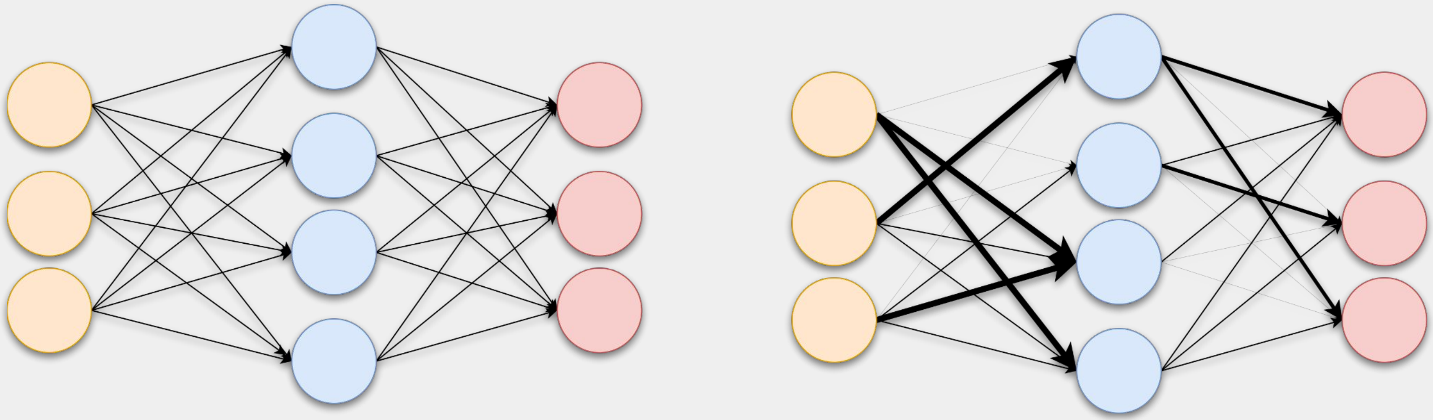
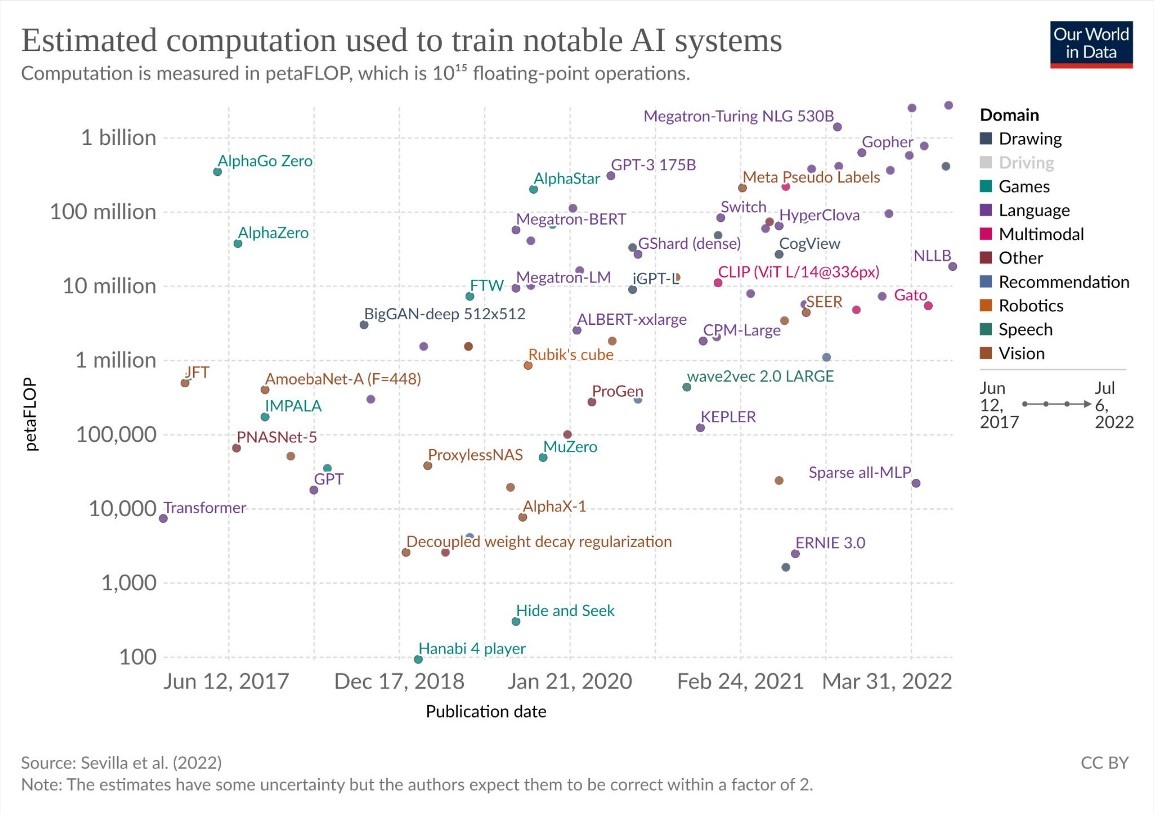
# Learning Dynamic Networks



Kale-ab Tessera, Chiratidzo Matowe, Arnu Pretorius,
Benjamin Rosman, Sara Hooker

# Motivation - Era of Large Scale Models




**Overparameterized** models have led to many **breakthroughs** in machine learning (Chowdhery et al., 2022; Brown et al., 2020; Zhai et al., 2021; Reed et al., 2022).

**Challenges**:

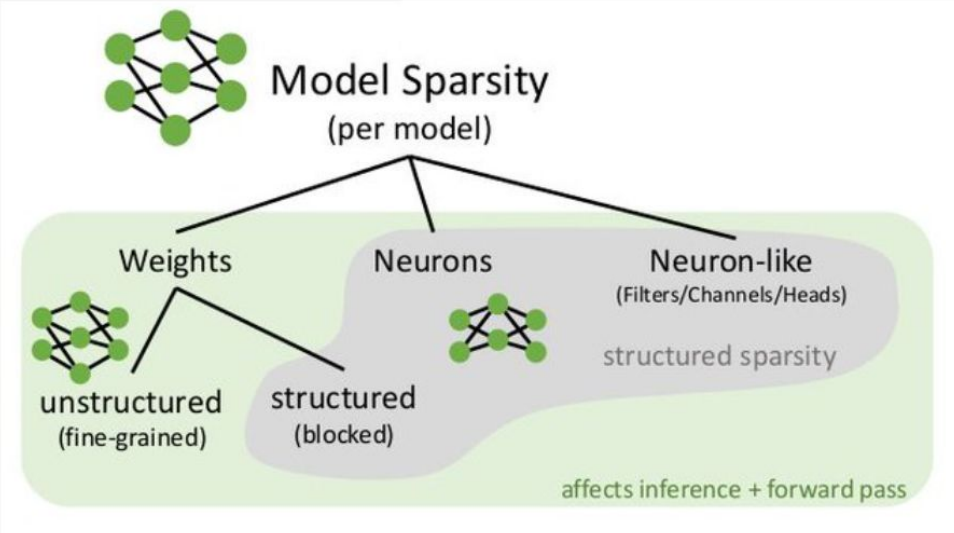
- ❖ Larger Models
  - ➢ Efficient **storage** and inference.
  - ➢ Efficient **training** of large models.
  - ➢ **Overfitting**, regularization and generalization.
- ❖ Train Longer
  - ➢ Handle **temporal** dynamics of training.

Source: Sevilla et al. (2022) and OurWorldInData

# Motivation - Sparsity

Common method to handle challenges of overparameterization - Sparsity/Pruning.

Benefits - similar performance, with a **fraction of the weights** (Gale et al., 2019; Frankle & Carbin, 2019), **faster** training (Dettmers & Zettlemoyer, 2019; Luo et al., 2017) and more **robust to noise** (Ahmad & Scheinkman, 2019).




Source: Hoefler et al. (2021)

# Train Longer - Schedules

When we train longer -> more temporal decisions to make.

**Temporal Decisions** (examples include):

➢ *Learning Rate:*
  - Initial Learning Rate.
  - Function for the rate of change - **Need LR Schedule.**

➢ *Sparsity:*
  - Initial Sparsity (% active neurons, weights, filters, channels etc).
  - Function for the rate of change - **Need Sparsity Schedule.**

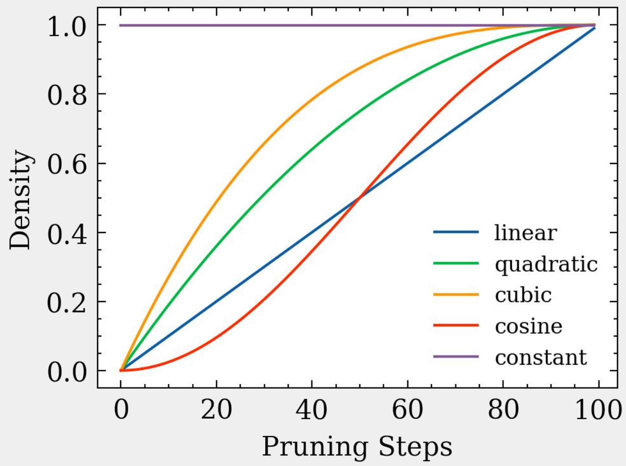
What about these choices per layer?! - **Layerwise Schedules.**

Standard approach - learn these schedules through **trial-and-error**.

**Related Work**

**Handcrafted** schedules.

- **Constant** - SET [1], Deep Rewiring (DeepR) [2] and Neural Network Synthesis Tool (NEST) [3].
- **Cosine** - RigL [4] and Sparse Network From Scratch (SNFS) [5]
- **Cubic** - [6],[7].



# Our Approach - Can we learn these schedules using RL?

**Algorithm 1** Learning Sparsity Schedules using Reinforcement Learning

**Input:** train dataset $X_{train\_set}$, test dataset $X_{test\_set}$, train network $f_{train}$, eval network $f_{eval}$, agent $a$, number of episodes $N$, minimum density per layer $min_d$ and maximum density per layer $max_d$.

$a \leftarrow init(min_d, max_d)$ ▷ Initialize agent $a$.

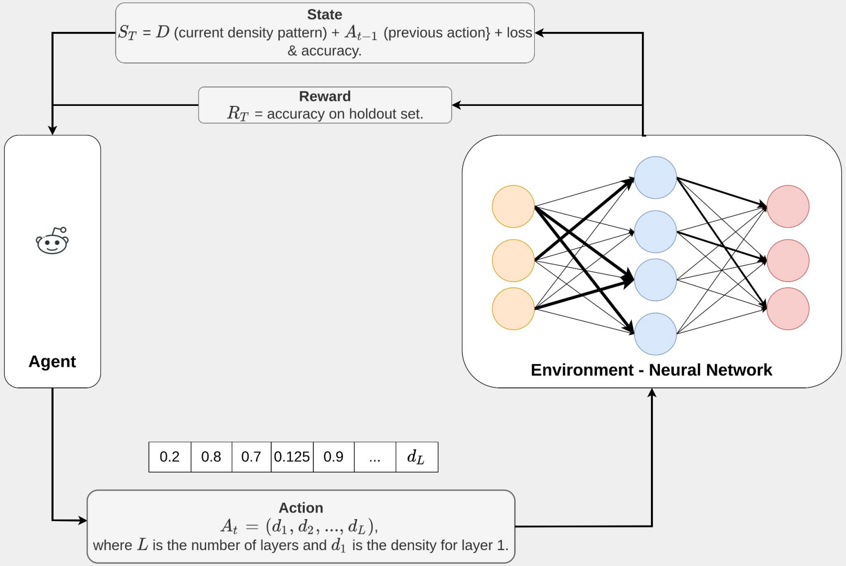
$X_{train\_split}, X_{val\_split} \leftarrow split(X_{train\_set})$ ▷ Split train dataset.

**for** episode=1,N **do**

  $a \leftarrow train\_loop(a, X_{train\_split}, X_{val\_split}, f_{train})$ ▷ Run train loop and retrieve trained agent $a$.

  eval_loop($a, X_{train\_set}, X_{val\_set}, f_{eval}$) ▷ Run evaluation loop on unseen network $f_{eval}$ using trained agent $a$.

**end for**




- Agent - PPO.
- Dataset - Cifar10.
- Sparsity:
  - Random Pruning, with Random Regrowth (**RP-RR**)
  - Magnitude Pruning, with Random Regrowth (**MP-RR**)

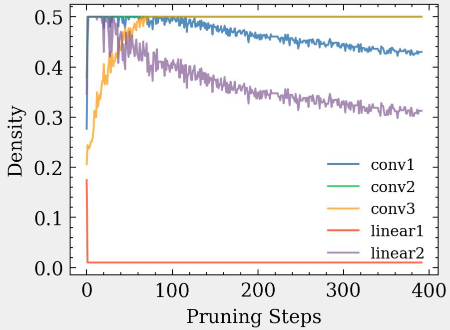
# Results - Simple CNN - 5 Layers

## 1. Learned Schedules are Competitive

Table 1: Test Accuracy (mean and standard deviation) of different schedules on CIFAR-10, using Simple-CNN.

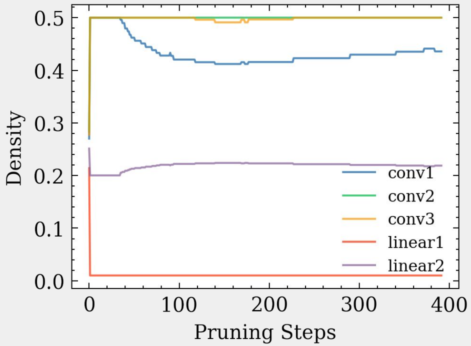
| Target Density (%) | Schedule | Random Pruning with Random Regrowth (RP-RR) | Magnitude Pruning with Random Regrowth (MP-RR) |
|---|---|---|---|
| 10 | Linear | 20.365 +- 17.952 | 60.418 +- 1.362 |
| | Quadratic | 23.15 +- 20.043 | 61.259 +- 1.485 |
| | Cubic | 33.721 +- 21.693 | 60.396 +- 0.832 |
| | Cosine | 18.302 +- 14.379 | 59.807 +- 0.384 |
| | Constant | **61.475 +- 0.731** | 62.536 +- 0.314 |
| | Learned (Ours) | 61.071 +- 1.574 | **63.191 +- 0.810** |
| 50 | Linear | 64.54 +- 0.477 | 64.78 +- 0.464 |
| | Quadratic | 64.987 +- 0.86 | 63.933 +- 0.431 |
| | Cubic | 65.31 +- 0.49 | 64.315 +- 0.437 |
| | Cosine | 64.672 +- 0.771 | 64.737 +- 0.345 |
| | Constant | 65.1 +- 0.283 | 65.388 +- 0.375 |
| | Learned (Ours) | **65.655 +- 0.515** | **65.686 +- 0.284** |
| 100 | Linear | 66.228 +- 0.691 | 66.711 +- 0.423 |
| | Quadratic | 66.947 +- 0.749 | 67.25 +- 0.578 |
| | Cubic | 66.857 +- 0.627 | 67.395 +- 0.547 |
| | Cosine | 66.074 +- 0.282 | 66.18 +- 1.027 |
| | Full Dense | **67.815 +- 0.146** | 67.878 +- 0.482 |
| | Learned (Ours) | 67.534 +- 0.174 | **67.908 +- 0.162** |

## 2. Learned Schedules are Layerwise Diverse




## 3. Learned a Handcrafted Schedule!

Piecewise Schedule for Random Pruning- [8].



# Results - ResNet18 & Conclusion

More **challenging** environment for our agent.

| Schedule | Test Accuracy |
|---|---|
| Linear | 93.019 +- 0.024 |
| Quadratic | 93.106 +- 0.107 |
| **Cubic** | **93.148 +- 0.156** |
| Cosine | 92.916 +- 0.105 |
| Constant (Fully Dense) | 92.481 +- 0.641 |
| Learned (Ours) | 92.818 +- 0.048 |

**Challenges:**

1. **Non-stationarity** environment.
   a. Our environment (the network we are learning a schedule for) is learning and adapting while our agent is learning to model the environment.
   b. Worse for challenging networks - use techniques like **data augmentation** and **learning rate decay** (e.g. ResNet-18).
2. **High dimension** action and (possibly) state space.
3. **Slow convergence** - 25-50 episodes.

**Conclusion:**

In this work, we demonstrate that it is **possible** to learn **well-performing dynamic sparsity schedules** using reinforcement learning. The schedules learned are not arbitrary and are distinct per layer and pruning method.

ICML Workshop Paper- **Workshop on Dynamic Neural Networks.**