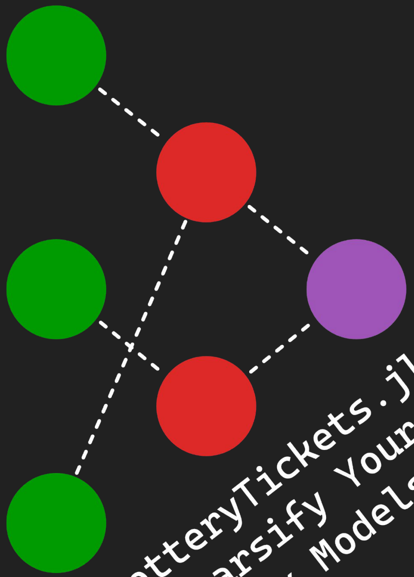


# LotteryTickets.jl
## Sparsify Your Flux Models

Marco Cognetta

# About Me

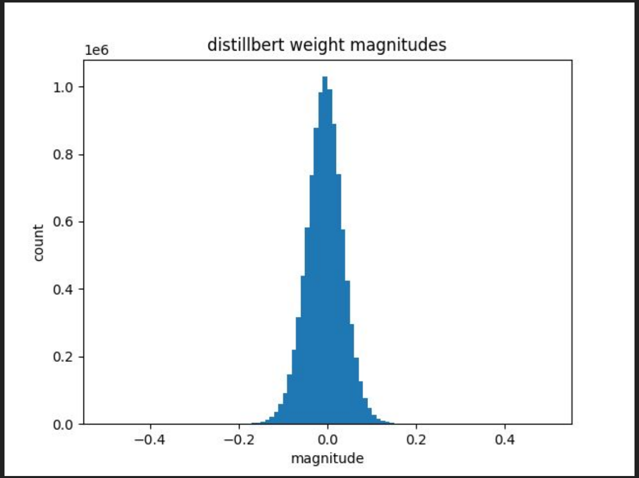
- PhD Student at Tokyo Institute of Technology
  - Tokenization and machine translation
- PhD Student Researcher at Google Tokyo
  - Keyboard language modeling and federated learning
- I am interested in making models *smaller* and *faster*

# A Problem with Modern Neural Networks

- Deep neural networks require too much space and are too slow
- Many orthogonal ways to address this:
  - Quantization
  - Distillation
  - Low-rank approximations

# Another Approach

- Observation: deep neural models are *overparameterized*
  - Tend to train faster and be more robust to noise
  - Implemented as dense matrix operations which is nice for current hardware
  - However, most parameters are not very impactful



# The Lottery Ticket Hypothesis

*Deep, dense neural networks contain sparse subnetworks that account for most of the performance of the overall model.*

# The Lottery Ticket Hypothesis

*Deep, dense neural networks contain sparse subnetworks that account for most of the performance of the overall model.*

- Some results find subnetworks (winning tickets) that have <10% of the number of parameters of the original model
- This enables models to run on hyper-resource-constrained systems

# Finding Lottery Tickets

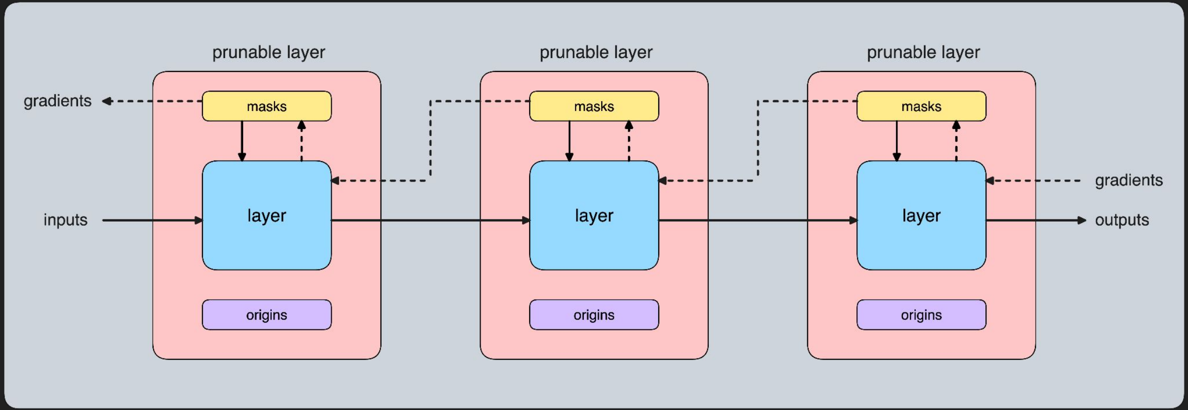
- An extremely simple loop
  - Train model to convergence
  - Prune the bottom X% of parameters
  - Reset the model
    - Reset to the *initial* parameters
  - Repeat until performance degrades

# LotteryTickets.jl

- Prunable layer wrappers
  - All Flux layers are supported
    - Dense → PrunableDense
  - An easy interface for defining prunable wrappers for custom layers
  - Mimics sparse matrices and gradients
- Pruner types
  - Describes the method for choosing parameters to prune
  - MagnitudePruneGroup
    - Prunes a group of layers collectively by magnitude

# Core (Layers)

- Thinly wrap layers to capture and mask gradients and to retain the initialization weights




- Masking vs Sparse Representations
  - Sparse-dense matrix multiply is substantially slower than element-wise-product + dense-dense multiply until things are really sparse

```julia
using Flux, LotteryTickets

function main(config)
  m = Chain(
        # a prunable dense layer
        PrunableDense(1024 => 256),
        # all flux layers are supported
        PrunableLSTM(256 => 256),
        PrunableDense(256 => 64),
        # mixing prunable and non-prunable is ok!
        Dense(64 => 10),
  )

  # pruning groups
  g1 = MagnitudePruneGroup([m[1], m[2]], 0.2)
  g2 = MagnitudePruneGroup([m[3]], 0.1)

  # the pruner controller
  p = Pruner([g1, g2])

  for _ in 1:config.pruning_rounds

    # run a full training job to convergence
    train_model!(m, config)

    # prune and reset the model for the next
    # training round
    pruneandrewind!(p)
  end

  # convert the model to a sparse representation
  sparsify(m)
end
```

```julia
julia> @prunable Chain(Dense(2=>5), Dense(5=>2))
Chain(
  PrunableDense(
    Dense(2 => 5),                    # 15 parameters
  ),
  PrunableDense(
    Dense(5 => 2),                    # 12 parameters
  ),
)         # Total: 4 trainable arrays, 27 parameters,
          # plus 4 non-trainable, 40 parameters
```

# Other Interesting Projects

- The Lottery Ticket Hypothesis: Finding Small, Trainable Neural Networks, *Frankle & Bieber (ICLR 2019)*
  - The original paper on this idea
- OpenLTH
  - A Python library for Lottery Ticket style pruning
- TinyNets.jl
  - A Julia library for iterative pruning
- Proving the Lottery Ticket Hypothesis for Convolutional Neural Networks, *da Cunha et al. (ICLR 2022)*
  - The authors use Julia to implement their experiments!
- EfficientML
  - An MIT course on efficient deep learning (including sparsification)

# Thanks!

repo

my site