The Lottery Ticket Hypothesis

Finding Sparse, Trainable Neural Networks

Aditya Mehta   Nikhil Goyal   Shardul Junagade

ES667: Deep Learning
Indian Institute of Technology Gandhinagar

December 4, 2025

Problem Description

The Pruning Paradox

Challenges in large networks:

- Overparameterized networks: millions to billions of parameters
- Deployment constraints: mobile devices, edge computing
- Post-training pruning works, but sparse networks fail when trained from scratch
- Difficult to find such optimal parameter to be pruned

Key Question

If networks can be pruned by 90%+ after training, why can't we train smaller networks from scratch?

The Lottery Ticket Hypothesis

Frankle & Carbin (2019)

"A randomly-initialized dense network contains a subnetwork that is initialized such that, when trained in isolation, it can match the test accuracy of the original network."

Key Insights:

- Dense networks contain sparse subnetworks ("winning tickets")
- Same structure + random initialization = poor performance
- **Initialization matters!**

Research Objectives

1. Validate winning ticket existence across architectures and datasets
2. Prove reinitialization to the original /starting weights is mandatory
3. Characterize sparsity ranges where LTH holds
4. Analyze learning dynamics and convergence while training
5. Examine the effect of various pruning techniques on generalization error

Solution Approach

Experimental Setup

Datasets:

- MNIST (handwritten digits)
- Fashion-MNIST (clothing items)
- CIFAR-10 (natural images)

Architectures:

- LeNet-300-100 (266K params)
- LeNet-5 (61K params)
- Conv-6 (1.2M params)

Training Config:

- Optimizer: Adam
- Learning rate: 0.0012
- Iterations: 40,000
- Batch Size: 60
- Pruning rate: 30% *
- Rounds: 16
- Pruning: IMP or Random
- Initialization: Rewind or Random

* 30% of existing weights per layer

IMP Algorithm

Algorithm 1 Iterative Magnitude Pruning (IMP)

```
Initialize weights θ_0 ∼ 𝒟; store θ_init ← θ_0; mask m ← 1
for round r = 1 to N do
    Train sparse network: optimize θ using f(x; m ⊙ θ) for T steps
    for each layer ℓ do
        Compute pruning threshold τ_ℓ = p-th percentile of |θ^(ℓ)| among currently active weights
        Update mask: m^(ℓ) ← m^(ℓ) ⊙ 𝕀{|θ^(ℓ)| > τ_ℓ} (cumulative)
    end for
    Rewind: θ ← m ⊙ θ_init
end for
Return sparse subnetwork (m, θ_init)
```

Key: Reset surviving weights to *original* initialization, not trained values!

Experimental Designs

Three Experiment strategies:

1. **Magnitude + Rewind:** Prune low-magnitude weights, reset to θ_0
   Purpose: Find winning tickets
2. **Magnitude + Random:** Same masks, reset to new random θ'_0
   Purpose: Test if initialization matters
3. **Random + Rewind:** Random pruning, reset to θ_0
   Purpose: Test if structure alone works

Data Split and Evaluation Strategy

Data Split:

- Training set: $\sim 78.5\%$
- Validation set: $\sim 7.1\%$
- Test set: $\sim 14.3\%$

Evaluation Procedure:

- Train the network across full epochs.
- After each epoch, compute validation loss.
- **Early Stop Iteration**: the iteration corresponding to the *minimum* validation loss.
- For each pruning round (40,000 iterations $\approx$ 44 epochs), test accuracy is reported at the early-stopped iteration.

Results

Winning Tickets Exist!

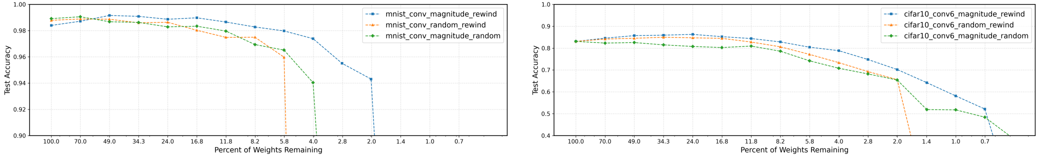


Figure: MNIST LeNet-5 (left) peaks at 99.16% with 49% sparsity. CIFAR-10 Conv-6 (right) improves 3.21% at 24% remaining weights!

Winning Ticket Performance Summary

Dataset-Model	Peak Acc	Sparcity	Range*
MNIST-FC	98.35%	24%	24-6%
MNIST-Conv	**99.16%**	49%	49-6%
Fashion-FC	89.10%	17%	17-6%
Fashion-Conv	88.92%	49%	49-9%
CIFAR10-Conv6	86.30%	24%	24-6%

***Key Finding:** Networks maintain accuracy at 5% of original size!

Initialization is Critical

Dataset-Model	Sparsity	Rewind	Random	Gap
MNIST-FC	1.98%	96.30%	76.55%	**19.75%**
MNIST-Conv	1.99%	94.31%	60.72%	**33.59%**
Fashion-FC	4.05%	86.25%	63.02%	**23.23%**

Critical Finding

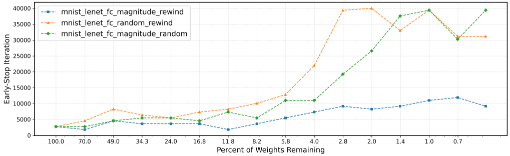
At extreme sparsity, random reinitialization causes **19-33% accuracy drops**. Structure alone is insufficient, initialization is essential!

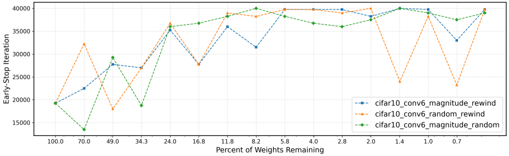
Magnitude Beats Random Pruning

Dataset-Model	Sparsity	Magnitude	Random	Gap
MNIST-FC	5.77%	97.79%	96.05%	1.74%
Fashion-Conv	5.78%	88.37%	84.27%	**4.10%**
CIFAR10-Conv6	5.77%	80.44%	77.15%	3.29%

Critical Finding:

Magnitude-based weight selection is superior to random masking (1.74-4.10% advantage).
Reinforces the idea that initialization rule matters more than pruning rule





Observations:

- As the % of remaining parameters decreases, the model takes more iterations to converge
- Random reinitialization needs much more iterations as compared to rewind
- The trend is easily observable in simple datasets like MNIST, but much more noisier in case of complex datasets like CIFAR-10

Conclusions

Limitations

- LTH validated only on small vision **datasets**.
- LTH validated only using small models.
- Unstructured sparsity is not **hardware-friendly**.
- We only test for magnitude pruning and random pruning.

Future Work

1. Apply LTH to **deeper architectures** (ResNets, VGG)
2. Test hypotheses for larger datasets
3. Structured pruning (**hardware-efficient pruning**)
4. Explore layer-wise adaptive pruning rates
5. **Early prediction** of winning tickets (before full training)

Key Findings

1. **LTH validated:** Winning tickets exist across all architectures
   - Networks maintain 98% accuracy at 5-10% sparsity (MNIST)
   - Pruning can *improve* accuracy (3.21% on CIFAR-10)
2. **Regularizing effect:** We observe an implicit regularizing effect
3. **Initialization critical:** 19-33% drops with random reinit
4. **Magnitude pruning wins:** 1.74-4.10% better than random
5. **Dataset dependency:** LTH stability varies with task complexity

Why Dense Networks Work

They contain multiple lottery tickets with favorable initializations!

Thank You!

Questions?

GitHub: `https://github.com/aditya-me13/lottery-ticket-hypothesis`

Appendix

Detailed Dataset Specifications

Dataset	Details
MNIST	28×28 grayscale images, 10 classes (digits 0-9) Split: 55,000 train / 5,000 val / 10,000 test Normalization: $\mu = 0.1307$, $\sigma = 0.3081$
Fashion-MNIST	28×28 grayscale images, 10 classes (clothing items) Split: 55,000 train / 5,000 val / 10,000 test Normalization: $\mu = 0.2860$, $\sigma = 0.3530$
CIFAR-10	32×32 RGB images, 10 classes (natural objects) Split: 45,000 train / 5,000 val / 10,000 test Per-channel normalization: $\mu = (0.4914, 0.4822, 0.4465)$ $\sigma = (0.2023, 0.1994, 0.2010)$

Model Architecture Details

LeNet-300-100 (Fully Connected):

- Architecture: Input (784) $\rightarrow$ FC1 (300) $\rightarrow$ ReLU $\rightarrow$ FC2 (100) $\rightarrow$ ReLU $\rightarrow$ Output (10)
- Total parameters: $\sim$266,000
- Used for: MNIST, Fashion-MNIST

LeNet-5 (Convolutional):

- Conv1: 1 $\rightarrow$ 6 (5$\times$5) $\rightarrow$ ReLU $\rightarrow$ MaxPool (2$\times$2)
- Conv2: 6 $\rightarrow$ 16 (5$\times$5) $\rightarrow$ ReLU $\rightarrow$ MaxPool (2$\times$2)
- FC: 256 $\rightarrow$ 120 $\rightarrow$ 84 $\rightarrow$ 10
- Total parameters: $\sim$61,000
- Used for: MNIST, Fashion-MNIST

Conv-6 Architecture

Deep VGG-style Convolutional Network:

Block 1:

- Conv1: $3 \to 64$ (3×3, padding=1) $\to$ ReLU
- Conv2: $64 \to 64$ (3×3, padding=1) $\to$ ReLU
- MaxPool (2×2)

Block 2:

- Conv3: $64 \to 128$ (3×3, padding=1) $\to$ ReLU
- Conv4: $128 \to 128$ (3×3, padding=1) $\to$ ReLU
- MaxPool (2×2)

Block 3:

- Conv5: $128 \to 256$ (3×3, padding=1) $\to$ ReLU
- Conv6: $256 \to 256$ (3×3, padding=1) $\to$ ReLU
- MaxPool (2×2)

Classifier: FC: $4096 \to 256 \to 10$

Complete Training Configuration

Optimization:

- Optimizer: Adam
- Learning rate: 0.0012
- Loss: Cross-entropy
- Batch size: 60 (128 for Conv-6)
- Total iterations: 40,000

Pruning:

- Pruning rate: 30% per round
- Total rounds: 16
- Scope: Layer-wise
- Strategy: Magnitude-based
- Mask enforcement: After every gradient step

Initialization:

- Method: Kaiming Normal
- Mode: fan_in
- Non-linearity: ReLU

Hardware:

- Platform: Kaggle Notebooks
- GPU: NVIDIA Tesla T4
- Framework: PyTorch 2.0+

Evaluation Metrics

- **Test Accuracy:** Final accuracy on test set after each pruning round
- **Validation Accuracy:** Used for early stopping criterion
- **Early-Stop Iteration:** Iteration number at minimum validation loss (proxy for convergence speed)
- **Sparsity Level:** Percentage of pruned weights

$$\text{Sparsity} = \left(1 - \frac{\|m\|_0}{\|\theta\|}\right) \times 100\%$$

- **Winning Ticket Range:** Pruning rounds where test accuracy remains within 2% of dense baseline
- **Accuracy Gap:** Difference between rewind and random reinitialization strategies