Tiny Recursive Model

- Samsung built this new .7b param model beating 671 billion param model on tough puzzels and ARC-AGI (Benchmark for AGI)

Breakdown: Big AI models solve problems in one massive forward pass one chance only and a massive NN, this tiny model does it differently it takes a guess then recursively improves that guess up to 16 times each time its looking at 3 things: original problem, current answer and reasoning process then it refines all of them together. Just 2 layers recursing over and over beats 100 Layer massive transformers.

"Dropout" in Neural Networks

- Dropout is a regularization technique used in NN to prevent Overfitting. During training, dropout randomly "drops out" (ie sets to 0) a certian precentage of neurons in a layer on each forward pass, this means on each mini-batch the network temporarily removes some nodes and there connections

- Why: to make the network less reliant on specific neurons, each training iteration trains a slightly different "subnetwork". Because no single neuron can rely on others being present all the time, the model learns more robust, generalizable features. in short: if your Model gets too comfertable relying on certian neurons it stops learning broadly usefull features.

- How: if you have layer with activations $h=[h_1,h_2,h_3,h_4]$ and you apply dropout with rate $p=0.5$ (Dropout rate (p) is the probability of dropping a neuron common range is 0.1→0.5 higher = more dropout → more reg → less overfitting but can underfit if too high)

1) Randomly sample binary mask $m=[1,0,1,0]$ 2) compute new activations $\hat{h}=m\cdot h=[h_1, 0, h_3, 0]$

3) During inference (actually making predictions ie testing/deployment) we dont drop neurons but we scale activations by $(1-p)$. This keeps expected output consistant between training and inference

Ex: if 50% of neurons active on avg in train then $(p=.5)$, then at inference time we multiply outputs by 0.5 so total activation strenghth stays same