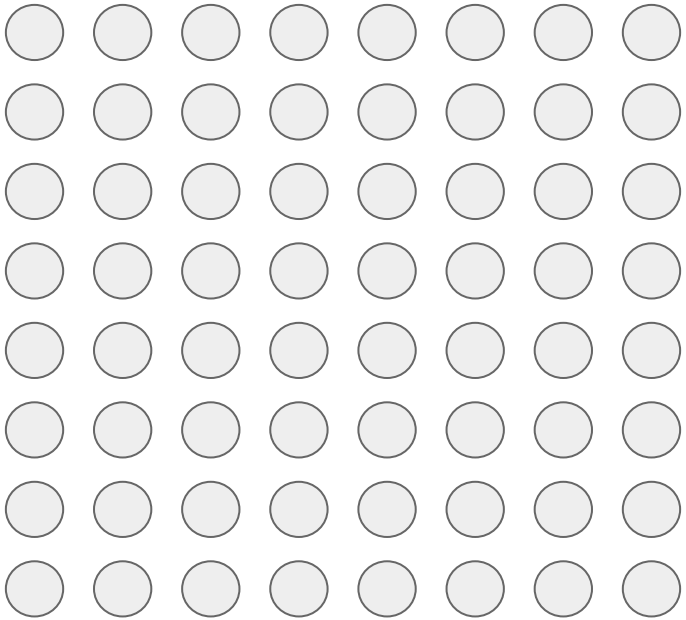
Mixture of experts

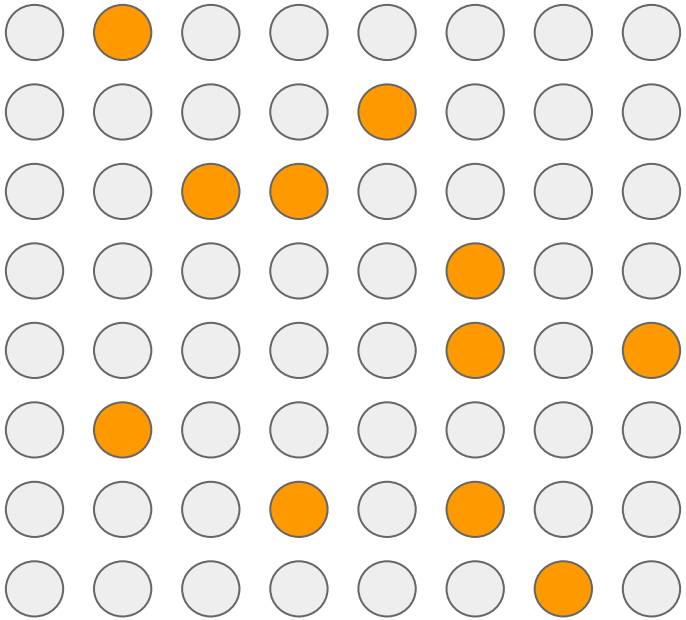
Trelis Research

Why MoE?



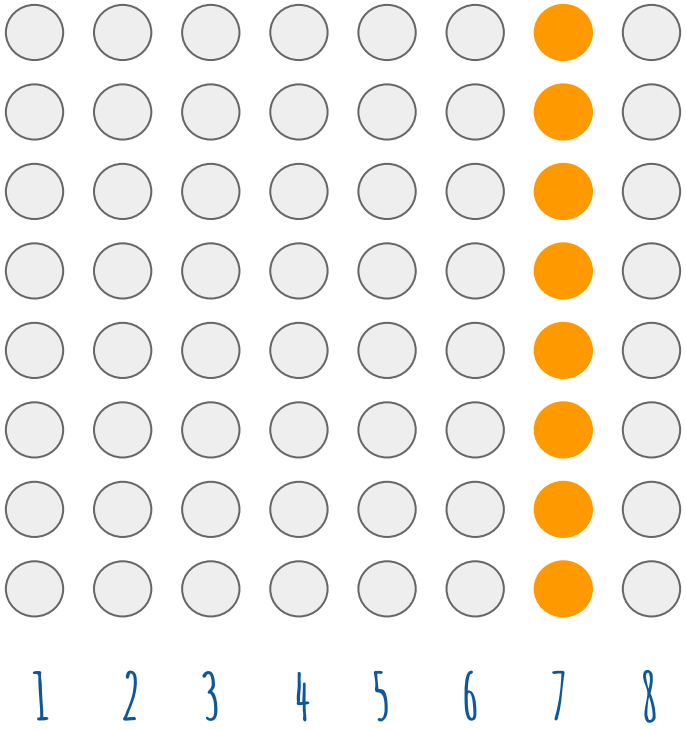
A traditional GPT uses all neurons in all matrices for forward passes

Why MoE?



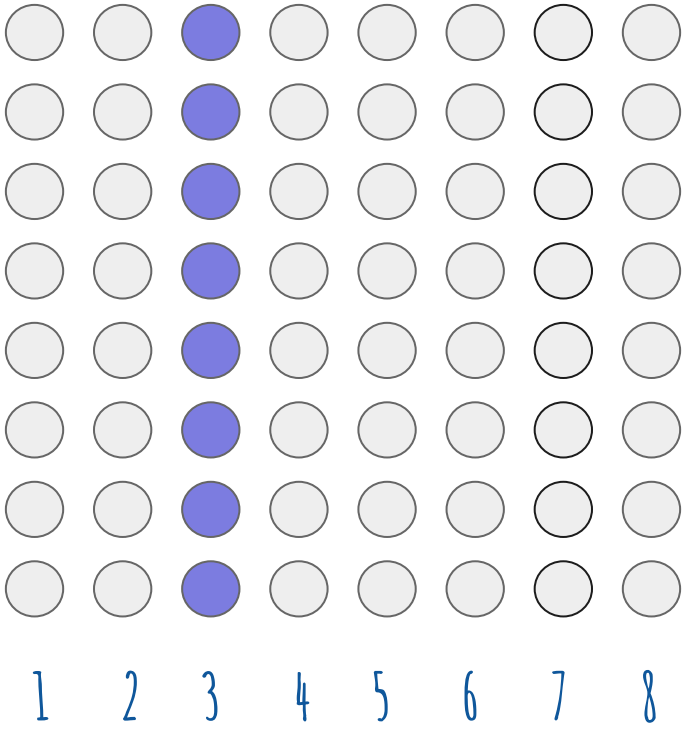
A traditional GPT uses all neurons in all matrices for forward passes

Why MoE?



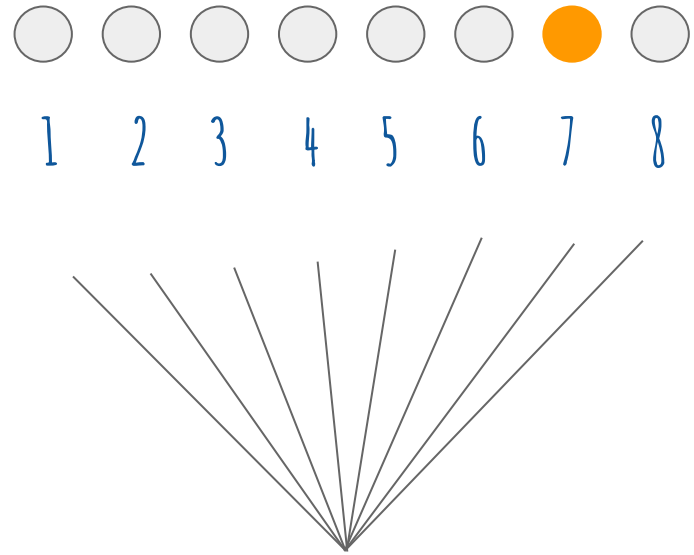
What if we could split the network + choose the best column of weights to use?

Why MoE?



What if we could split the network + choose the best column of weights to use?

How is MoE trained? Routers



[0,0,0,0,0,0,1,0]

Expert choice

Router Matrix weights

[0.34, 0.35, 0.73, 0.94]

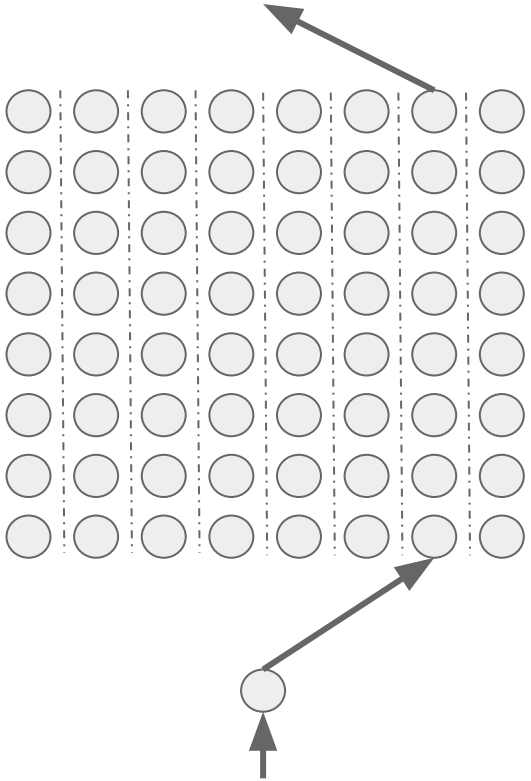
Input vector

TRAINING

OUTPUT TOKENS

LOSS = ACTUAL - PREDICTED

TRANSFORMER WEIGHTS

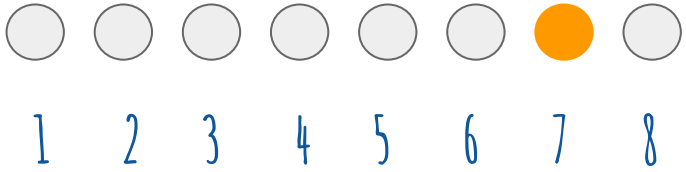


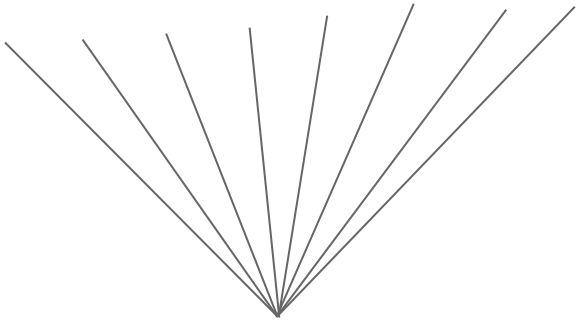
BACK PROPAGATION THROUGH GPT + ROUTER

ROUTER WEIGHTS

INPUTS

A problem training Mixture of experts





One expert can dominate!!!

router

Trick #1

[0,0,0,0,0,0,1,0]

Expert choice

↑ +randomness

Router Matrix weights

↑

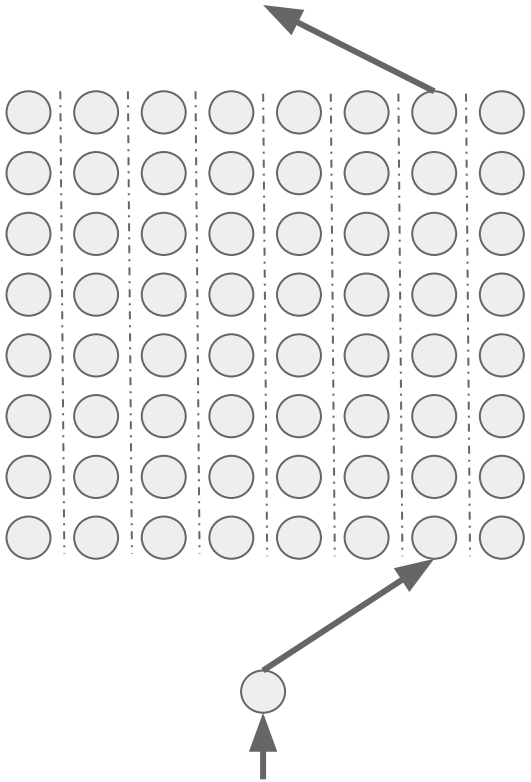
[0.34, 0.35, 0.73, 0.94]

Input vector

Trick #2

Transformer weights

Router weights



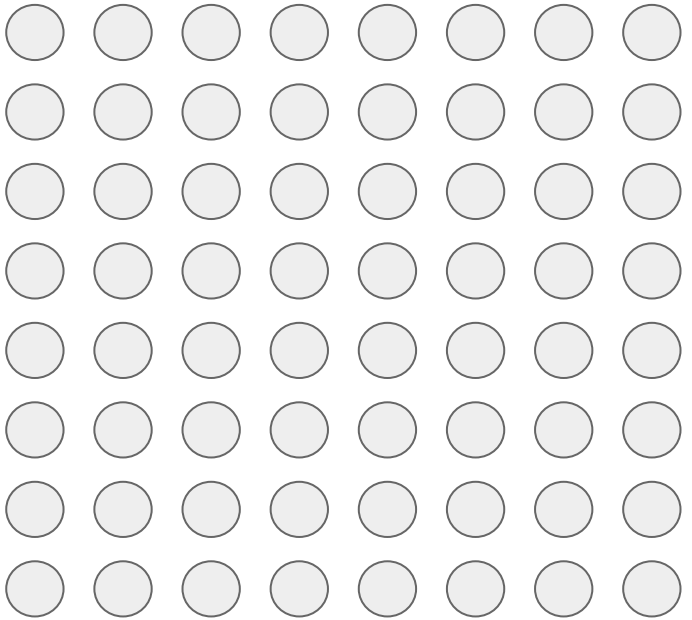
Loss = actual - predicted tokens

\+ Penalty for uneven router choice

Back propagation through GPT + router

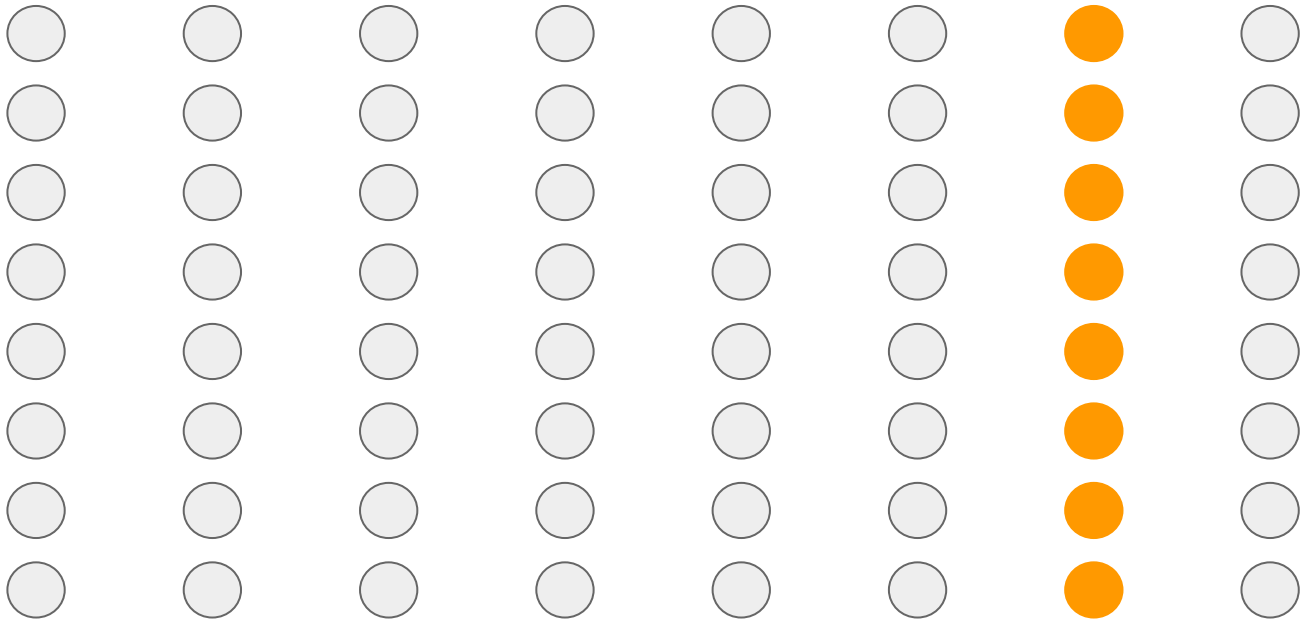
Is MoE useful for laptop inference?

MoE



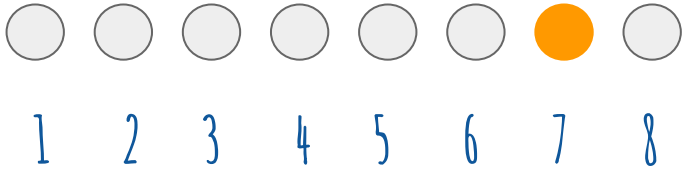
Standard GPT

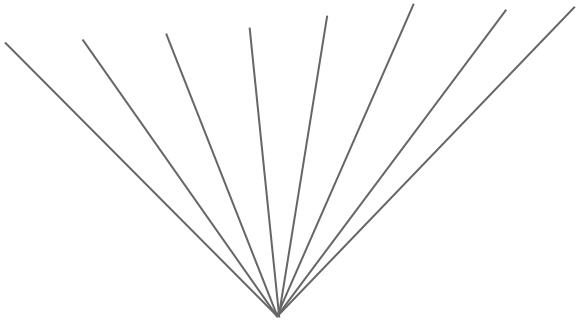
Does MoE reduce costs at scale?



Each expert gets its own GPU and queries are routed and batched

What are the problems with MoE

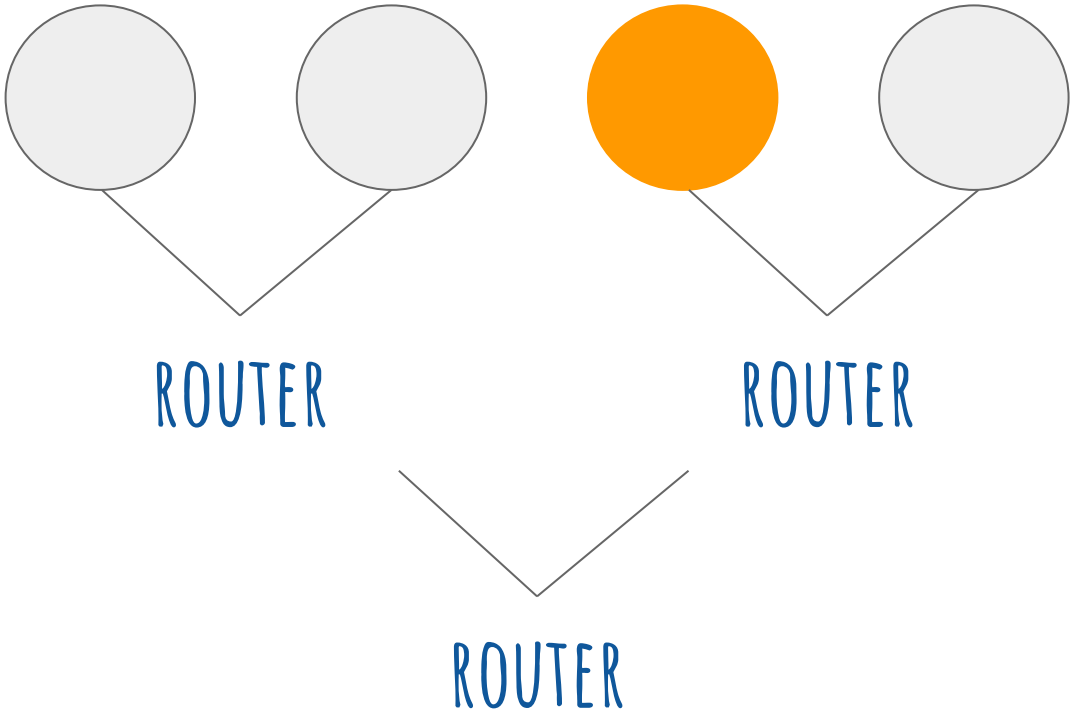




Slow to train experts evenly due to noise

router

Fast feed forward - better than MoE?



[1] or [0] with probability p

Expert choice

Router Matrix weights

[0.34, 0.35, 0.73, 0.94]

Input vector

Fast feed forward - better than MoE?

Binary tree networks allow:

- Same training time as GPT
- Faster inference

Fast feed forward - better than MoE?

Width	Model											
	feedforward				mixture-of-experts (e=16, k=2)				fast feedforward (ℓ=32)			
	M_A	ETT	G_A	ETT	M_A	ETT	G_A	ETT	M_A	ETT	G_A	ETT
$w = 64$	87.2	307	49.3	55	57.8	5354	29.4	4880	85.8	302	45.9	22
$w = 128$	95.5	200	51.5	46	62.0	6074	33.6	938	90.1	305	45.5	22
$w = 256$	99.9	105	52.0	48	62.4	2001	33.9	372	91.2	244	44.4	17
$w = 512$	99.9	85	52.4	31	65.4	3834	34.5	315	96.2	175	43.7	10
$w = 1024$	99.9	82	53.0	21	65.3	1575	35.2	327	96.0	180	41.3	9

Table 2: The results of the comparison of feedforward, mixture-of-experts, and fast feedforward networks, for various training widths. The inference width is fixed to 32 for mixture-of-experts and fast feedforward networks. The ETT columns to the right of metric columns list the "epochs to train", i.e. the number of training epochs that have elapsed until the score to the left was observed.

ETT = epoches to train. M_A=memorisation, G_A=generalisation

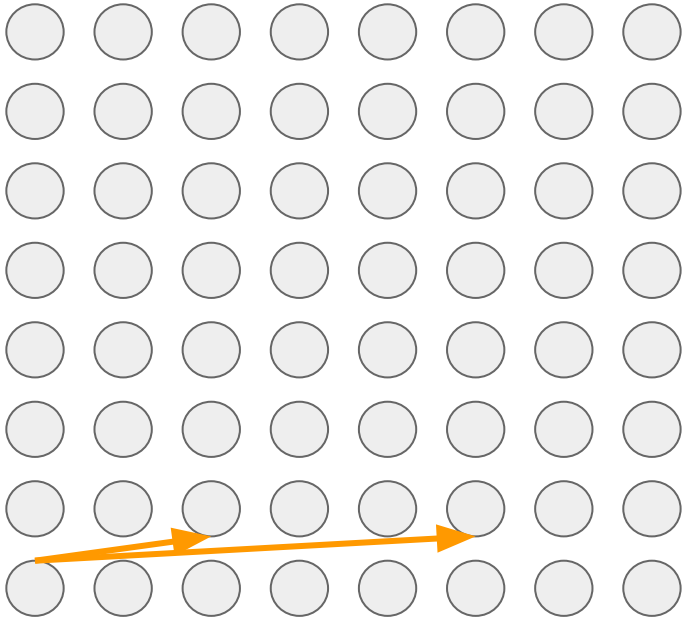
Fast feed forward - inference

Model		Property						
		depth	training width	training size	inference width	inference size	speedup	G_A
FF	$w = 128$	–	128	128 (100%)	128 (100%)	128 (100%)	1.00x	84.7
fast FF	$\ell = 32$	2	128	131 (102%)	32 (25%)	34 (27%)	2.44x	83.6
	$\ell = 16$	3	128	135 (105%)	16 (13%)	19 (15%)	2.80x	83.2
	$\ell = 8$	4	128	143 (112%)	8 (6%)	12 (9%)	3.29x	82.8
	$\ell = 4$	5	128	159 (124%)	4 (3%)	9 (7%)	3.39x	81.6
	$\ell = 2$	6	128	191 (149%)	2 (1%)	8 (6%)	3.47x	80.1
	$\ell = 1$	7	128	255 (199%)	1 (1%)	8 (6%)	3.93x	79.8

Table 3: The results of the testing of vision transformers leveraging feedforward and fast feedforward layers. All sizes are given in neurons. Bracketed percentages describe quantities relative to their counterparts in the vanilla feedforward layers. G_A is the generalization accuracy of the fully trained vision transformer and "speedup" gives the performance improvement over vanilla feedforward layers in our testing setup.

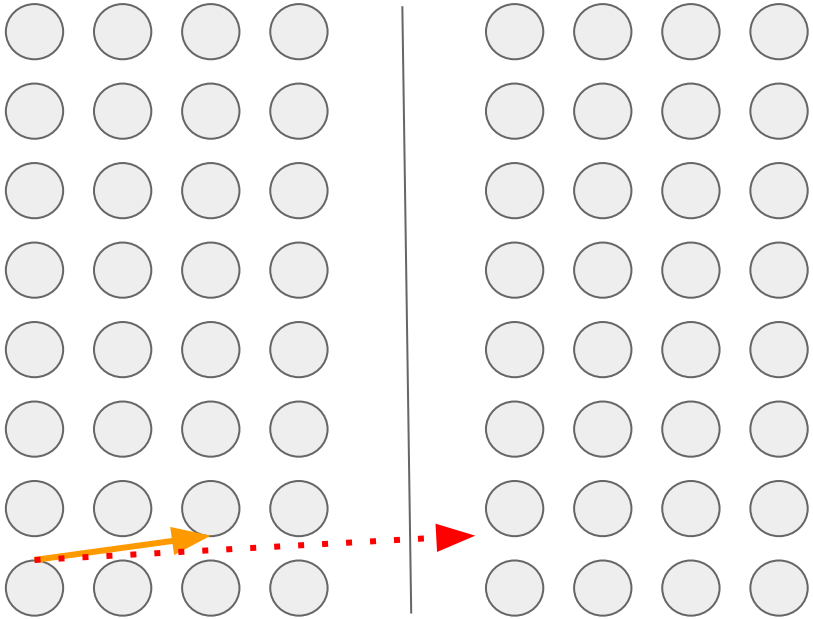
Caveat - the above is a very small neural net for vision.

WHY DOES MOE (OR FFF) WORK?



STANDARD GPT

Why does MoE (or FFF) make sense?



MoE/FFF

Same model size

Fewer connections

Small accuracy loss

~1/N speed gain

Papers and links

Binary-tree/FFF paper: https://arxiv.org/pdf/2308.14711.pdf

MoE papers: https://arxiv.org/pdf/2208.02813.pdf; https://arxiv.org/pdf/1701.06538.pdf

Reddit thread: https://tinyurl.com/ythsu2nd

Youtube video: https://youtu.be/0u_65floTq0

Trelis.com