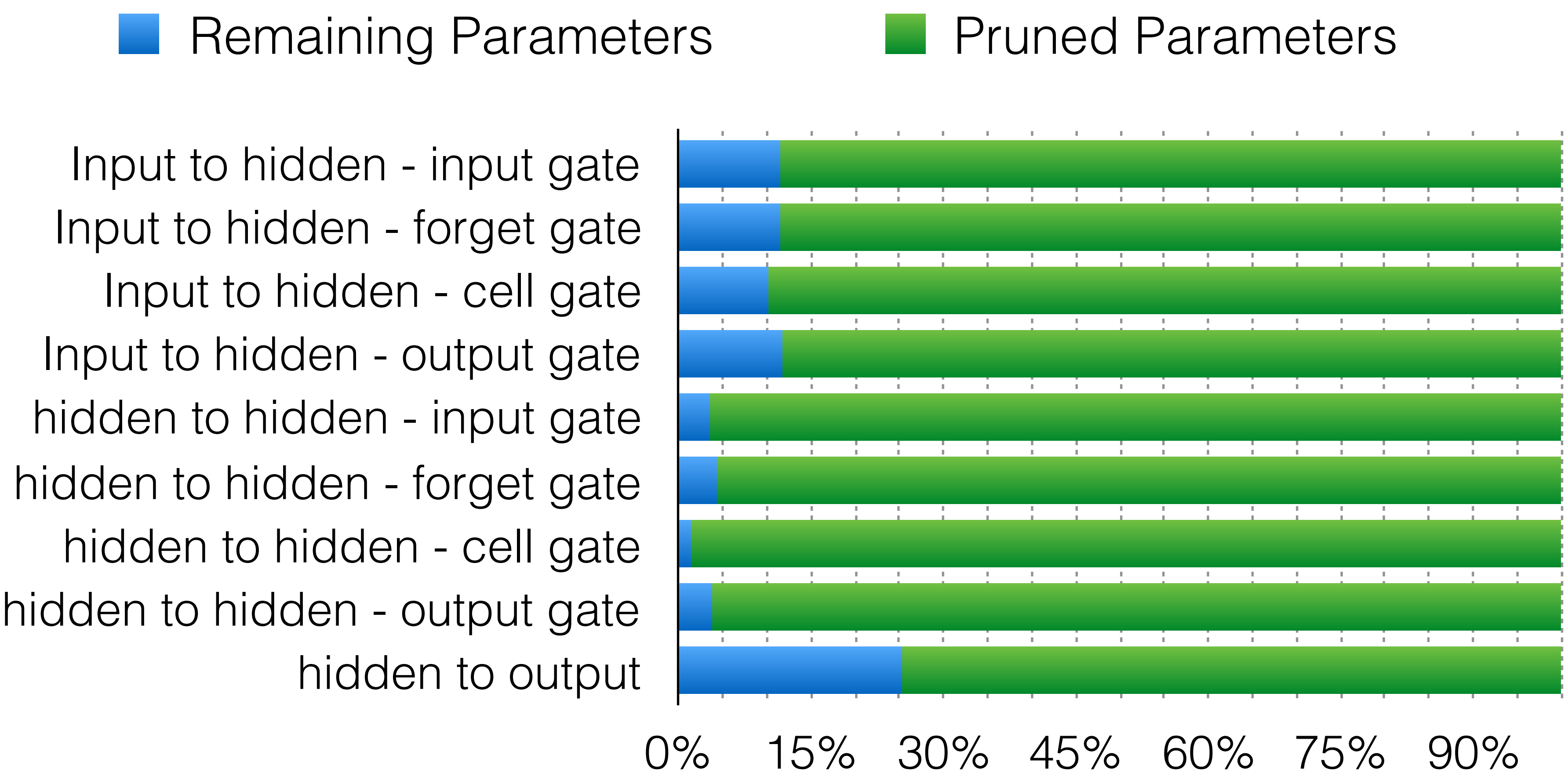
$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{b}^f + \boldsymbol{W}^f\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{g}^{(t)}$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{b}^o + \boldsymbol{W}^o\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

1 Layer LSTM - 95% Pruned Output Gate



Hidden to hidden more redundancy

$$\boldsymbol{i}^{(t)} = \sigma\left(\boldsymbol{b}^i + \boldsymbol{W}^i\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

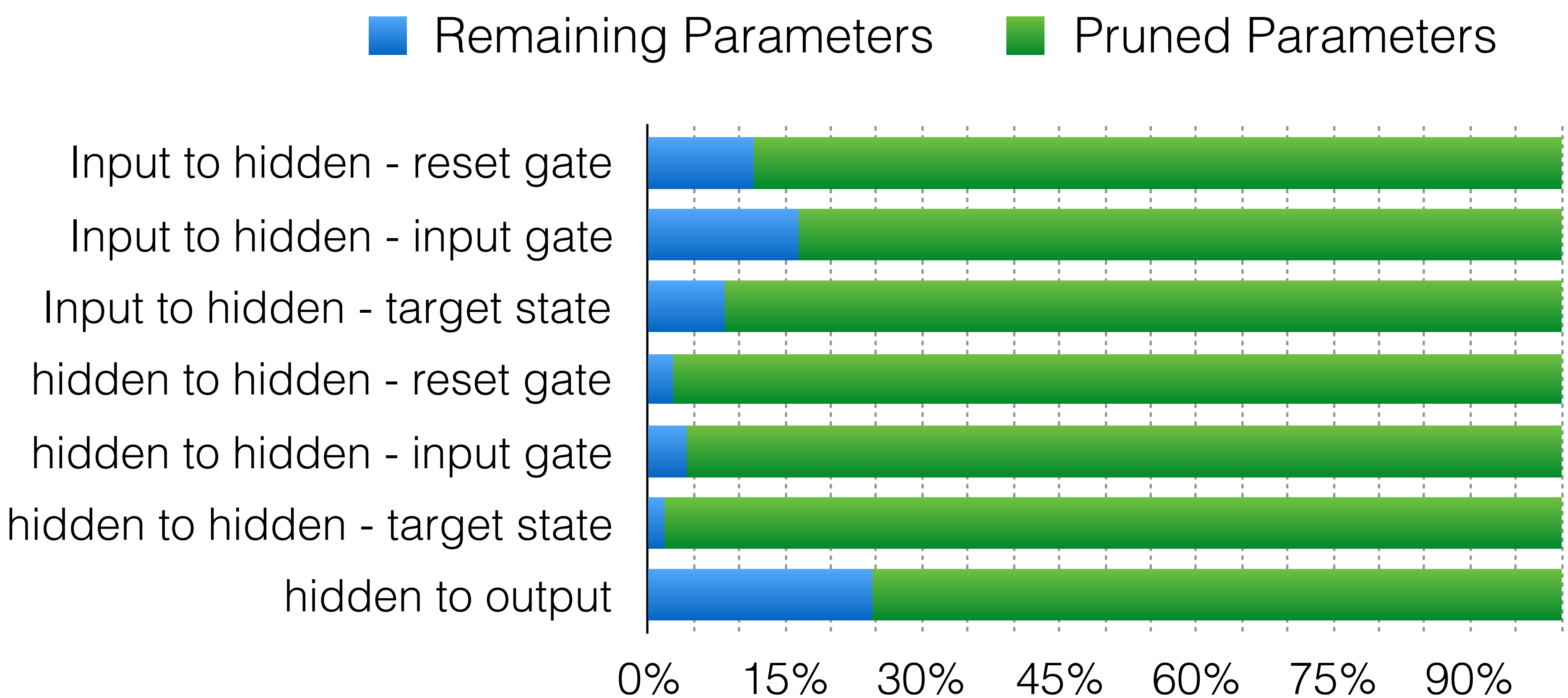
$$\boldsymbol{f}^{(t)} = \sigma\left(\boldsymbol{b}^f + \boldsymbol{W}^f\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{c}^{(t)} = \boldsymbol{f}^{(t)} \odot \boldsymbol{c}^{(t-1)} + \boldsymbol{i}^{(t)} \odot \boldsymbol{g}^{(t)}$$

$$\boldsymbol{o}^{(t)} = \sigma\left(\boldsymbol{b}^o + \boldsymbol{W}^o\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{h}^{(t)} = \boldsymbol{o}^{(t)} \odot \tanh(\boldsymbol{c}^{(t)})$$

1 Layer GRU - 95% Pruned Input Gate



Hidden to hidden more redundancy
Input gate less redundancy

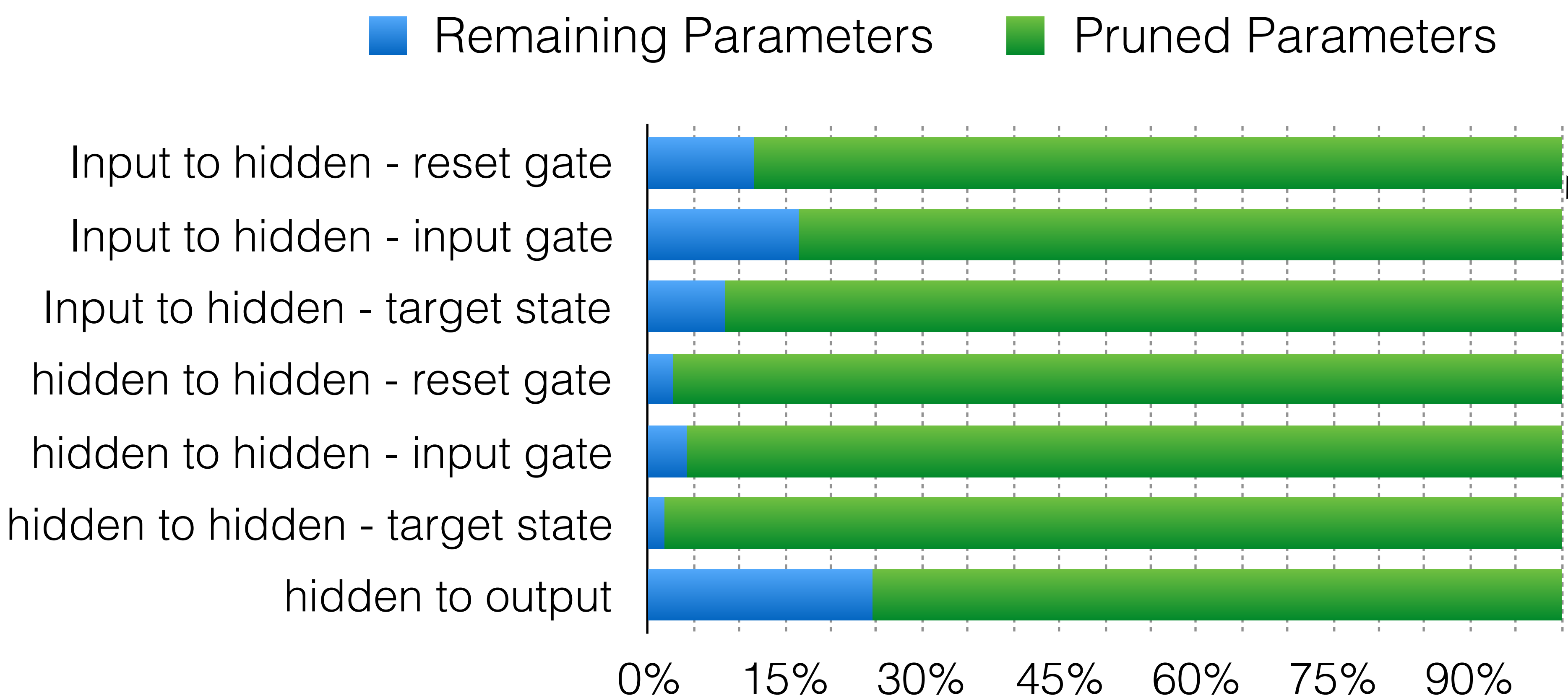
$$\boldsymbol{z}^{(t)} = \sigma\left(\boldsymbol{b}^{z} + \boldsymbol{W}^{z}\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{b}^{r} + \boldsymbol{W}^{r}\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^{g} + \boldsymbol{W}^{g}\left[\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \boldsymbol{g}^{(t)}$$

1 Layer GRU - 95% Pruned Reset Gate



Hidden to hidden more redundancy
Input gate less redundancy

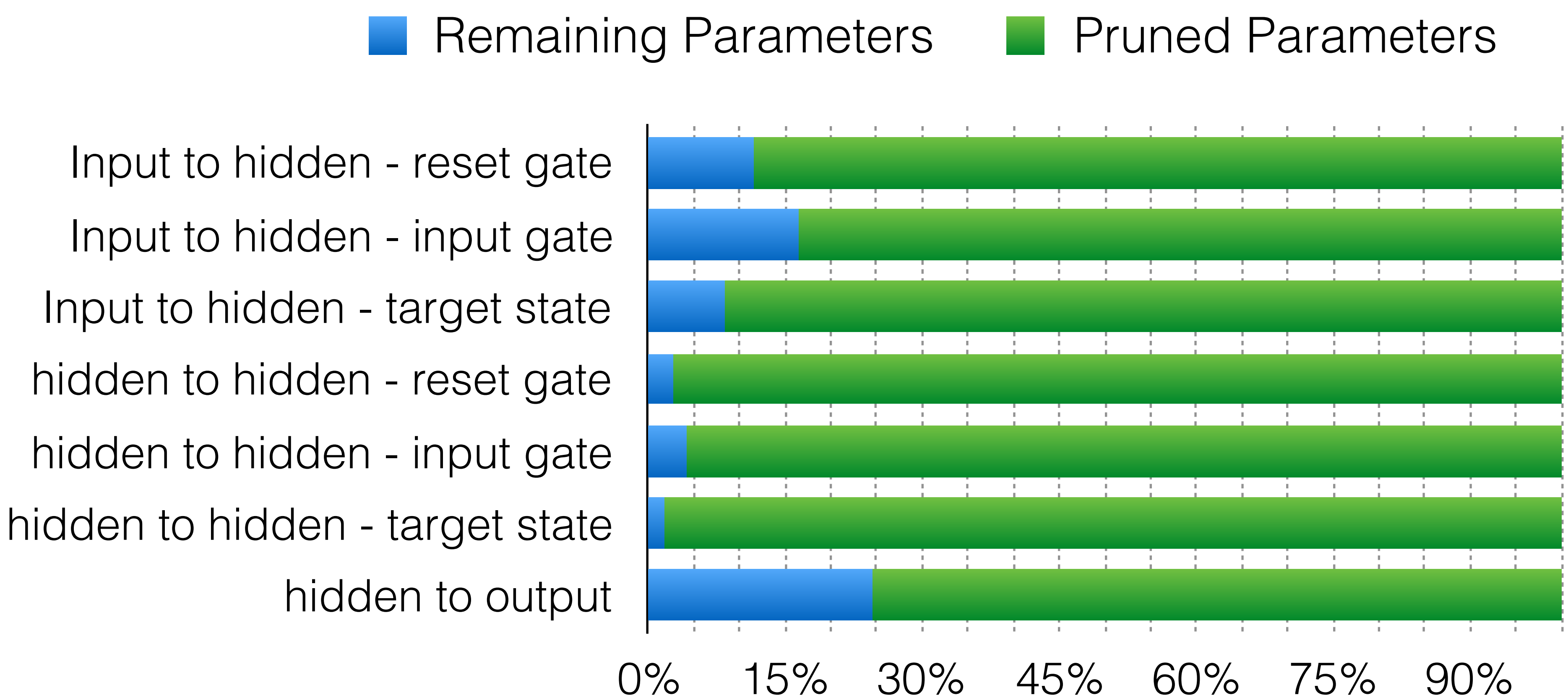
$$\boldsymbol{z}^{(t)} = \sigma\left(\boldsymbol{b}^z + \boldsymbol{W}^z\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{b}^r + \boldsymbol{W}^r\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g\left[\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \boldsymbol{g}^{(t)}$$

1 Layer GRU - 95% Pruned Target State



Hidden to hidden more redundancy
Input gate less redundancy

$$\boldsymbol{z}^{(t)} = \sigma\left(\boldsymbol{b}^z + \boldsymbol{W}^z\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{r}^{(t)} = \sigma\left(\boldsymbol{b}^r + \boldsymbol{W}^r\left[\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{g}^{(t)} = \tanh\left(\boldsymbol{b}^g + \boldsymbol{W}^g\left[\boldsymbol{r}^{(t)} \odot \boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}\right]\right)$$

$$\boldsymbol{h}^{(t)} = (1 - \boldsymbol{z}^{(t)}) \odot \boldsymbol{h}^{(t-1)} + \boldsymbol{z}^{(t)} \odot \boldsymbol{g}^{(t)}$$

Summary

- On MNIST, RNN, LSTM, GRU can be pruned by 95% without significant accuracy loss
- LSTM and GRU have more redundancy than RNN
- Hidden to hidden layers have more redundancy