Google Research

# Rigging The Lottery: Making All Tickets Winners (RigL)

## Efficient and accurate training for sparse networks.

*Utku Evci, Trevor Gale, Jacob Menick, Pablo Samuel Castro, Erich Elsen*

# AI Residency

Google AI Residency

Brain Montreal

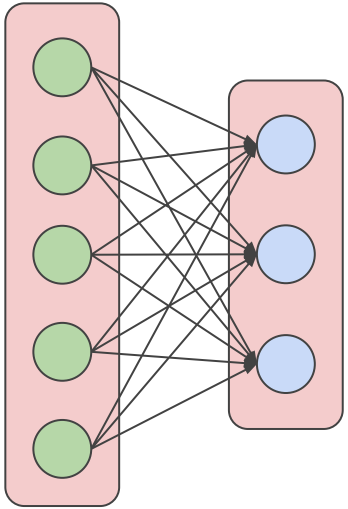
DeepMind

Google Research

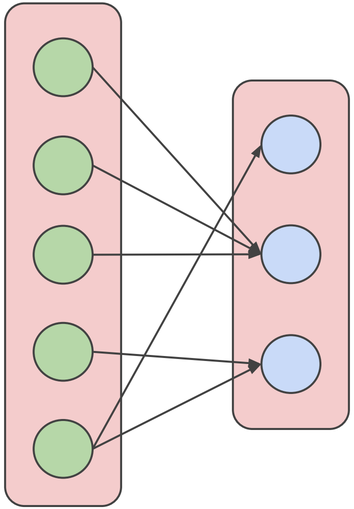
# Motivation

Sparse networks and their advantages

# Sparse Networks



Dense Network
Sparsity: 0



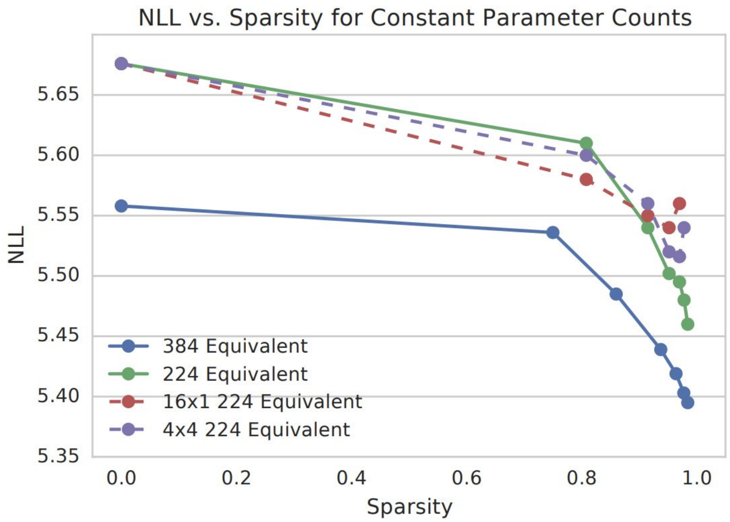
Sparse Network
Sparsity: 60%

Google Research

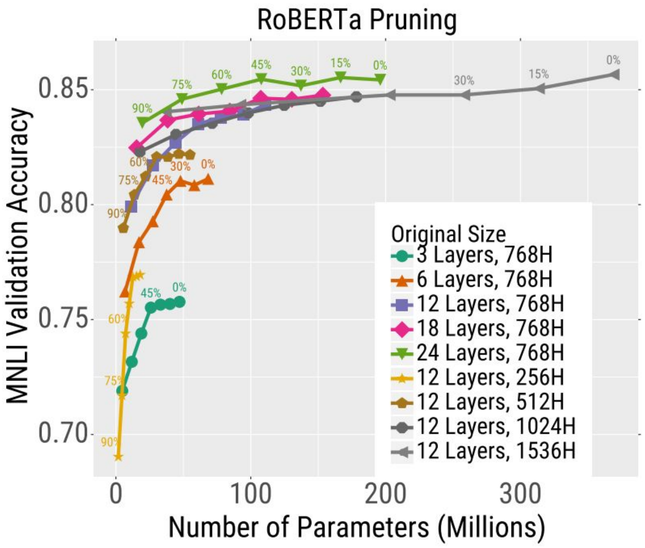
- ➔ **On device training/inference:** Reduces FLOPs and network size drastically without harming performance.
- ➔ **Reduced memory footprint:** We can fit wider/deeper networks in memory and get better performance from the same hardware.
- ➔ **Architecture search:** Causal relationships? Interpretability?

# Sparse networks perform better

*for the same parameter count.*




**Efficient Neural Audio Synthesis**, Kalchbrenner, N., Elsen, E., Simonyan, K., Noury, S., Casagrande, N., Lockhart, E., Stimberg, F., Oord, A.,Dieleman, S., and Kavukcuoglu, K., 2018




**Train Large, Then Compress: Rethinking Model Size for Efficient Training and Inference of Transformers**, Zhuohan Li, Eric Wallace, Sheng Shen, Kevin Lin, Kurt Keutzer, Dan Klein, Joseph E. Gonzalez, 2020

# Accelerating sparsity

*Is difficult, but possible*

→ Block Sparse Kernels: Efficient sparse operations.

→ Efficient WaveRNN: Text-to-Speech

→ Optimizing Speech Recognition for the Edge: Speech Recognition

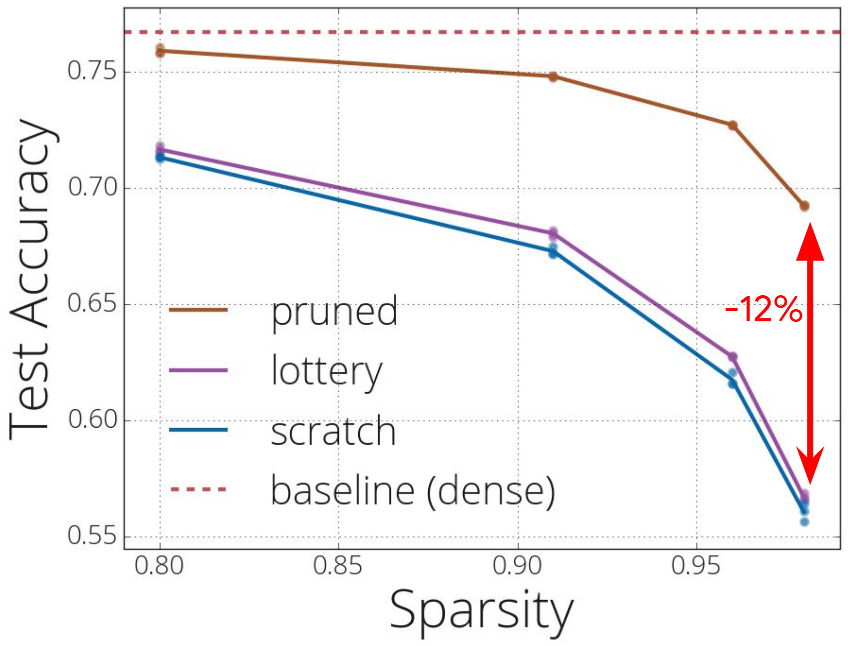
→ Fast Sparse Convnets: Fast mobile inference for vision models with sparsity (1.3 - 2.4x faster).

....and more to come.

Google Research

# How do we find sparse networks?

- ➔ **Pruning** method requires dense training: (1) limits the biggest sparse network we can train (2) not efficient.
- ➔ **Training from scratch** performs much worse.
- ➔ **Lottery*** initialization doesn't help.




Test accuracy of ResNet-50 networks trained on ImageNet-2012 dataset at different sparsity levels*.

*** The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks**
Jonathan Frankle, Michael Carbin, ICLR 2019

*** The Difficulty of Training Sparse Neural Networks**
Utku Evci, Fabian Pedregosa, Aidan Gomez, Erich Elsen, 2019

# Can we train sparse neural networks end-to-end?

(without ever needing the dense parameterization)

(as good as the dense-to-sparse methods)

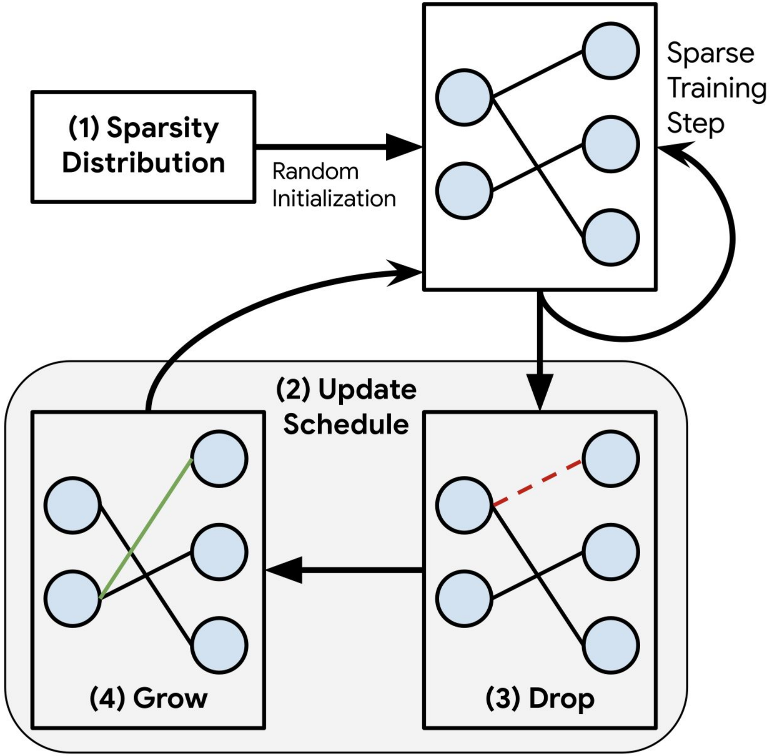
YES. RigL!

# Rigging The Lottery:

## Making all Tickets Winners

and surpassing pruning performance.

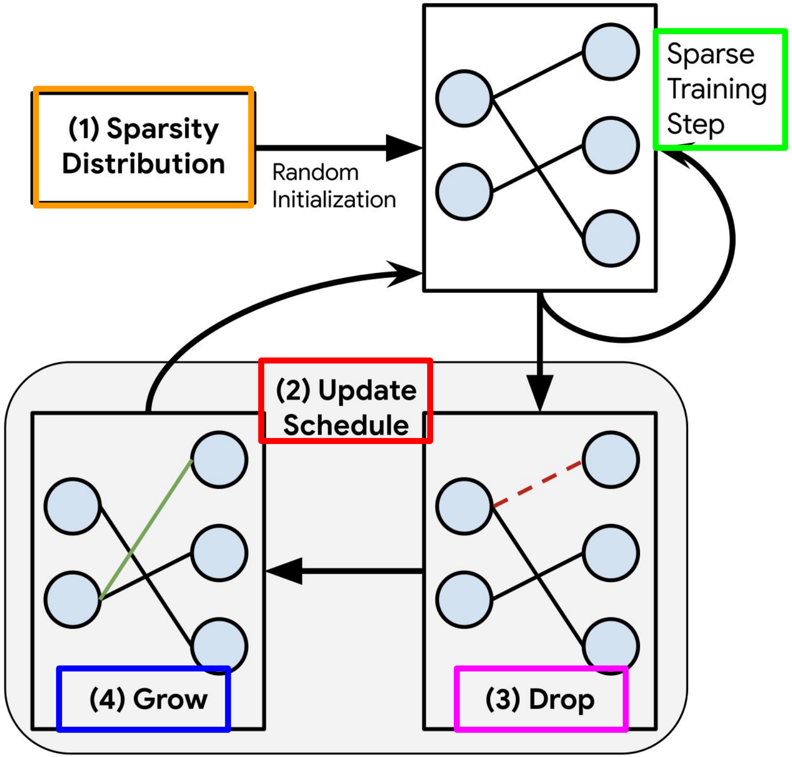
# The Algorithm




- ➔ **Start from a random sparse network.**
- ➔ **Train the sparse network.**
- ➔ **Every N steps update connectivity:**
  - ◆ Drop least magnitude connections
  - ◆ Grow new ones using gradient information.

Google Research

# The Algorithm




**Algorithm 1** RigL

**Input:** Network $f_\Theta$, dataset $D$

  Sparsity Distribution: $\mathbb{S} = \{s^1, \dots, s^L\}$

  Update Schedule: $\Delta T$, $T_{end}$, $\alpha$, $f_{decay}$

$\theta \leftarrow$ Randomly sparsify $\Theta$ using $\mathbb{S}$

**for** each training step $t$ **do**

  Sample a batch $B_t \sim D$

  $L_t = \sum_{i \sim B_t} L((f_\theta(x_i), y_i)$

  **if** $t \pmod{\Delta T} == 0$ and $t < T_{end}$ **then**

    **for** each layer $l$ **do**

      $k = f_{decay}(t; \alpha, T_{end})(1 - s^l)N^l$

      $\mathbb{I}_{drop} = ArgTopK(-|\theta^l|, k)$

      $\mathbb{I}_{grow} = ArgTopK_{i \notin \theta^l \setminus \mathbb{I}_{drop}}(|\nabla_{\Theta^l} L_t|, k)$

      $\theta \leftarrow$ Update connections $\theta$ using $\mathbb{I}_{drop}$ and $\mathbb{I}_{grow}$

    **end for**

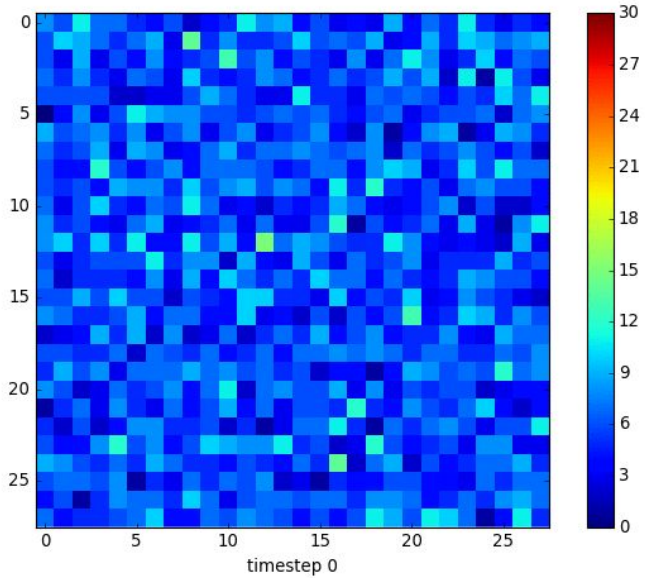
  **else**

    $\theta = \theta - \alpha \nabla_\theta L_t$

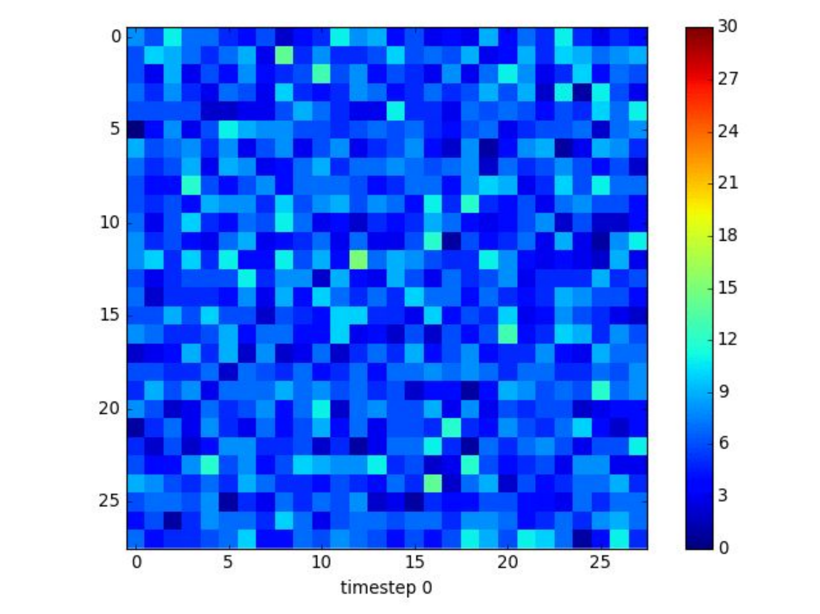
  **end if**

**end for**

# Evaluation of Connections



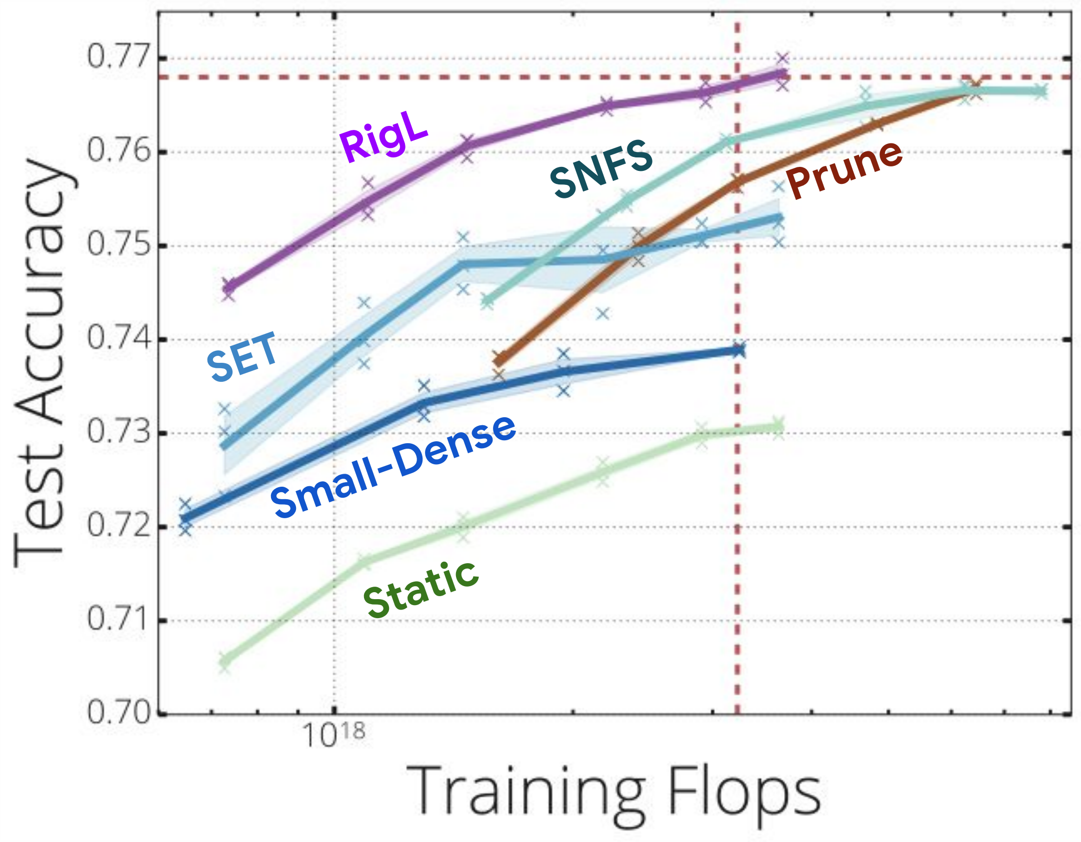

Before training




Evaluation of first layer connections during MNIST MLP training.

# Resnet-50

RigL matches pruning performance using significantly less resources.

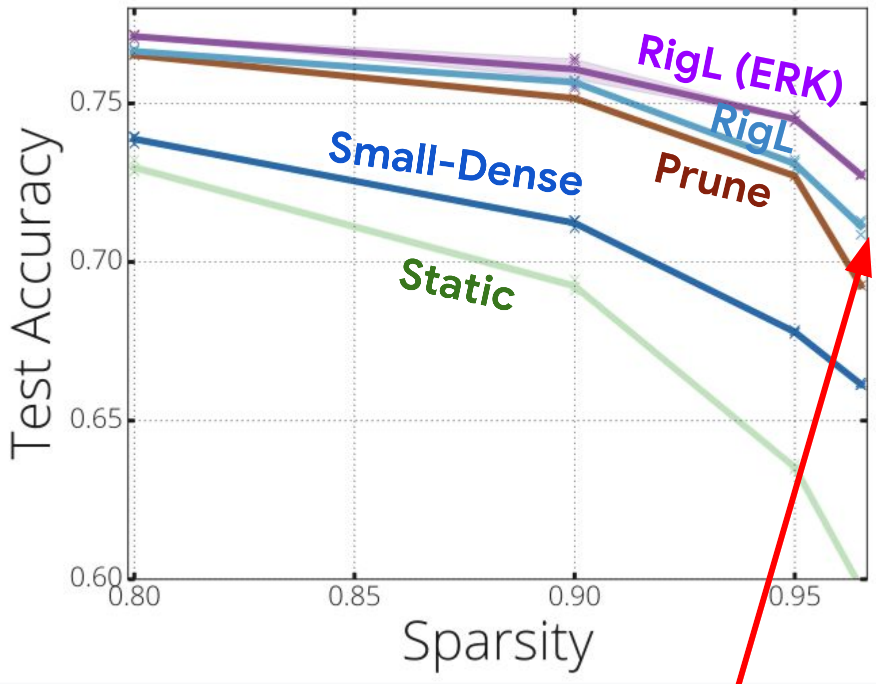



Google Research

Google Research

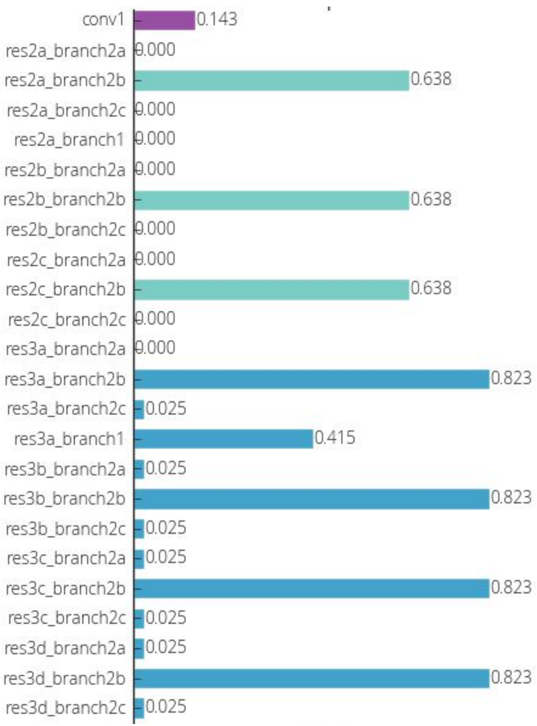
# Exceeding Pruning Performance

- ➔ RigL
  - ◆ **outperforms** pruning
- ➔ ERK sparsity distribution:
  - ◆ greater performance
  - ◆ more FLOPs.

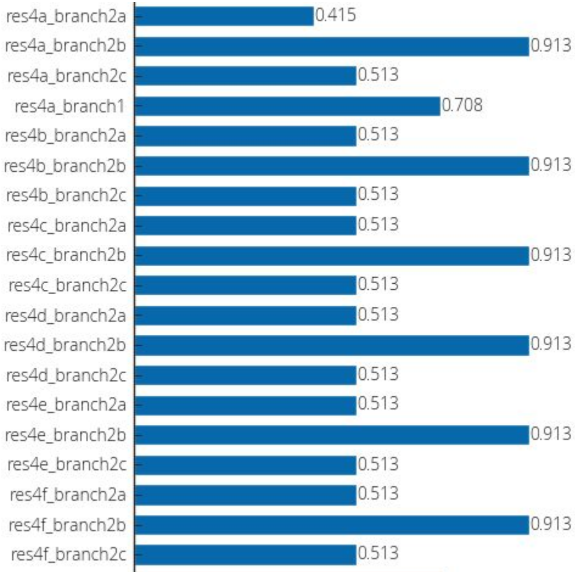



14x less FLOPs and parameters.

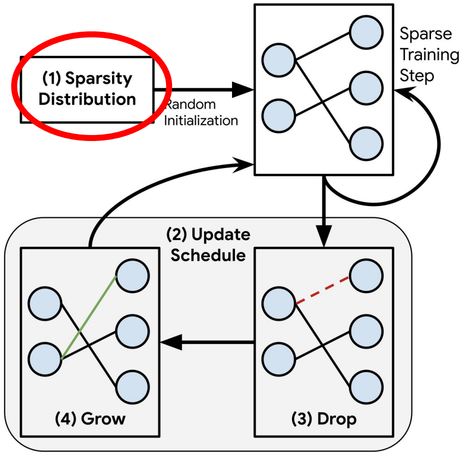
# 80% ERK on Resnet-50

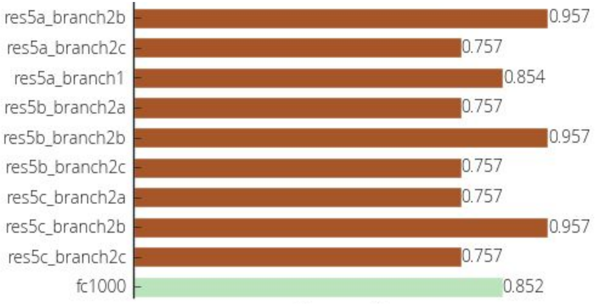



Stage 1-2-3




Stage 4




Stage 5

# Sparse MobileNets

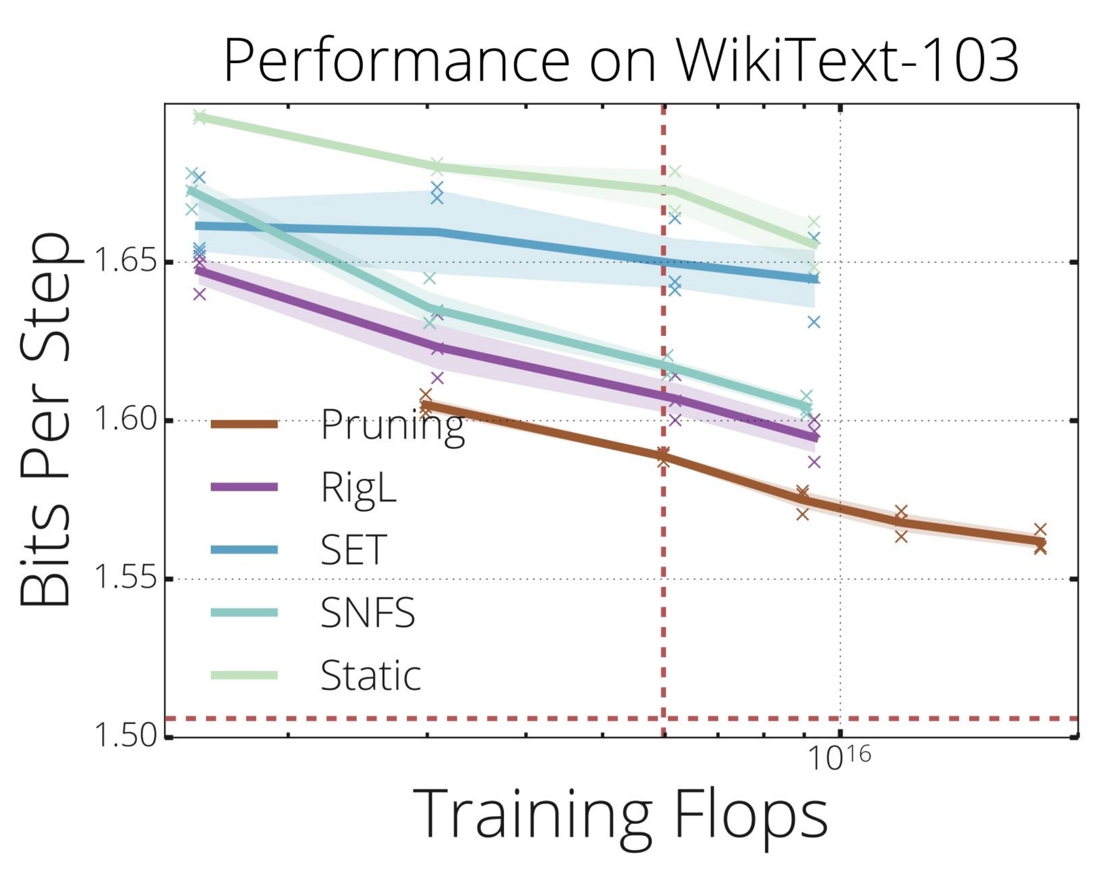
- Difficult to prune.
- Much better results with **RigL.**

| S | Method | Top-1 | FLOP-Inf |
|---|---|---|---|
| 0.75 | Small-Dense$_{5\times}$ | 66.0±0.11 | 0.23x |
| | Pruning (Zhu) | 67.7 | 0.27x |
| | RigL$_{5\times}$ | 71.5±0.06 | 0.27x |
| | RigL$_{5\times}$ (ERK) | **71.9±0.01** | 0.52x |
| 0.90 | Small-Dense$_{5\times}$ | 57.7±0.34 | 0.09x |
| | Pruning (Zhu) | 61.8 | 0.12x |
| | RigL$_{5\times}$ | 67.0±0.17 | 0.12x |
| | RigL$_{5\times}$ (ERK) | **68.1±0.11** | 0.27x |
| | Dense | 72.1±0.17 | 1x (1.1e9) |
| 0.75 | Big-Sparse$_{5\times}$ | 76.4±0.05 | 0.98x |
| | Big-Sparse$_{5\times}$ (ERK) | **77.0±0.08** | 1.91x |

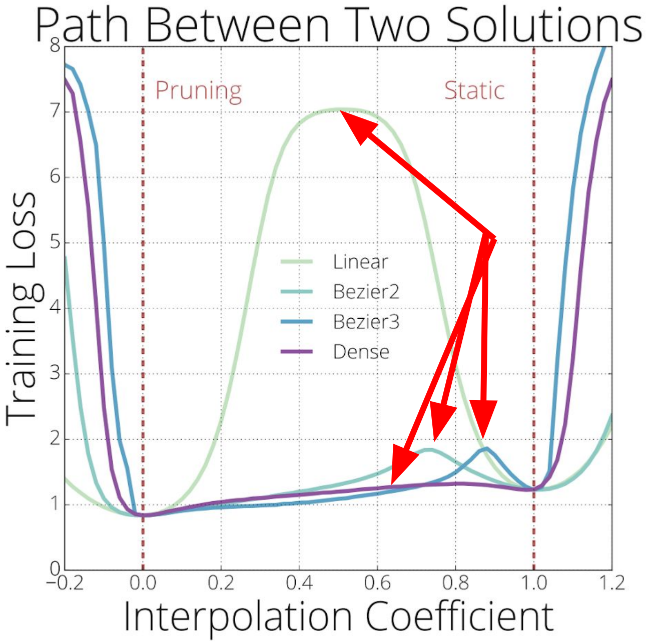
same parameter/flops: **4.3% absolute improvement** in Top-1 Accuracy.

# Character Level Language Modelling on WikiText-103

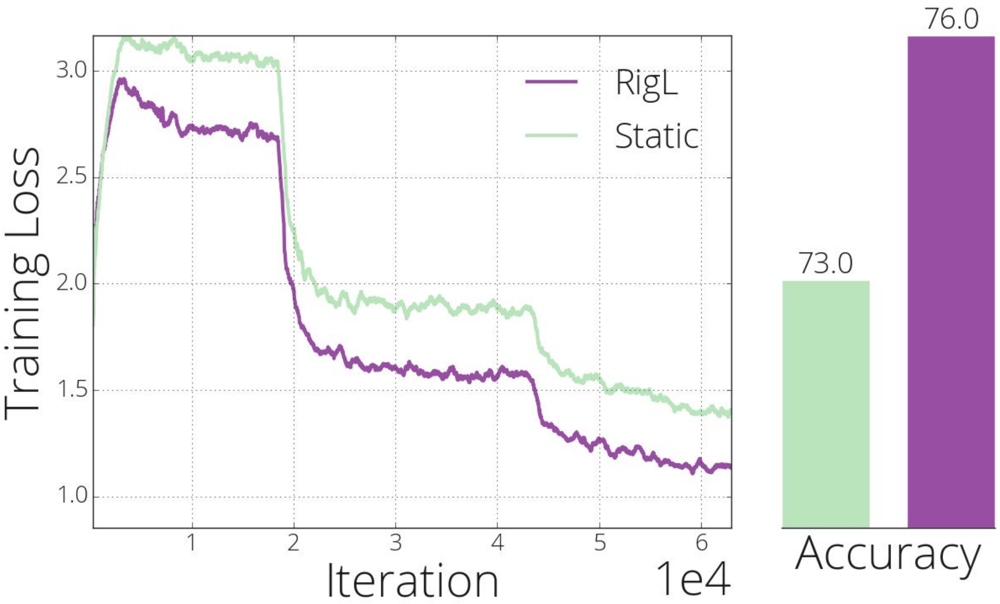
- Similar to WaveRNN.
- RigL falls short of matching the pruning performance



# Bad Local Minima and RigL




Static training stucks in a suboptimal basin.




RigL helps escaping from it.

More results in our workshop paper
https://arxiv.org/pdf/1906.10732.pdf

Google Research

## Summary of Contributions

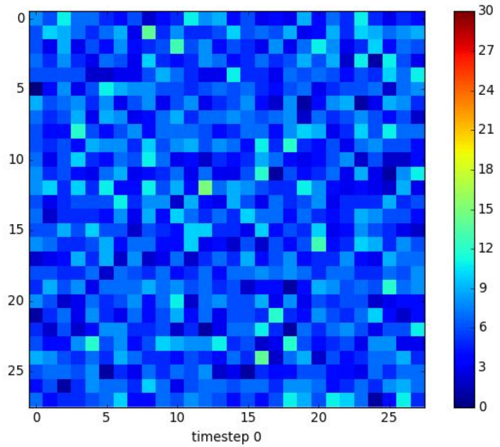
- Training randomly initialized sparse networks without needing the dense parametrization is possible.
- Sparse networks found by RigL can exceed the performance of pruning.
- Non-uniform sparsity distribution like ERK brings better performance.
- RigL can helps us with feature selection.

## Limitations

- RigL requires more iterations to converge.
- Dynamic sparse training methods seem to have suboptimal performance training RNNs.
- Sparse kernels that can utilize sparsity during training are not widely available (yet).

# Thank you!

- Sparse networks are promising.
- End-to-end sparse training is possible and it has potential to replace dense->sparse training.




@evcu

@tgale

@jmenick

@psc

@eriche

https://github.com/google-research/rigl