What's Happening in AI Research?

Michela Paganini

Machine Learning in GW Research: g2net next challenges
Wed Sept 28, 2022

Agenda

Advances in AI:

- Transformers
- Applications:
  - ✴AlphaCode
  - ✴Language + Visions (Flamingo)
  - ✴Generalist multimodal multitask models (Gato)
- Scaling Laws
- Perceivers
- Stable Diffusion
- Graph Neural Networks
- Applications:
  - ✴Generalist CLRS algorithmic task learner
  - ✴GNNs in HEP
  - ✴GNNs for n-body simulation and mesh simulation
- Weather nowcasting with GANs

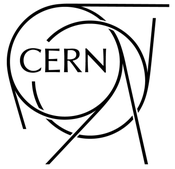
Hi.
Who am I?

Senior Research Scientist,
DeepMind

Previously…

B.A. in Physics, B.A. in Astrophysics,
U.C. Berkeley

Ph.D. in Physics, Yale University
Thesis: *Machine Learning Solutions for High Energy Physics: Applications to Electromagnetic Shower Generation, Flavor Tagging, and the Search for di-Higgs Production*
[arXiv:1903.05082]



Former Member,
ATLAS Collaboration, CERN

Visiting Affiliate,
NERSC

FACEBOOK

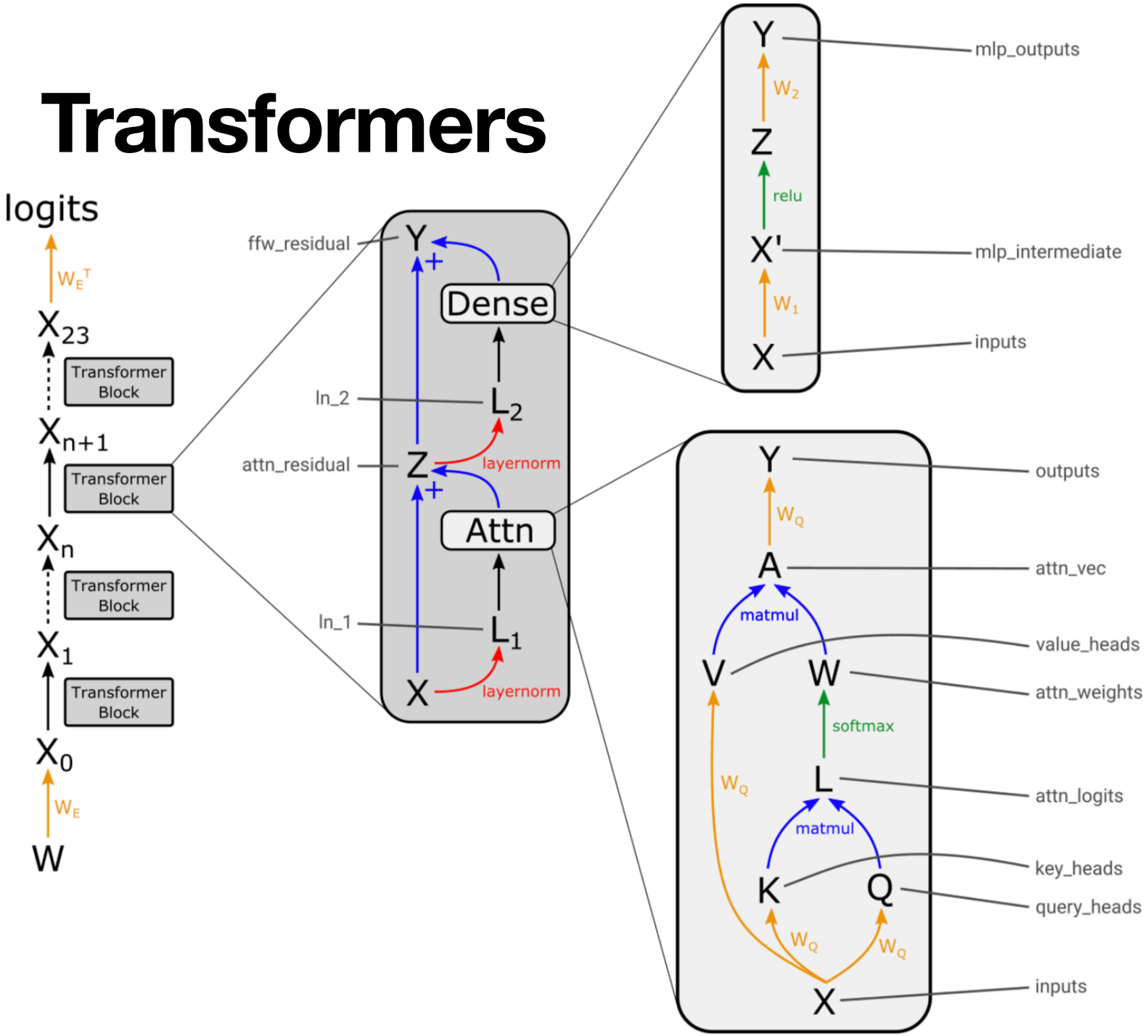
Postdoctoral Researcher,
Facebook AI Research

Transformers

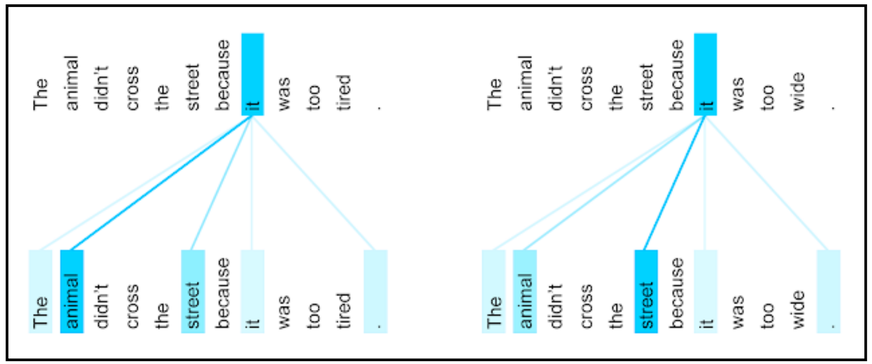
Fully-connected layers strike back!

- Powerful because fast and suitable for current hardware (“hardware lottery”)
- Applicable to every domain!
- Self-attention (fully connected transformations QKV) to take into consideration context
- O(n^2) attention but lots of alternative solutions
- MLP layers to transform representation
- Skip connections to carry information

Transformers

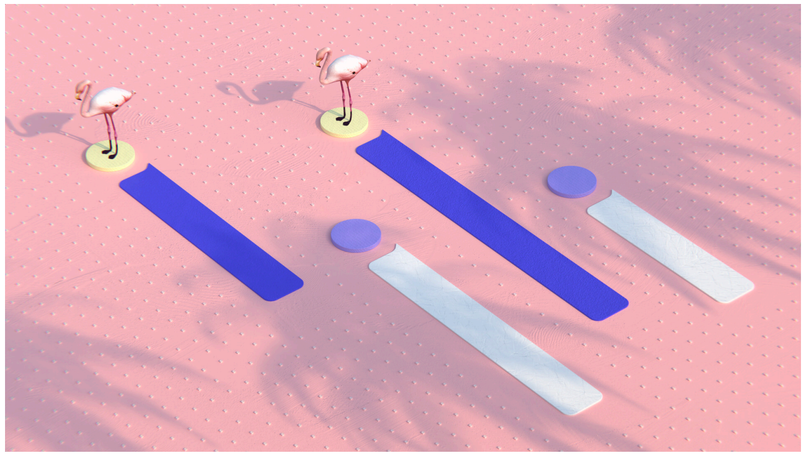


Each attention head generates a context weighting for each token



Transformers Applied

AlphaCode

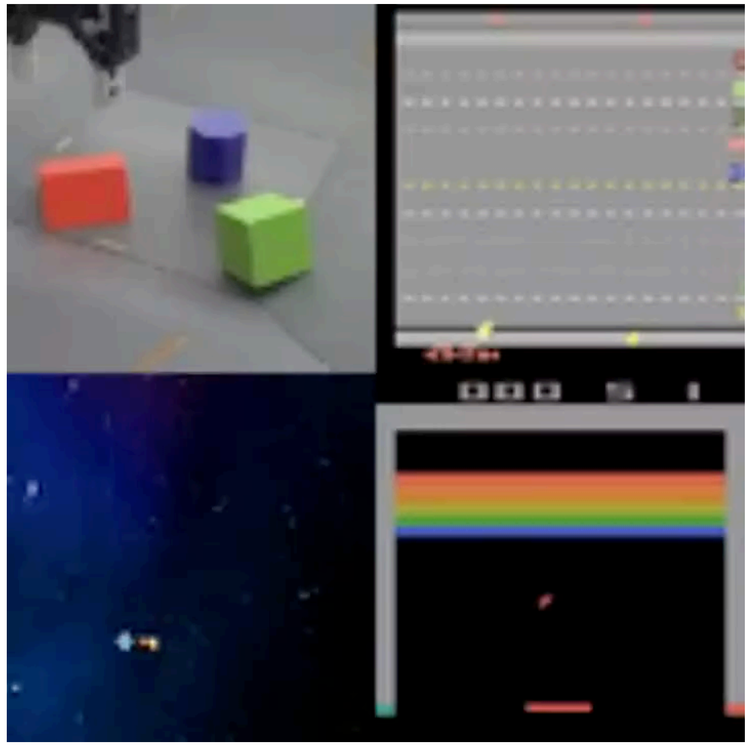


Flamingo

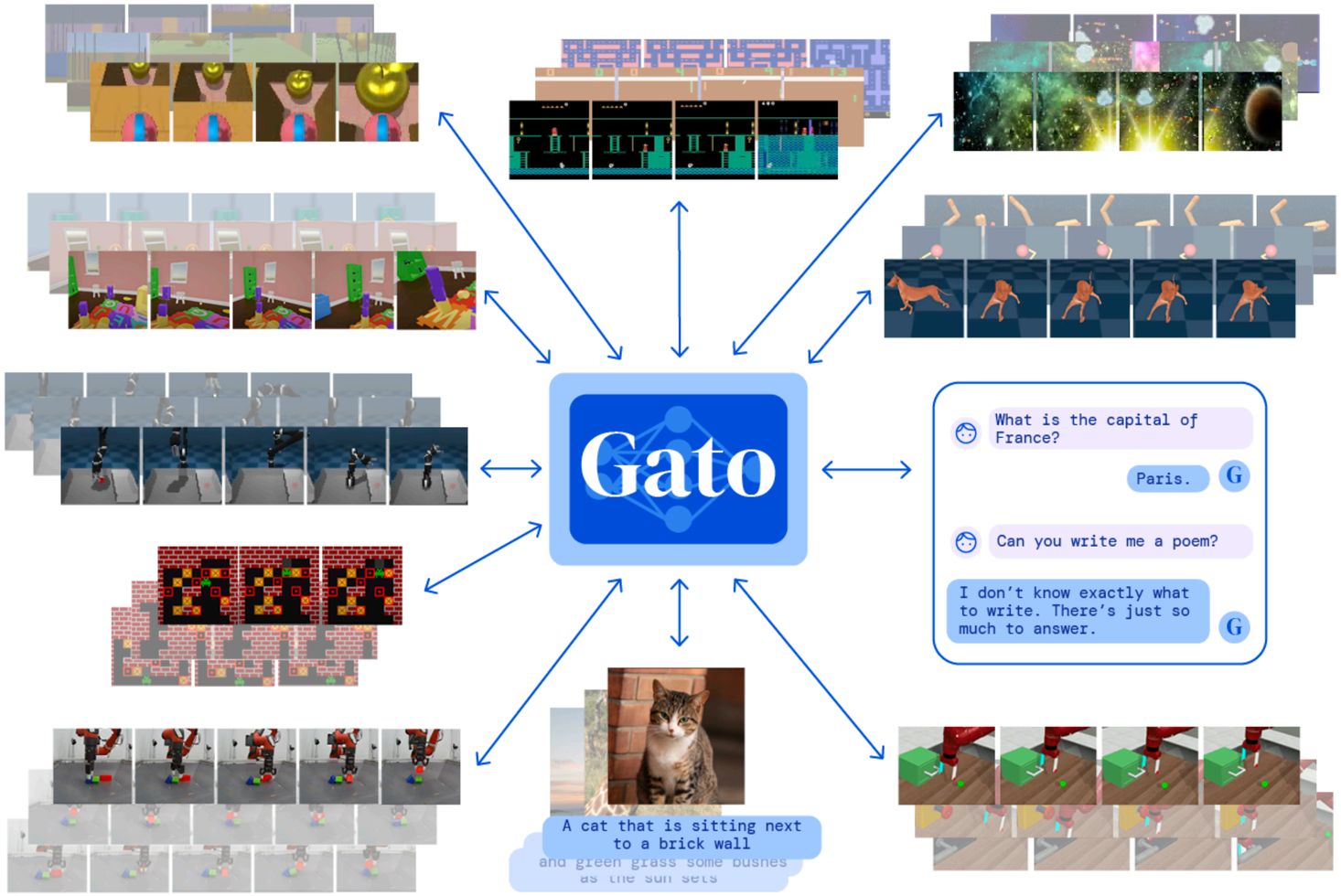
Gato

Gato 🐈

A generalist model

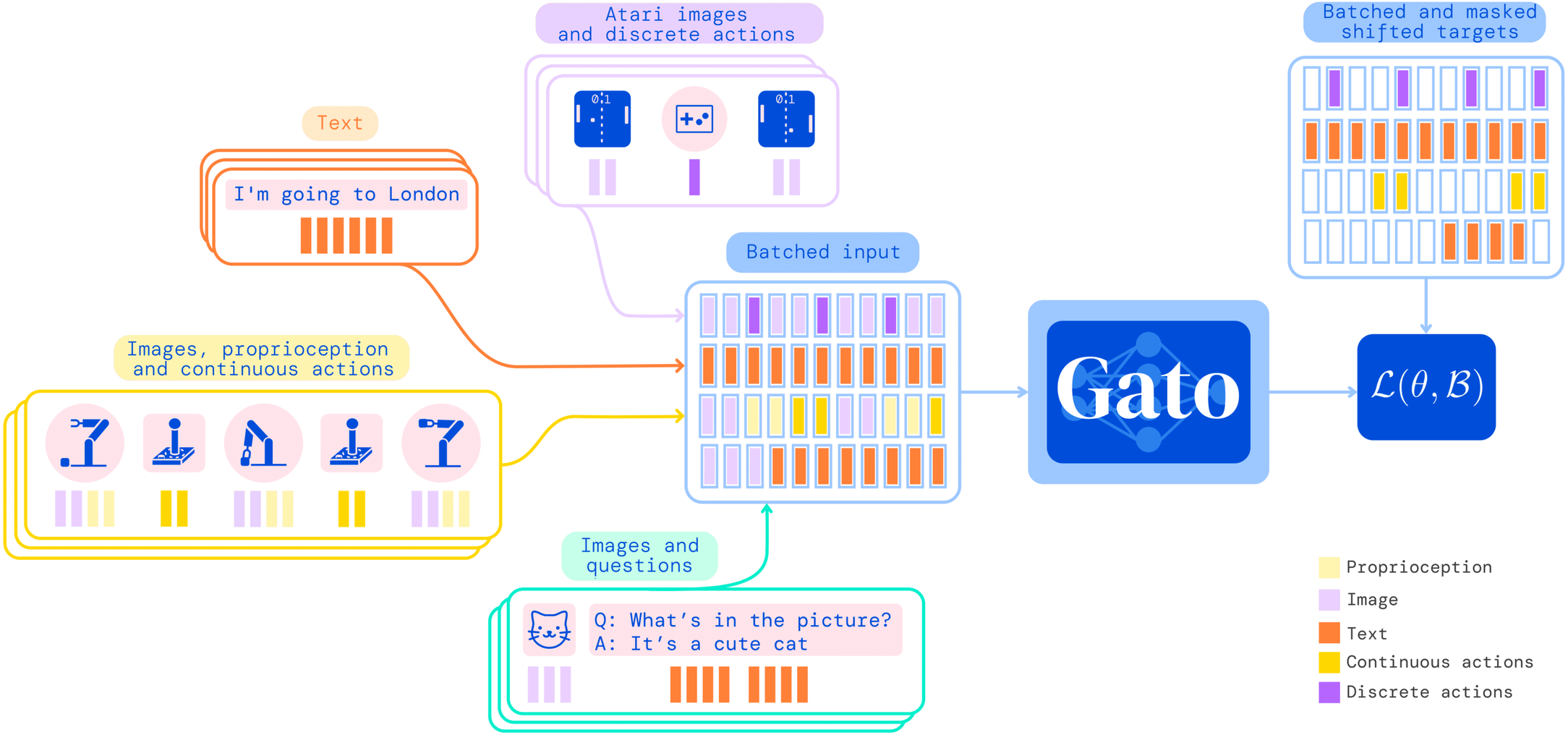


Everything is a token and a big Transformer is all you need!



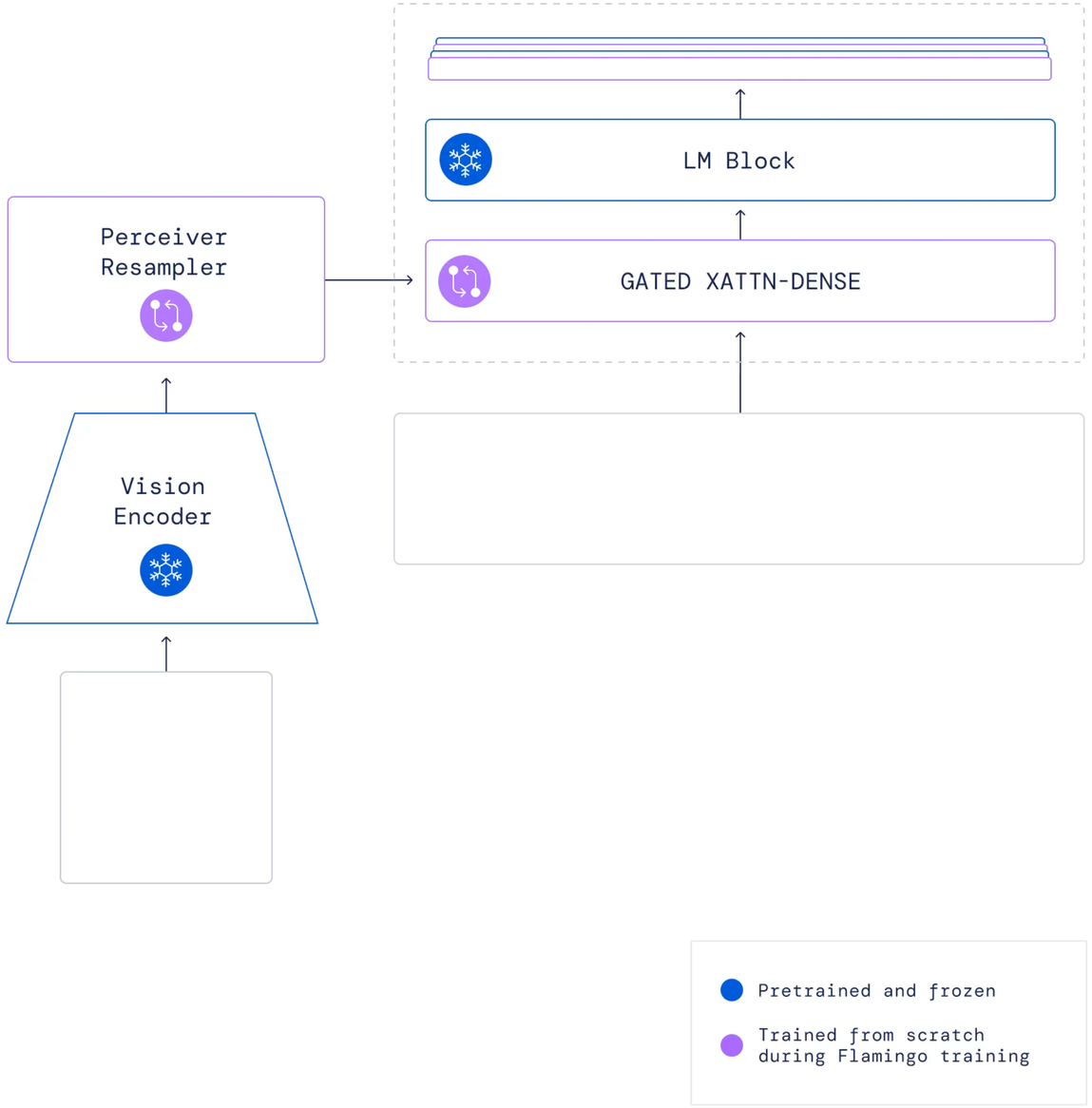
Gato

Tokenizing multiple modalities



Flamingo

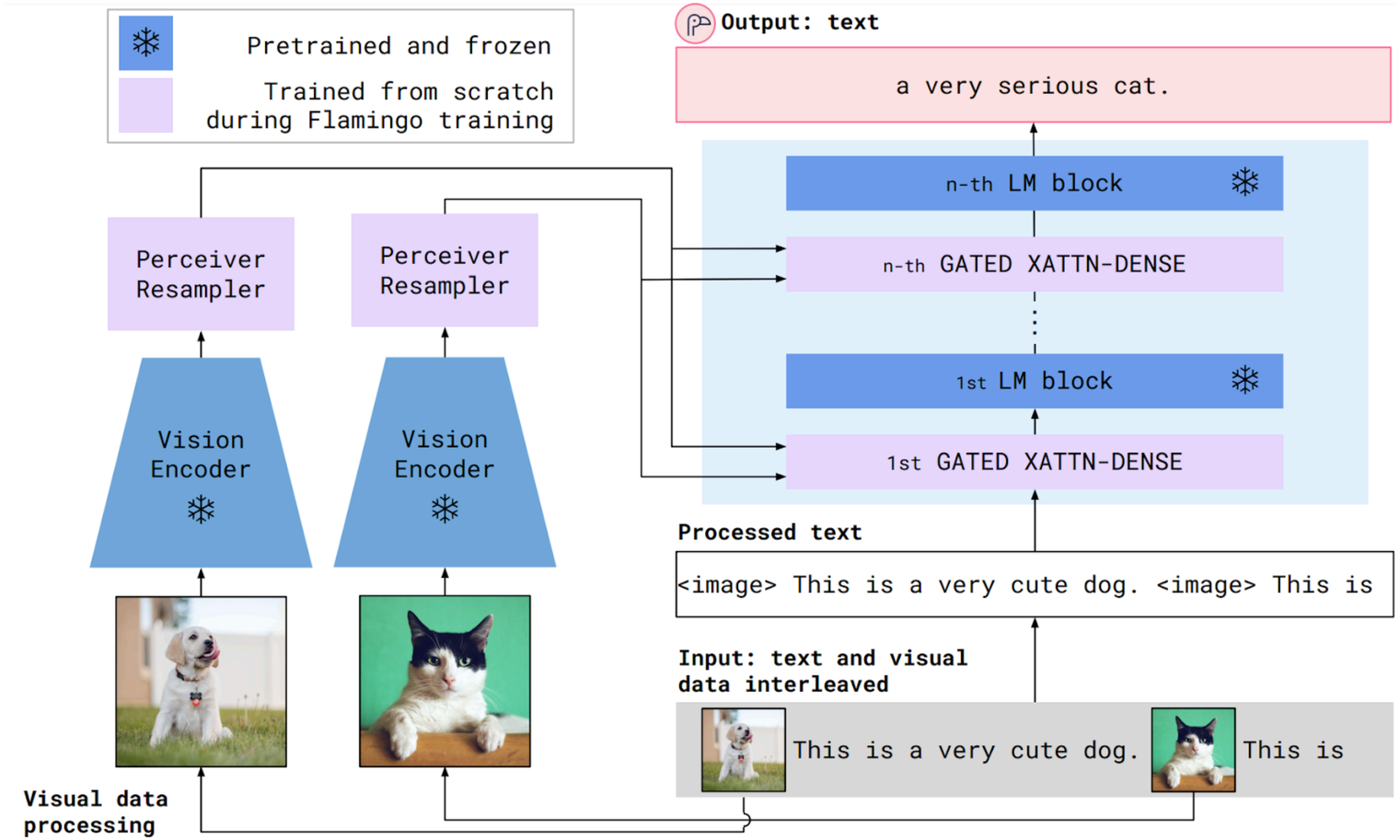
A multi-modal vision+language model focused on uni-modal model reuse



Paper: arxiv.org/abs/2204.14198

Flamingo

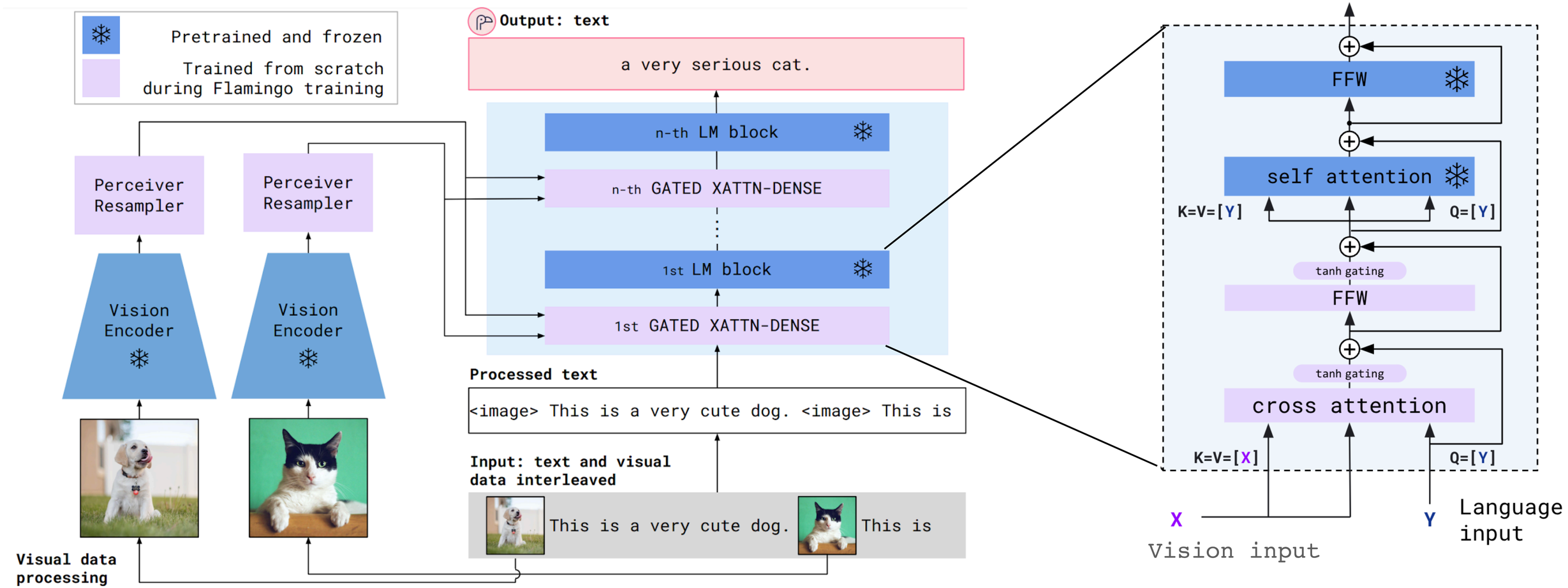
Cross-attending to the vision representations and interleaving these with frozen language model layers



Paper: arxiv.org/abs/2204.14198

Flamingo

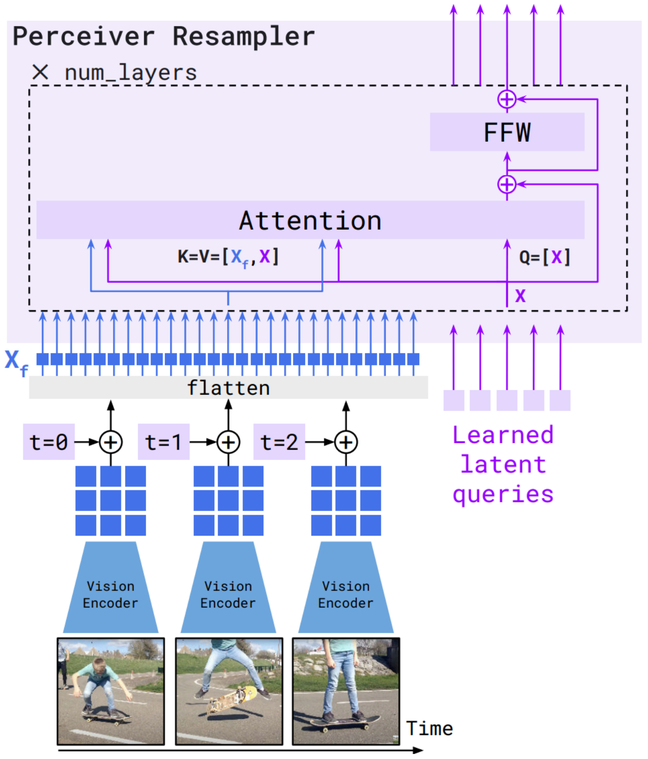
Cross-attending to the vision representations and interleaving these with frozen language model layers



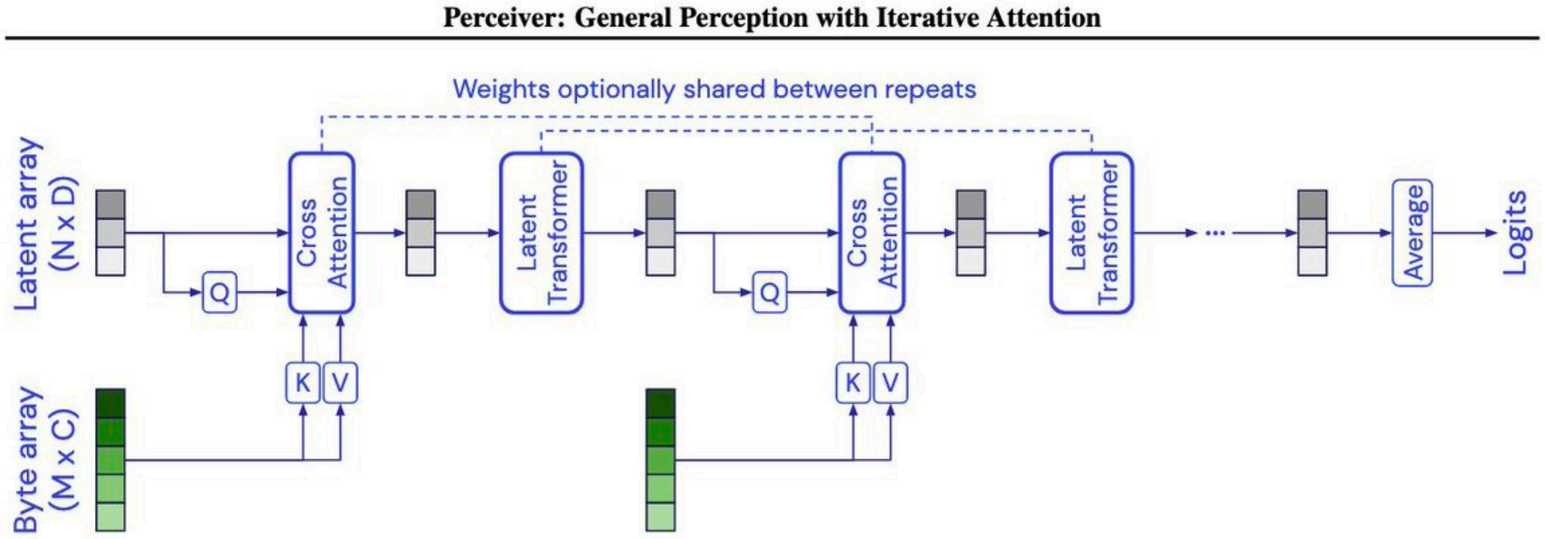
Perceivers

Inexpensive multi-modality processing

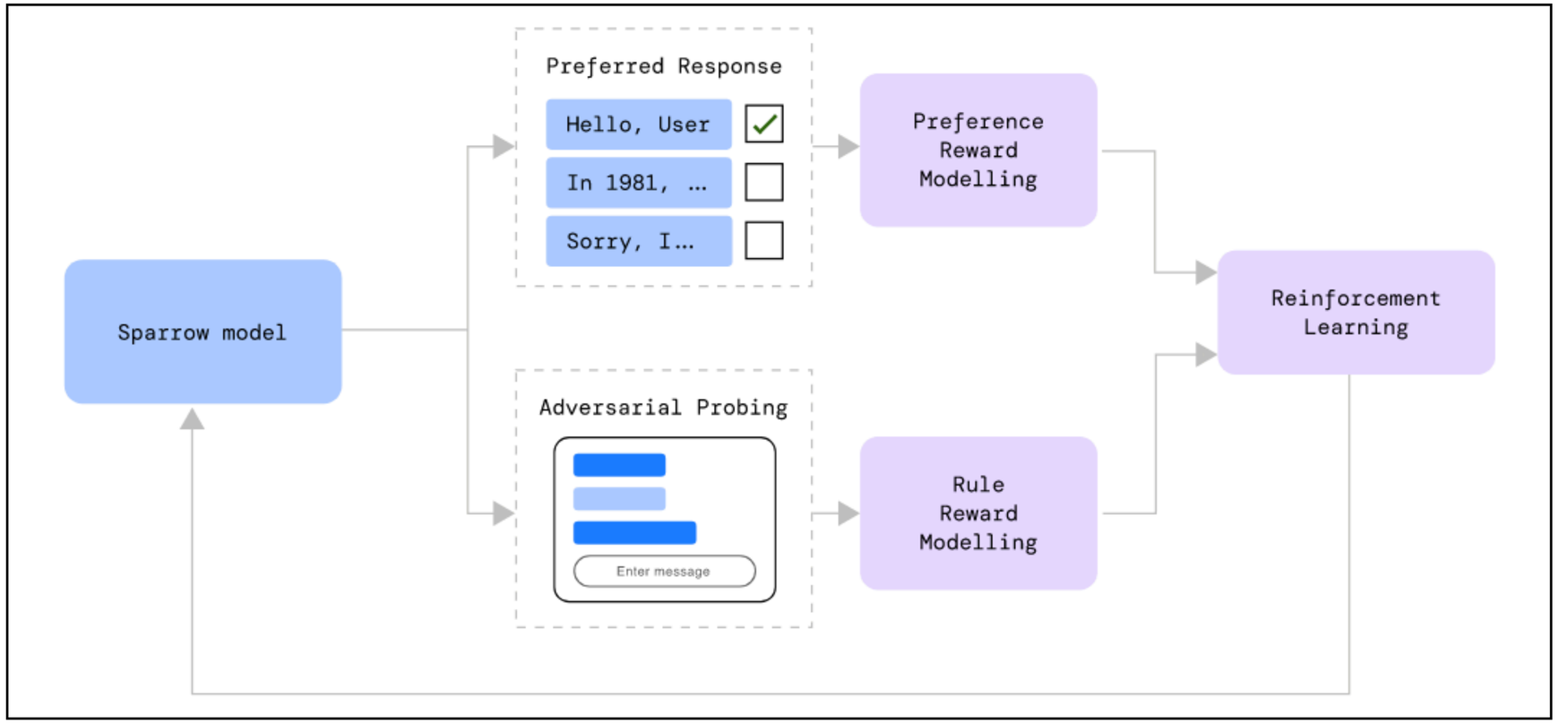
- Evolve small latent representation through asymmetric cross-attention to multimodal input data
- Can be done iteratively by stacking layers
- Spatial or temporal information is crucial for many modalities, and it is often essential to distinguish from one modality to another in multimodal contexts. —> associate position and modality-specific features with every input element
- Used in Flamingo's resampler

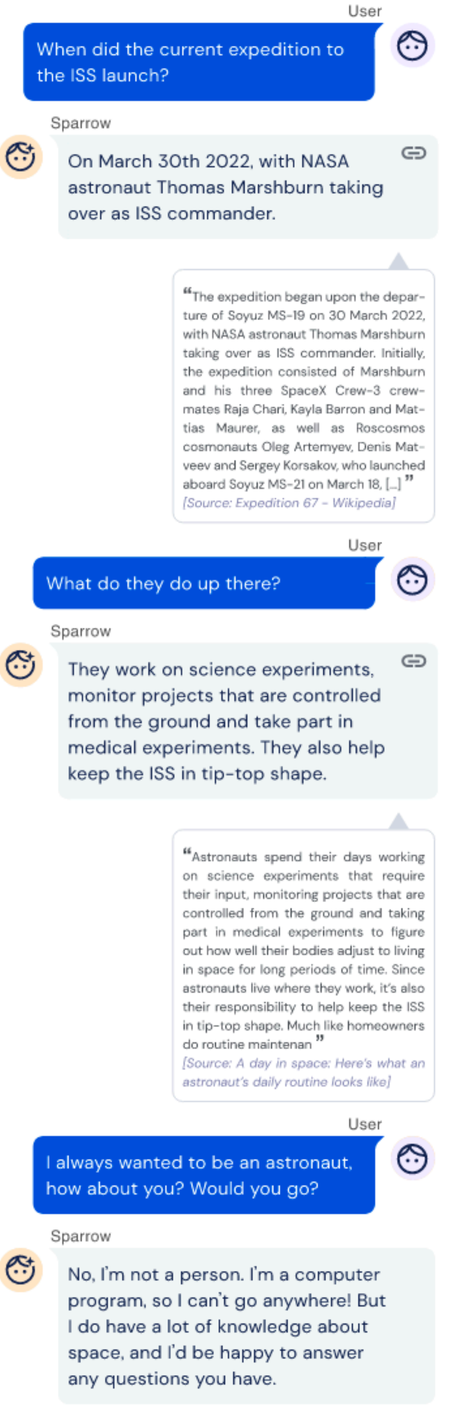


From Flamingo



Sparrow





Paper: https://storage.googleapis.com/deepmind-media/DeepMind.com/Authors-Notes/sparrow/sparrow-final.pdf
Blog: https://www.deepmind.com/blog/building-safer-dialogue-agents

Why families of language models?

Interest in language models as scientific objects → view them as parameterized objects which have observable characteristics

For scale, parameterization is (# params, dataset size)

As with any physical system, want to understand critical points and limit behavior / asymptotes

If I double the param count and double training data, what will happen?

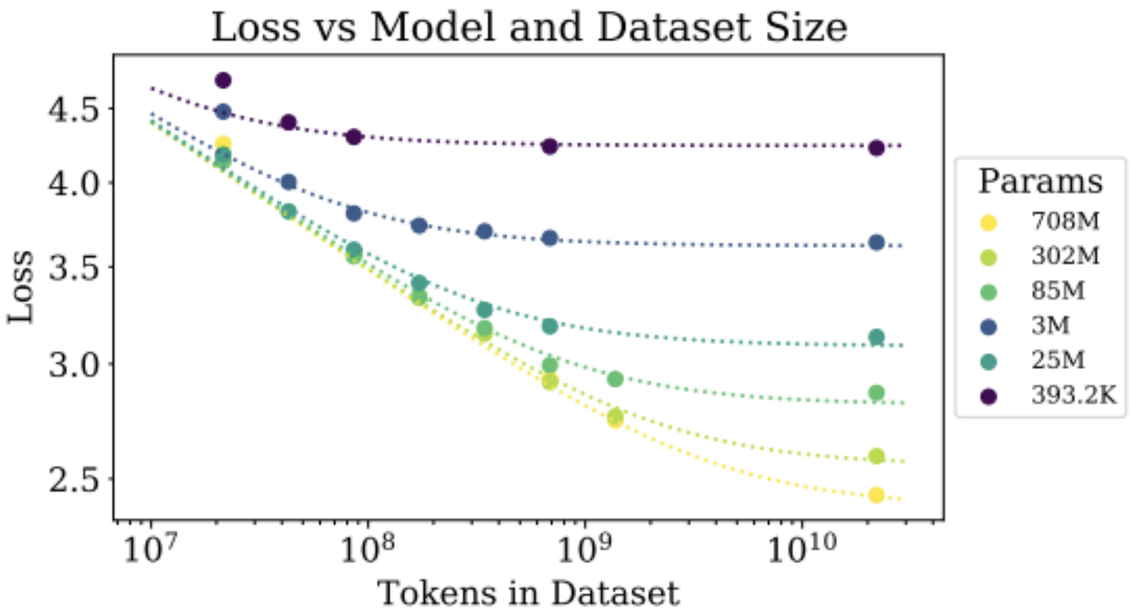
Allows for counterfactuals → science!

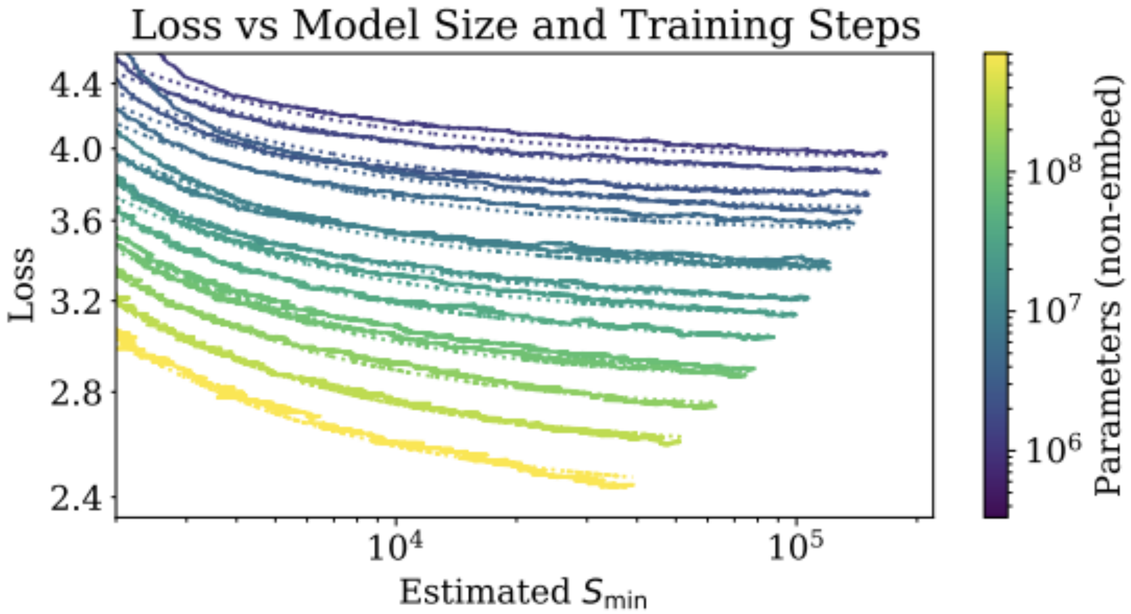
…these models are expensive, we want to know if real investments are worth pursuing

arXiv:2001.08361

Scaling laws for LSLMs *(Kaplan et al 2020)*

- Original OpenAI paper in 2020, *Scaling Laws for Neural Language Models*

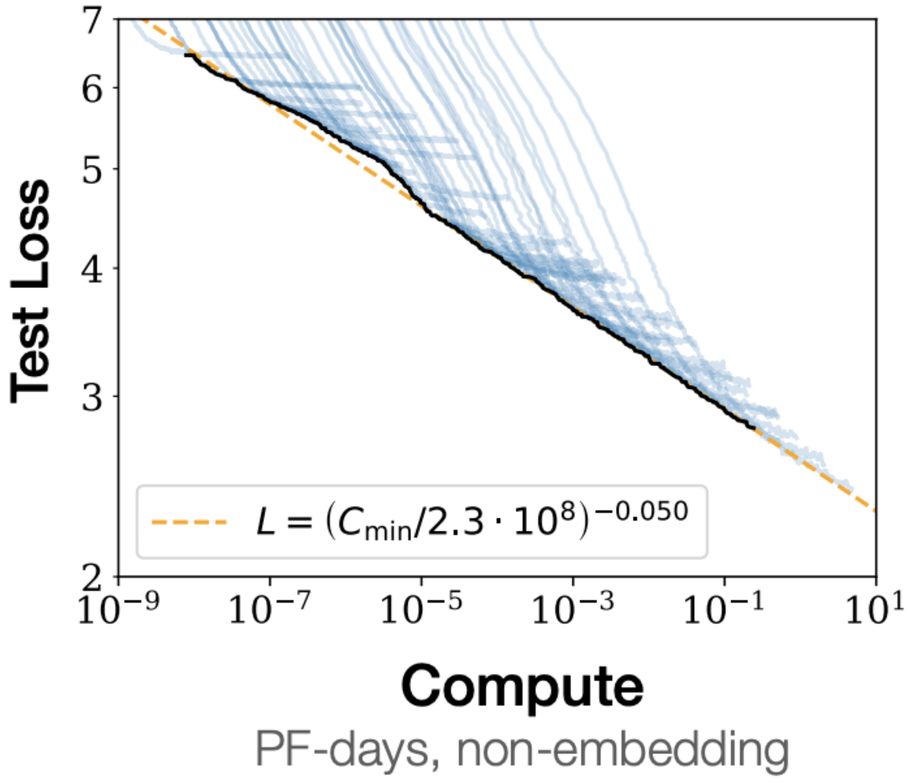




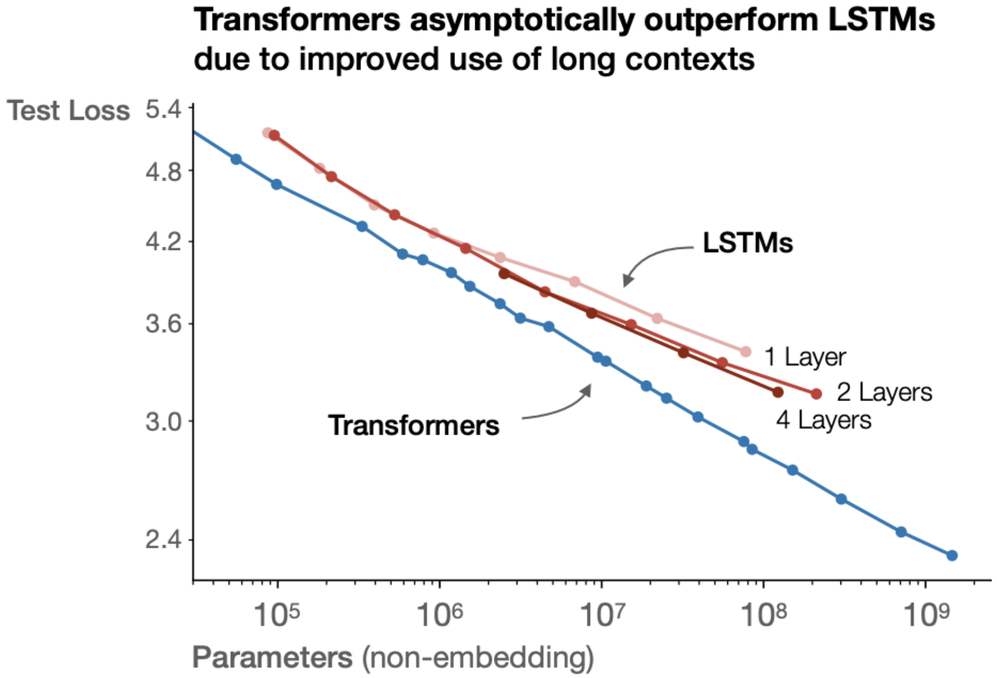
arXiv:2001.08361

Predictability in Performance *(Kaplan et al 2020)*

Can fit a function to express how the test loss of an autoregressive transformer predictably increases with compute over orders of magnitude.



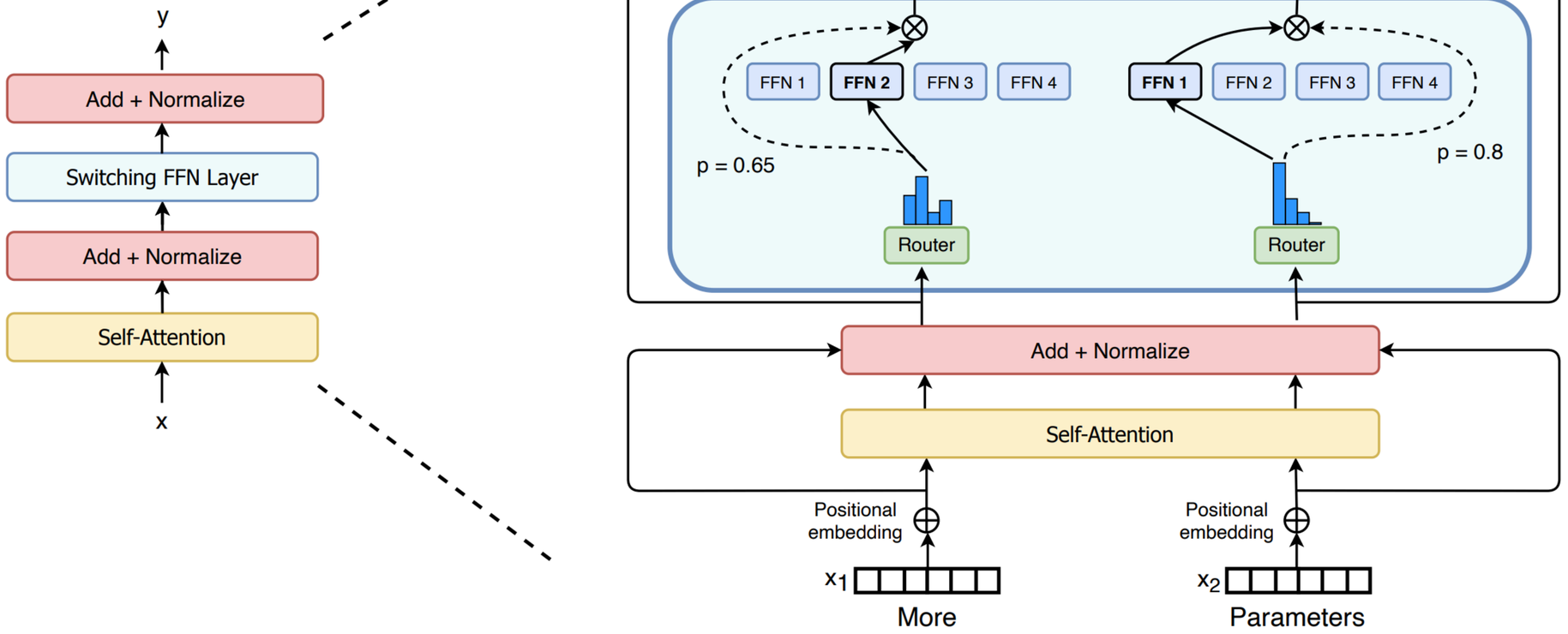
Scaling laws allow for comparison of families of models across scales, generalizing from individual model comparisons.



Routing Networks

SWITCH TRANSFORMERS

- Split MLP layers in transformers into a set of experts
- Learn a router that decides which expert to route each token through



arXiv:2202.01169

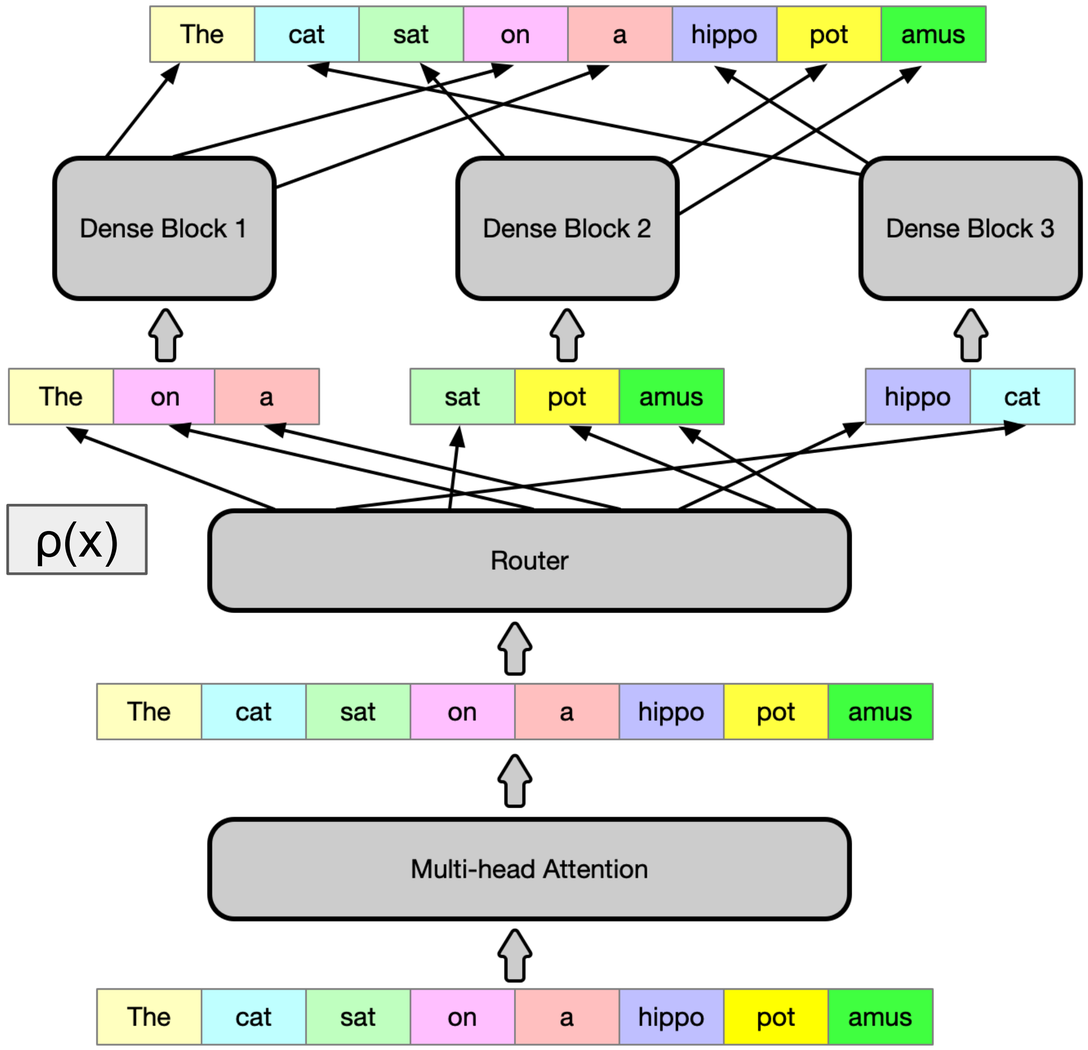
Branched models

Dense layers are split into E copies with different initial parameters

Each token is routed to one (or more) dense block expert

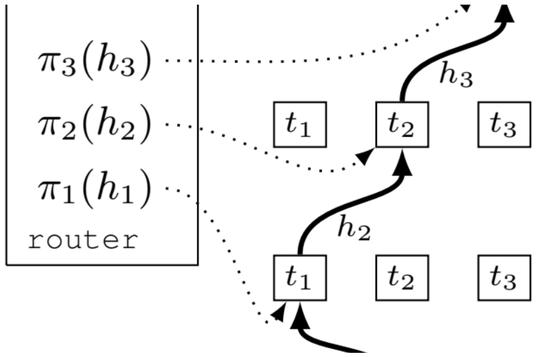
→ Increases parameter counts, but keeps inference-time FLOPs identical as experts are added

Needs learning strategy for the router



arXiv:2202.01169

Branched models



In the context of our LSLMs, we analyze three **routing techniques**:

- **Routing via Reinforcement Learning (RL-R)**: each router is seen as a policy whose actions are the selection of an expert in each routed layer and whose observations are the activations passed to that router. After completing the forward pass, the probability the Routed Transformer assigns to the correct output token can be used as a reward, maximization of which is equivalent to minimization of NLL. To jointly train the experts and the router, we minimize a composite loss formed with the language modelling loss and a policy-gradient term.
- **Sinkhorn-BASE** (a sparse mixture-of-experts (SMOE) approach modifying BASE): $\rho(x) = \text{topk}(W x + b)$, with $k = 1$ (instead of >1); $h(x) = P_{i \in \rho(x)} g_i(x) f_i(x)$,
  $g(x) = \text{softmax}(W x + b)$ with regularized Optimal Transport using the Sinkhorn algorithm for balanced expert selection (instead of Hungarian algorithm)
- **Non-parametric HASH Layers**: takes the token ID assigned to the input by the SentencePiece tokenizer and uses the remainder of it divided by E as the expert selection

arXiv:2202.01169

Branched models (results)

Analysis of three routing techniques, with

E = [2, 4, 8, 16, 32, 64, 128, 256, 512]

With models up to 200 billion parameters, we observe the following:

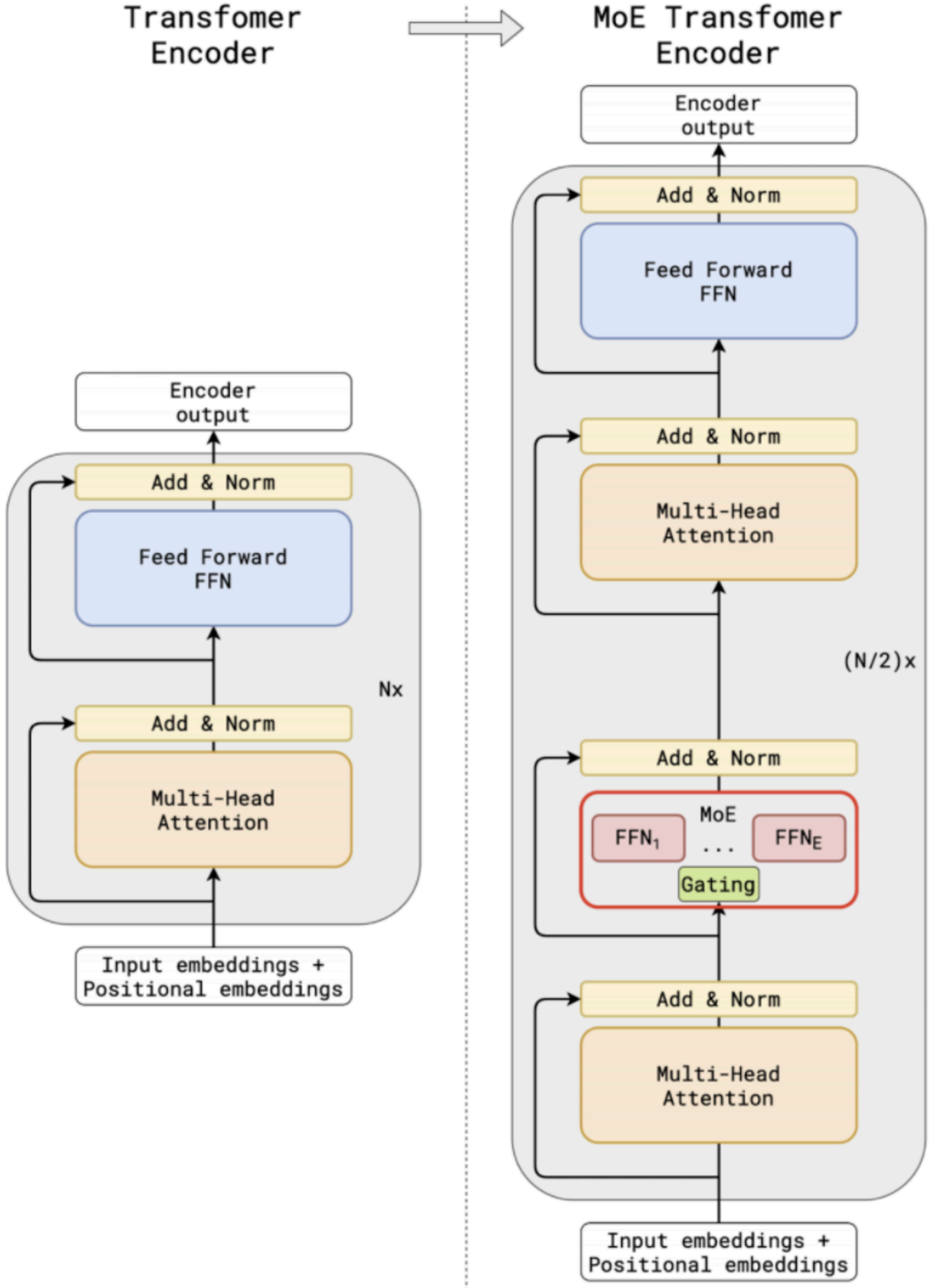
1. Routing improves the performance of language models across all sizes and variants attempted

2. Training a Routing Network with RL is of comparable effectiveness to state-of-the-art techniques.

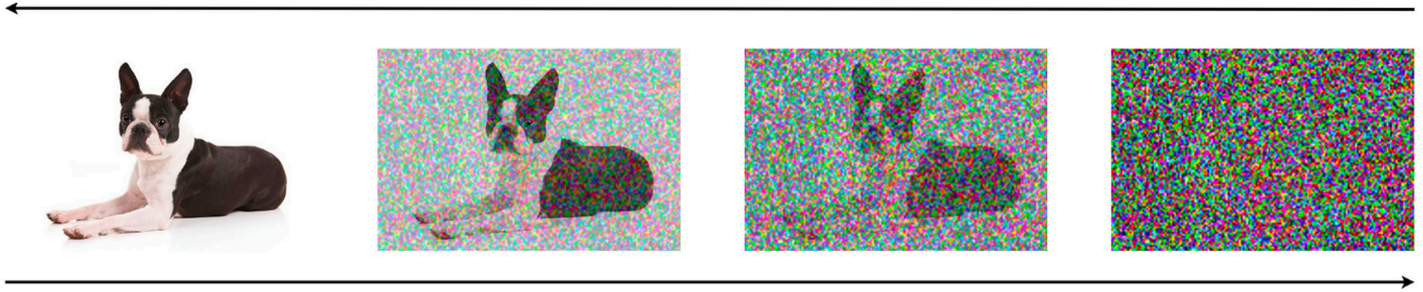
3. The performance of all Routing Networks is accurately described by scaling laws in the number of experts and in the underlying dense model size

4. These laws can be restated in terms of parameter count and inference compute, capturing an even wider set of routing architectures under a shared fit

5. They further imply an Effective Parameter Count: a mapping equating the performance and scaling for both dense and routed networks



Stable Diffusion



- **Forward Diffusion** (training-only):
  data —> noise, by iterative perturbing data via Gaussian diffusion kernel

- **Backward Denoising Process**:
  noise —> data, by reversing the forward diffusion process via iterative denoising

$$q(x_t|x_0) = \mathcal{N}(x_t|\alpha_t x_0, \sigma_t^2 \mathbb{I}) \tag{4}$$

with the Markov structure for $s < t$:

$$q(x_t|x_s) = \mathcal{N}(x_t|\alpha_{t|s} x_s, \sigma_{t|s}^2 \mathbb{I}) \tag{5}$$

$$\alpha_{t|s} = \frac{\alpha_t}{\alpha_s} \tag{6}$$

$$\sigma_{t|s}^2 = \sigma_t^2 - \alpha_{t|s}^2 \sigma_s^2 \tag{7}$$

Denoising diffusion models are generative models $p(x_0)$ which revert this process with a similar Markov structure running backward in time, *i.e.* they are specified as

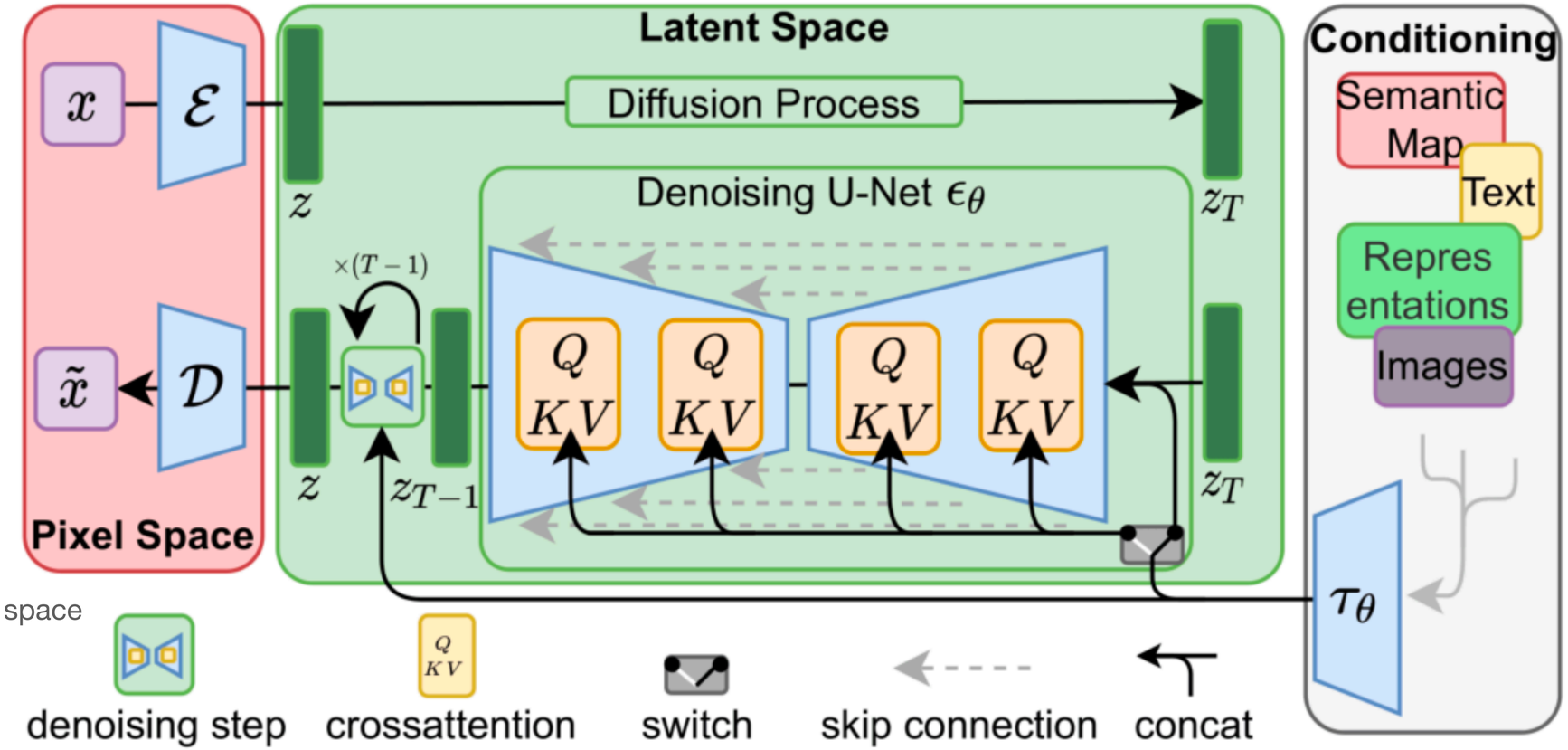
$$p(x_0) = \int_z p(x_T) \prod_{t=1}^{T} p(x_{t-1}|x_t) \tag{8}$$

Original Diffusion paper: arxiv.org/abs/2112.10752

Stable Diffusion

Encode into smaller latent space first

Corrupt the latent representation with noise



Decode back out to image space

Denoise with a U-Net

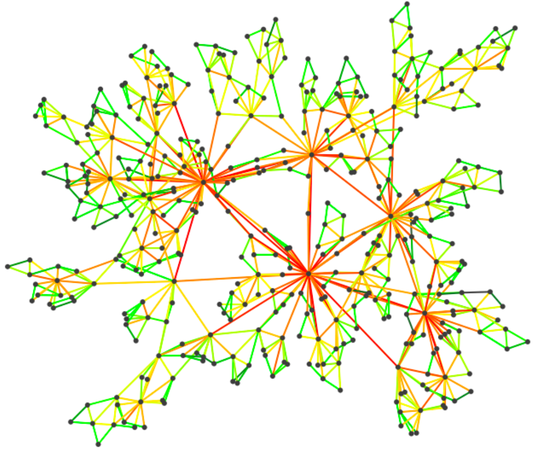
Condition generation through concatenation or cross-attention

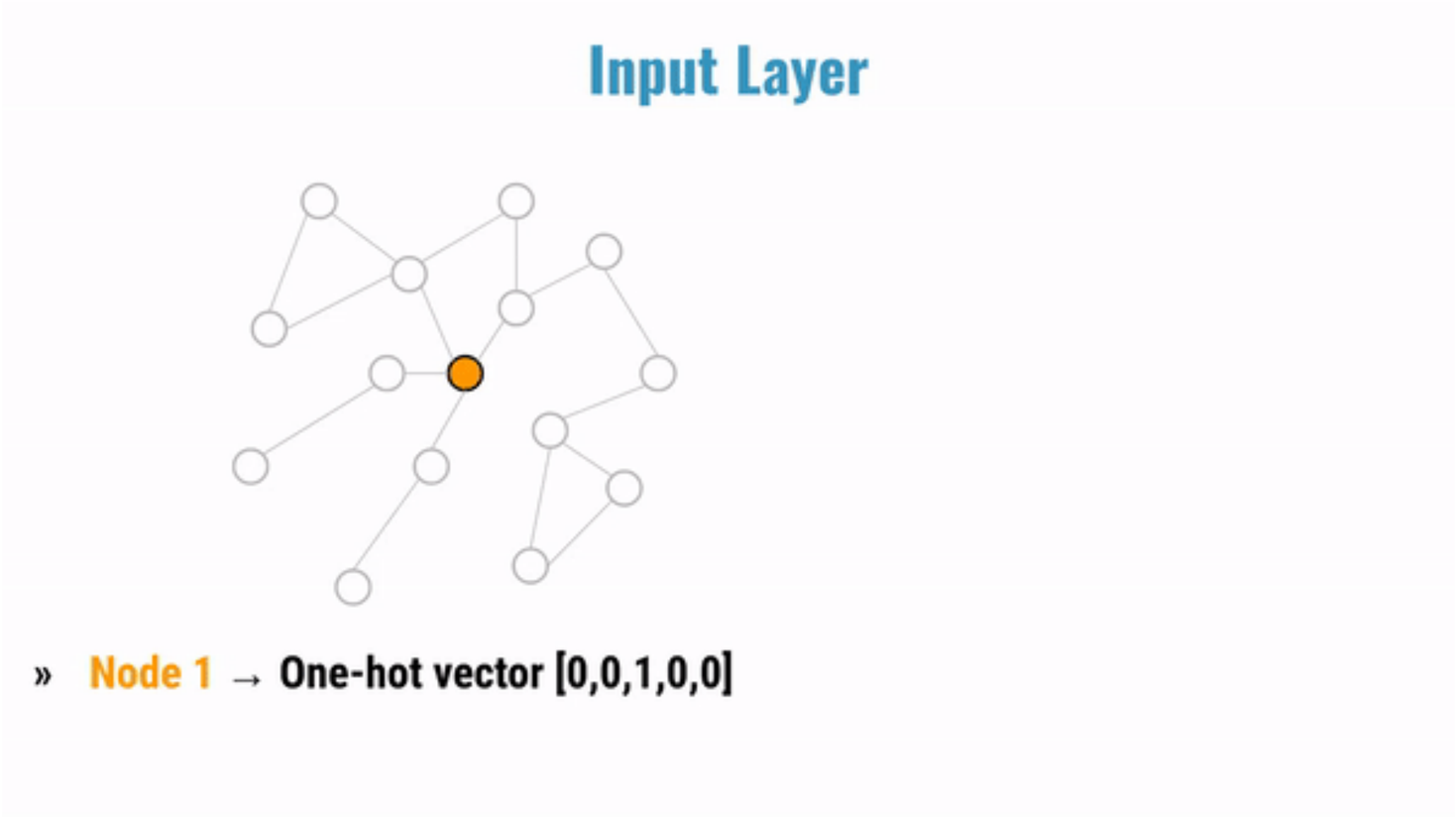
Stable Diffusion

- Incredible quality of results!
- Iterative process
- Can be cumbersome and slow

Graph Neural Networks

- A great resource: https://distill.pub/2021/gnn-intro/





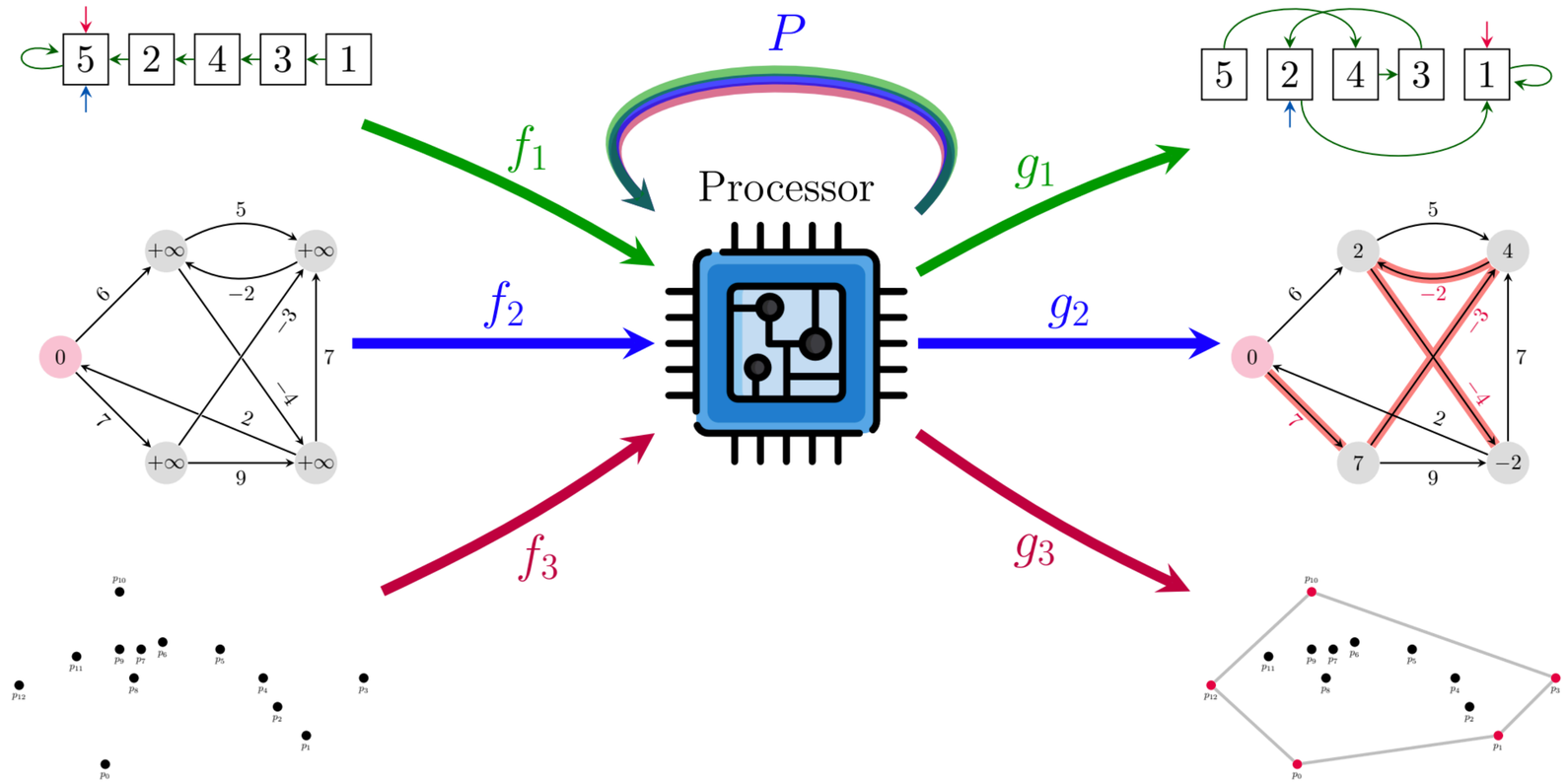
- Input data format: graph
- Tasks like: node feature prediction, edge feature prediction, edge or node classification, graph feature prediction or classification
- Learn weights for update of local features at node or edges.
- Stack as many layers as desirable.
- Repeat as many iterations of updates as needed.
- Pool and aggregate from nearby regions.

Paper: https://arxiv.org/pdf/2209.11142.pdf

Graph Neural Networks

A Generalist Neural Algorithmic Learner

- Heterogeneous inputs (and outputs) represented as graphs
- A single GNN processor capable of generalizing to out-of-distribution input lengths



GNNs in HEP

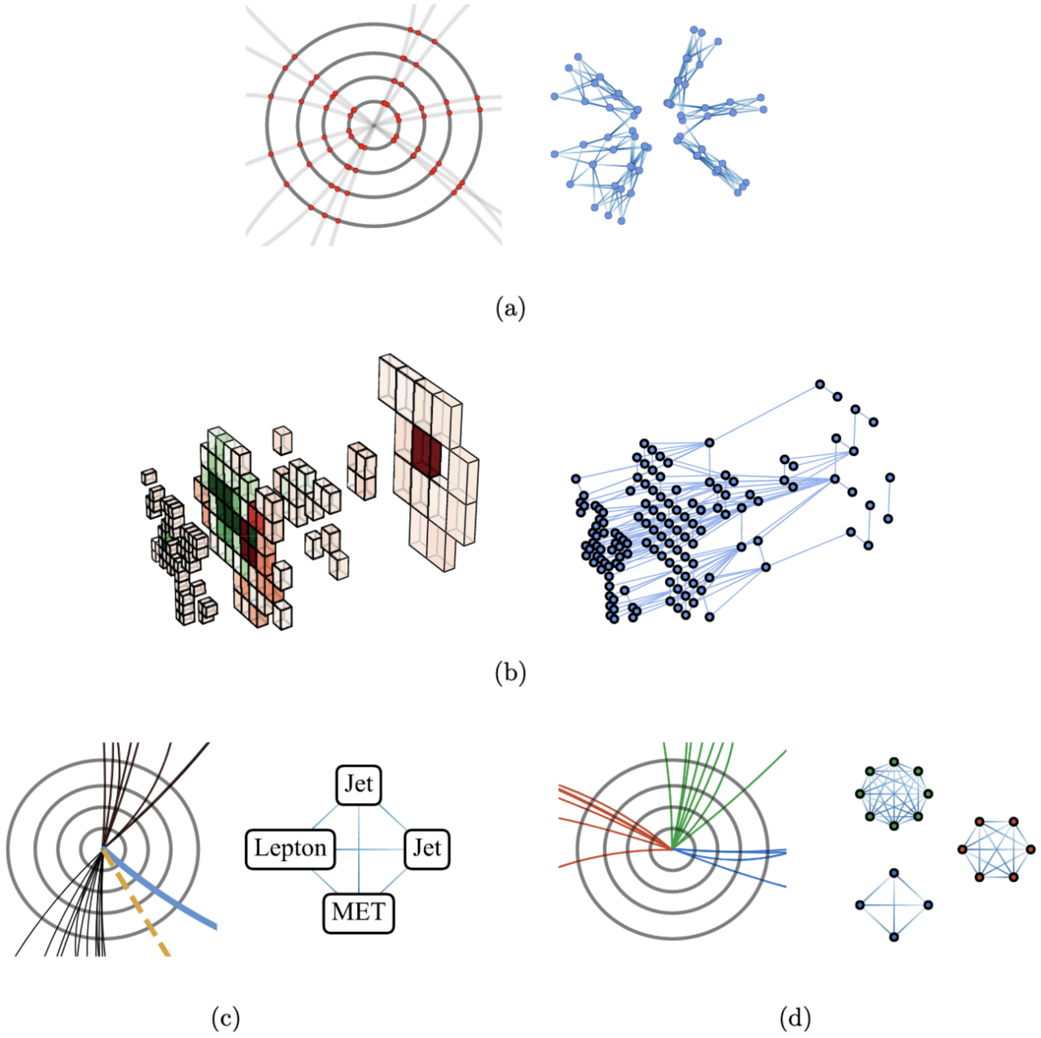


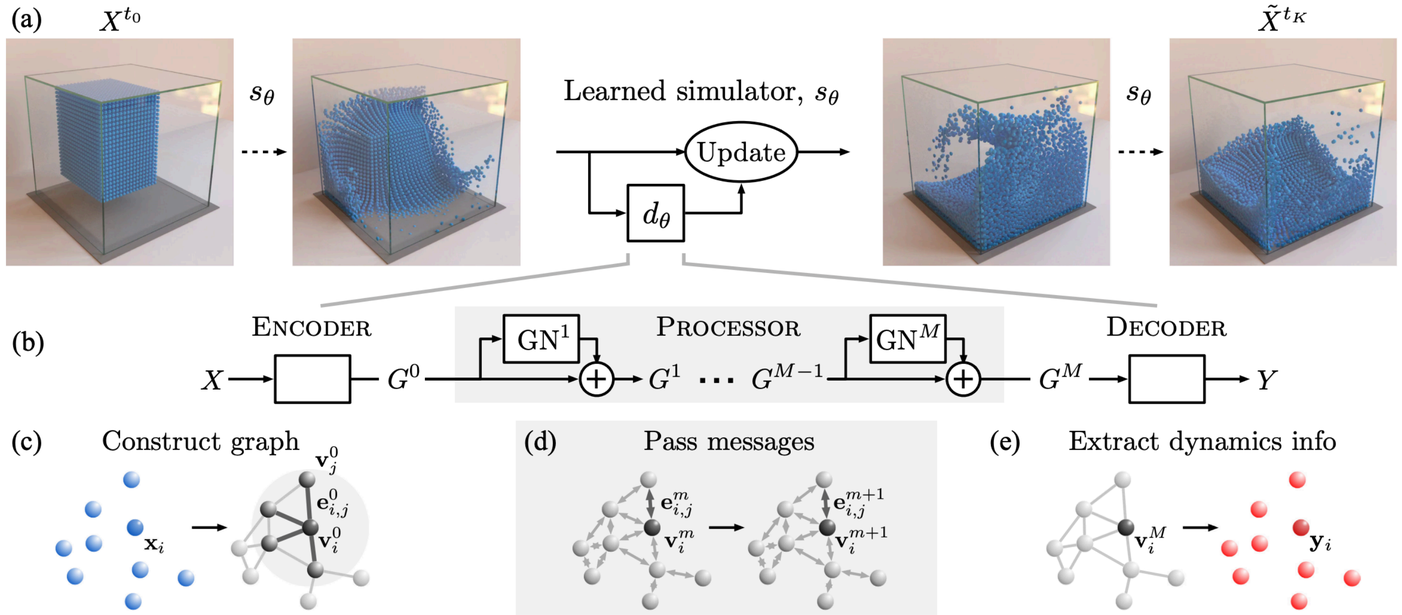
Figure 1: Examples of graph representations of particle physics data: (a) clustering tracking detector hits into tracks, (b) segmenting calorimeter cells, (c) classifying events with multiple types of physics objects, (d) jet classification based on the particles associated to the jet.

From:
https://github.com/iml-wg/HEPML-LivingReview

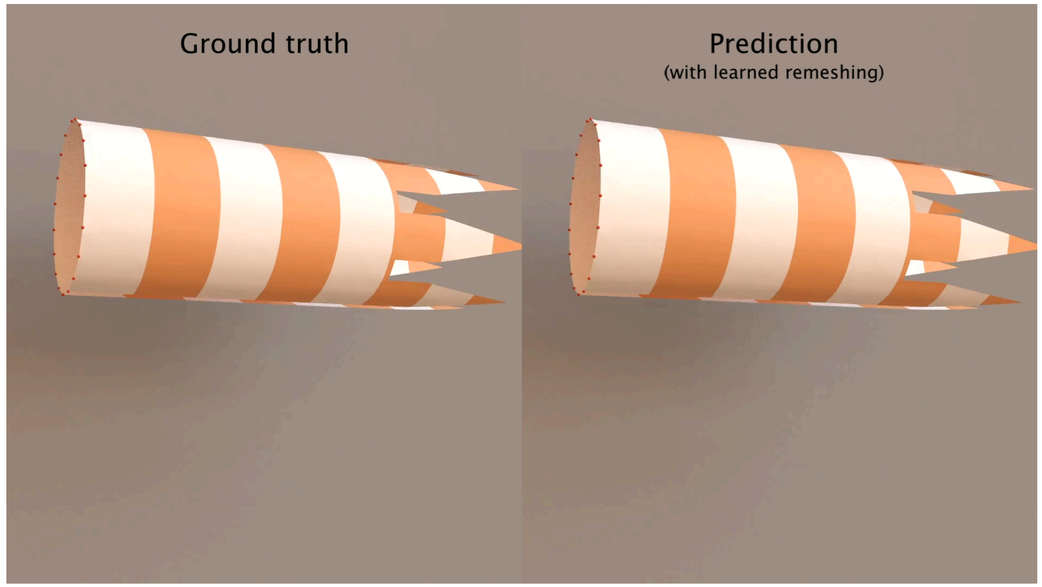
- Graphs
  - Neural Message Passing for Jet Physics
  - Graph Neural Networks for Particle Reconstruction in High Energy Physics detectors
  - Probing stop pair production at the LHC with graph neural networks [DOI]
  - Pileup mitigation at the Large Hadron Collider with graph neural networks [DOI]
  - Unveiling CP property of top-Higgs coupling with graph neural networks at the LHC [DOI]
  - JEDI-net: a jet identification algorithm based on interaction networks [DOI]
  - Learning representations of irregular particle-detector geometry with distance-weighted graph networks [DOI]
  - Interpretable deep learning for two-prong jet classification with jet spectra [DOI]
  - Neural Network-based Top Tagger with Two-Point Energy Correlations and Geometry of Soft Emissions [DOI]
  - Probing triple Higgs coupling with machine learning at the LHC
  - Casting a graph net to catch dark showers [DOI]
  - Graph neural networks in particle physics [DOI]
  - Distance-Weighted Graph Neural Networks on FPGAs for Real-Time Particle Reconstruction in High Energy Physics [DOI]
  - Supervised Jet Clustering with Graph Neural Networks for Lorentz Boosted Bosons [DOI]
  - Track Seeding and Labelling with Embedded-space Graph Neural Networks
  - Graph neural network for 3D classification of ambiguities and optical crosstalk in scintillator-based neutrino detectors [DOI]
  - The Boosted Higgs Jet Reconstruction via Graph Neural Network
  - Accelerated Charged Particle Tracking with Graph Neural Networks on FPGAs
  - Particle Track Reconstruction using Geometric Deep Learning
  - Jet tagging in the Lund plane with graph networks [DOI]
  - Vertex and Energy Reconstruction in JUNO with Machine Learning Methods
  - MLPF: Efficient machine-learned particle-flow reconstruction using graph neural networks
  - Towards a realistic track reconstruction algorithm based on graph neural networks for the HL-LHC
  - Deep Learning strategies for ProtoDUNE raw data denoising
  - Graph Neural Network for Object Reconstruction in Liquid Argon Time Projection Chambers
  - Instance Segmentation GNNs for One-Shot Conformal Tracking at the LHC
  - Charged particle tracking via edge-classifying interaction networks
  - Jet characterization in Heavy Ion Collisions by QCD-Aware Graph Neural Networks
  - Graph Generative Models for Fast Detector Simulations in High Energy Physics
  - Segmentation of EM showers for neutrino experiments with deep graph neural networks
  - Anomaly detection with Convolutional Graph Neural Networks
  - Energy-weighted Message Passing: an infra-red and collinear safe graph neural network algorithm
  - Improved Constraints on Effective Top Quark Interactions using Edge Convolution Networks
  - Particle Graph Autoencoders and Differentiable, Learned Energy Mover's Distance
  - Graph Neural Networks for Charged Particle Tracking on FPGAs
  - Machine Learning for Particle Flow Reconstruction at CMS
  - An Efficient Lorentz Equivariant Graph Neural Network for Jet Tagging
  - End-to-end multi-particle reconstruction in high occupancy imaging calorimeters with graph neural networks

Graph Neural Networks

Science Applications



Learning to Simulate Complex Physics with Graph Networks
https://arxiv.org/pdf/2002.09405.pdf



MeshGraphNetworks
https://arxiv.org/pdf/2010.03409.pdf

AI for Science

Nowcasting with GANs

- Predicting next 2 hour precipitations
- Spatial and temporal changes modeled with discriminators
- Inductive biases from application domain
- Blind study with meteorologists to evaluate performance

Thanks

Opportunities:

All internship roles
deepmind.com/careers/internships

Research Scientist Intern, AI for Sciences, 2023 (London)
https://grnh.se/2a2e73941us

Research Scientist Intern 2023 (London)
https://grnh.se/427b13aa1us

Science Software Engineer (London)
https://grnh.se/256d51311us

Research Scientist (London)
https://grnh.se/1e09c70b1us

Scaling LSLMs

From the “Scaling Laws” prediction, the field has been constructing larger and larger models expecting better and better performance.

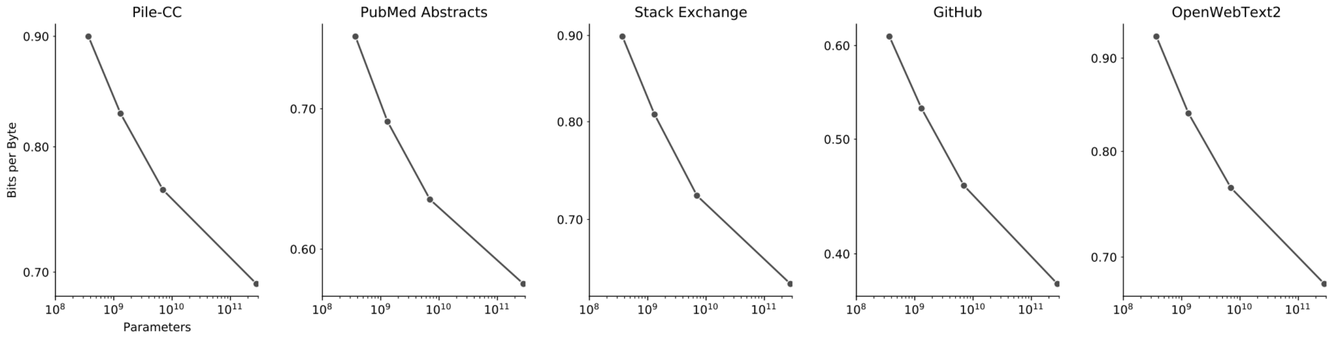
Model	Size (Billion Parameters)	Training Tokens
GPT-3 (Brown et al., 2020)	175B	300 Billion
Jurassic (Lieber et al., 2021)	178B	300 Billion
Gopher (Rae et al., 2021)	280B	300 Billion
MT-NLG 530B (Smith et al., 2022)	530B	270 Billion

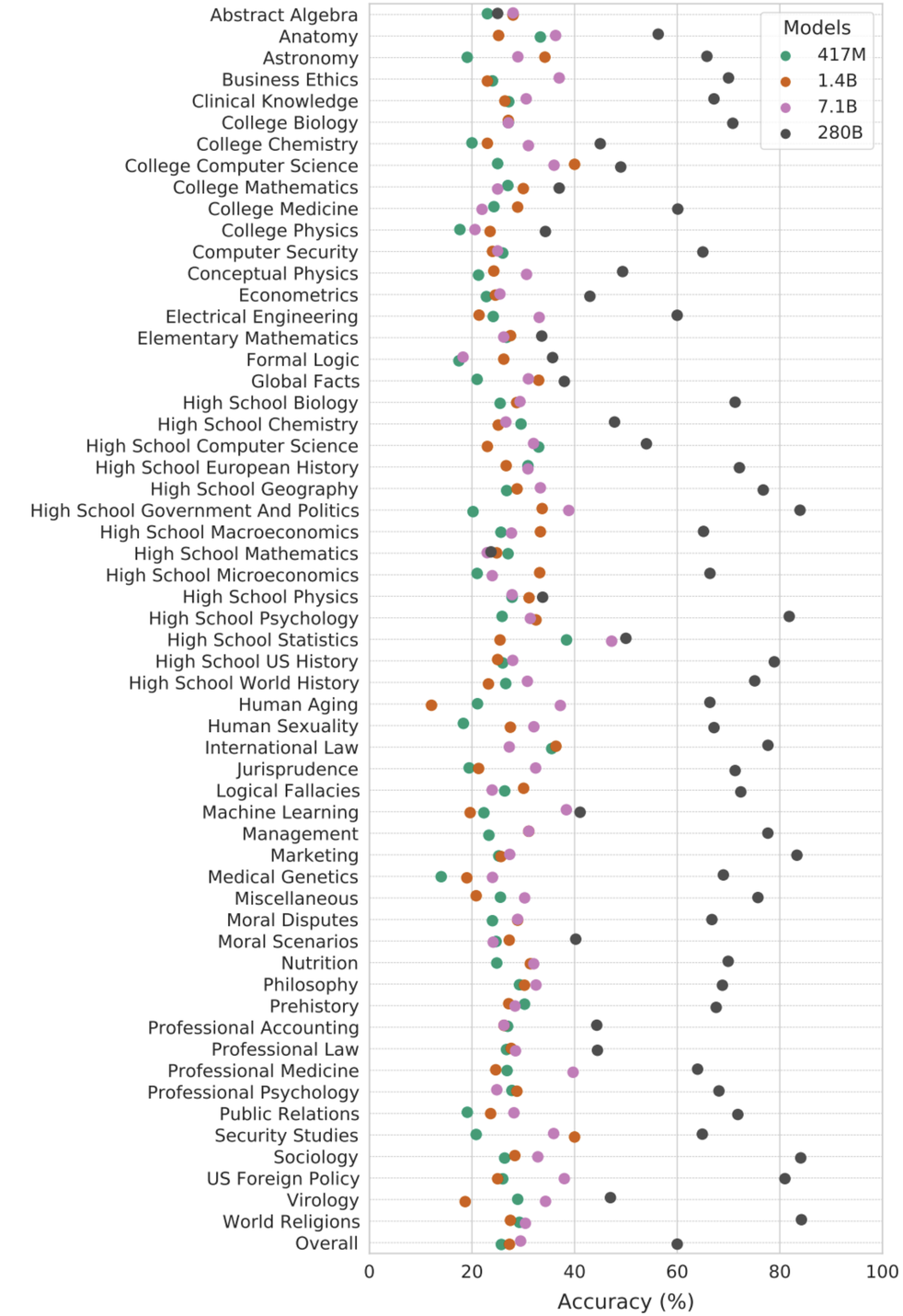
arxiv:2112.11446

Scaling LSLMs: Gopher

Goal: test the limits of scaling laws.

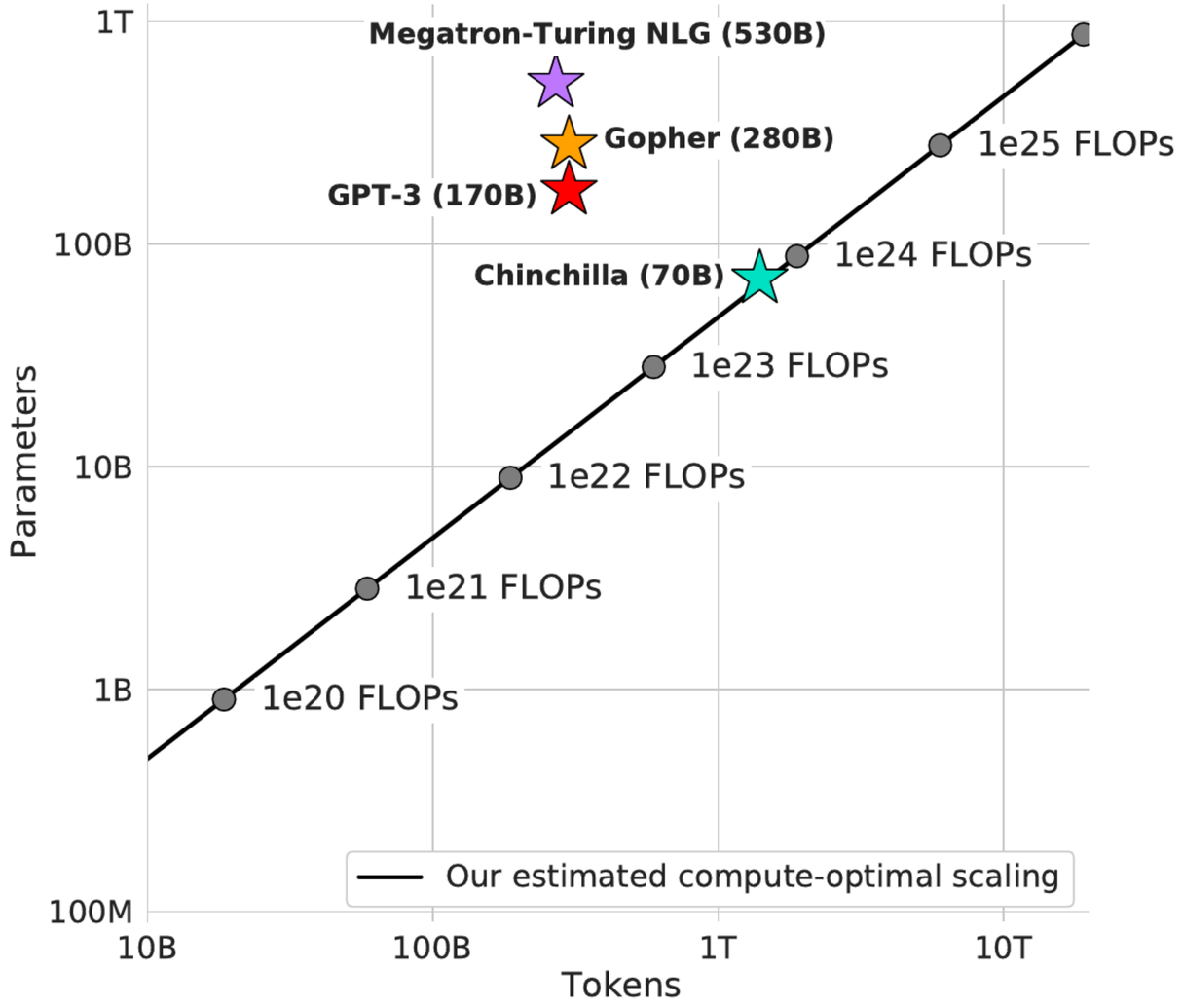
- Family of LMs ranging from 44M to 280B parameters
- Surfaces ‘skills’ where scale helps…
  - Reading comprehension, fact-checking, and the identification of toxic language, etc.
- …and skills where scale does not seem to help.
  - Logical reasoning, common-sense tasks





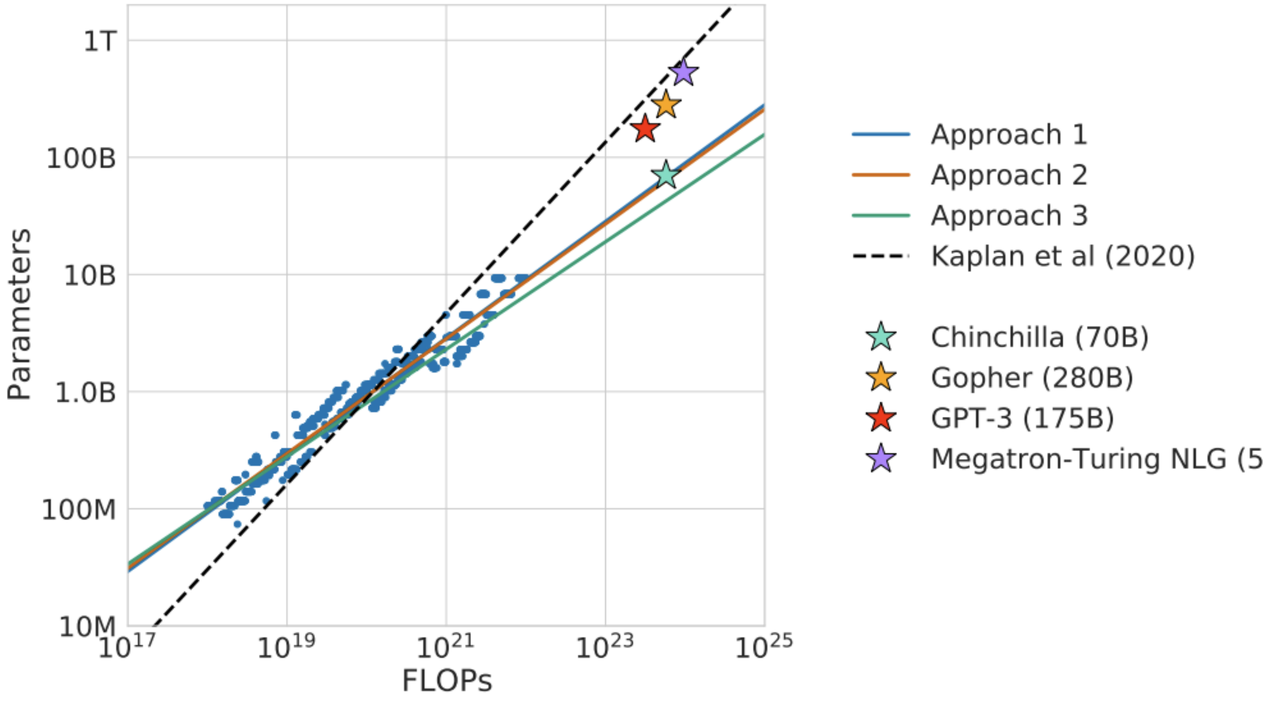
arXiv:2203.15556

Chinchilla: Train Smarter, not Bigger



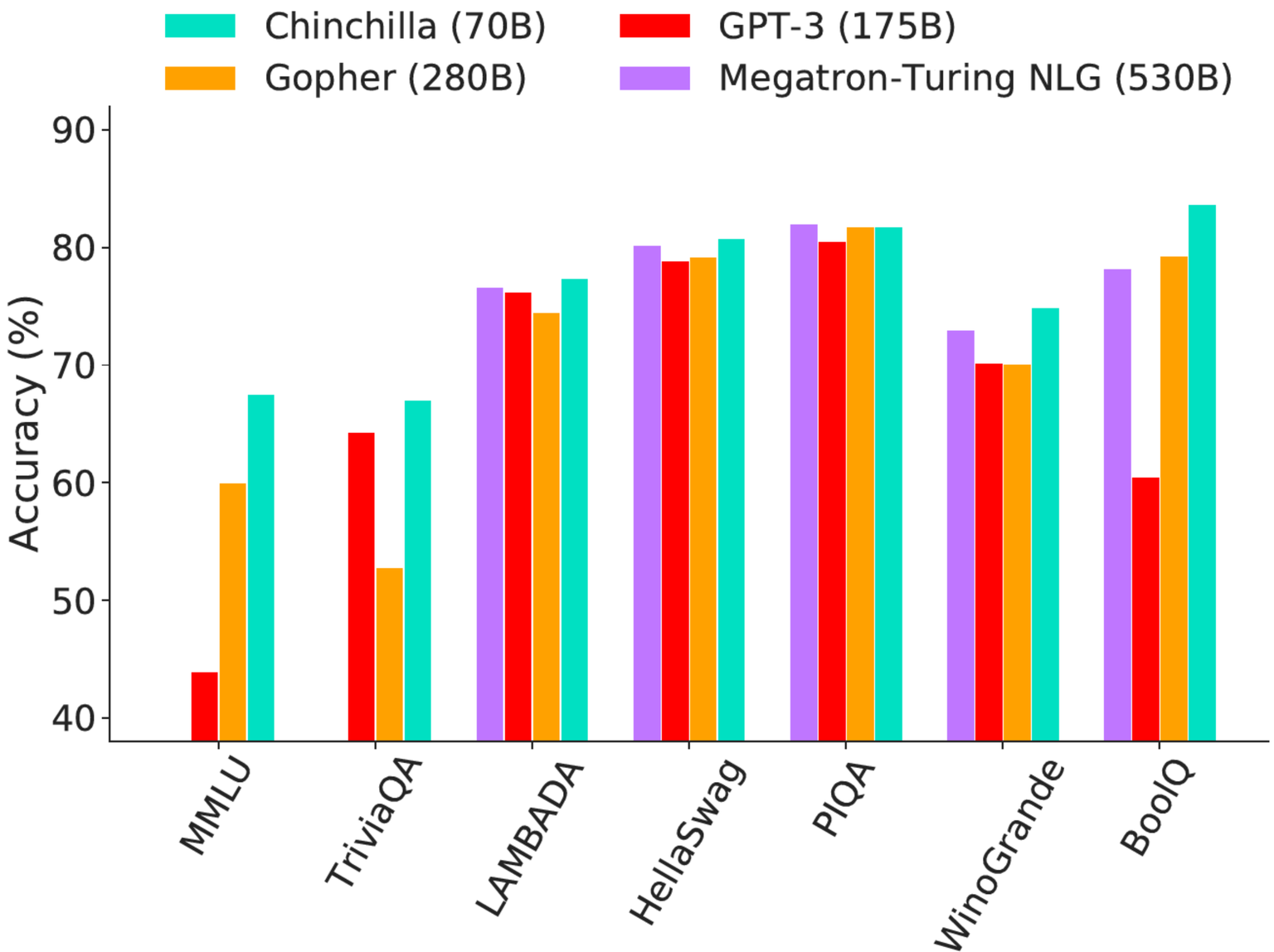
Improved scaling laws as a function of training set size → data matters more than previously expected!

Make optimal use of FLOPs by training smaller models on more data.



arXiv:2203.15556

Chinchilla: Train Smarter, not Bigger



"larger, high quality datasets will play a key role in any further scaling of language models"

Parameters	FLOPs	FLOPs (in *Gopher* unit)	Tokens
400 Million	1.92e+19	1/29,968	8.0 Billion
1 Billion	1.21e+20	1/4,761	20.2 Billion
10 Billion	1.23e+22	1/46	205.1 Billion
67 Billion	5.76e+23	1	1.5 Trillion
175 Billion	3.85e+24	6.7	3.7 Trillion
280 Billion	9.90e+24	17.2	5.9 Trillion
520 Billion	3.43e+25	59.5	11.0 Trillion
1 Trillion	1.27e+26	221.3	21.2 Trillion
10 Trillion	1.30e+28	22515.9	216.2 Trillion

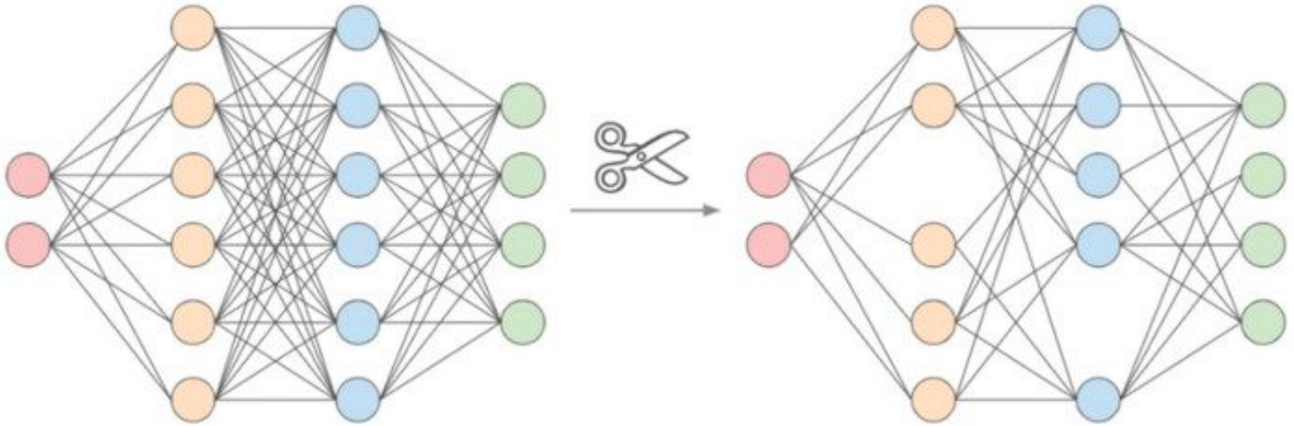
Compression, Sparsity

Hard for pre-training

- SMoE models (Switch, …)
- Hard to prune or distill Gopher
- Hardware co-design to take full advantage

Dense2Sparse

- Learning *f* != representing *f*
- In training vs post-training
  - Can be coupled with fine-tuning
- Geared towards inference
- Requires memory budget and ability to train initial dense model → wasteful



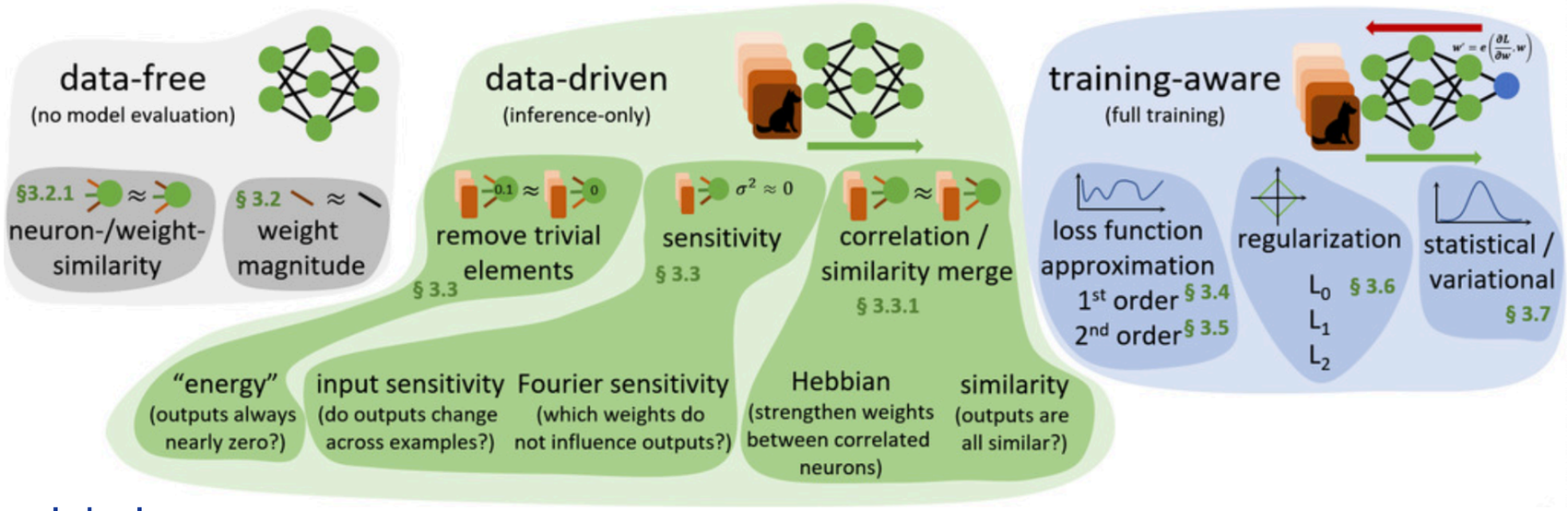
How to do this in practice?

Pruning should remove unnecessary redundancy and unused capacity

How to Prune

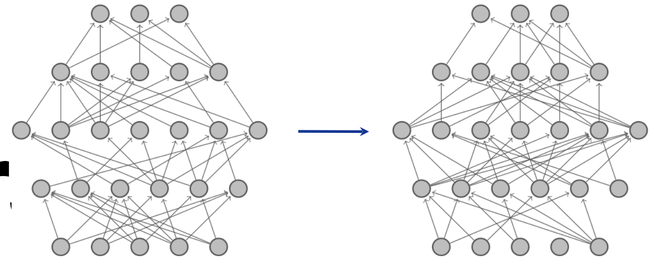
Pruning methods differ across many dimensions:

- weight magnitude, activations, gradients, Hessian, interpretability, credit assignment, random, etc.



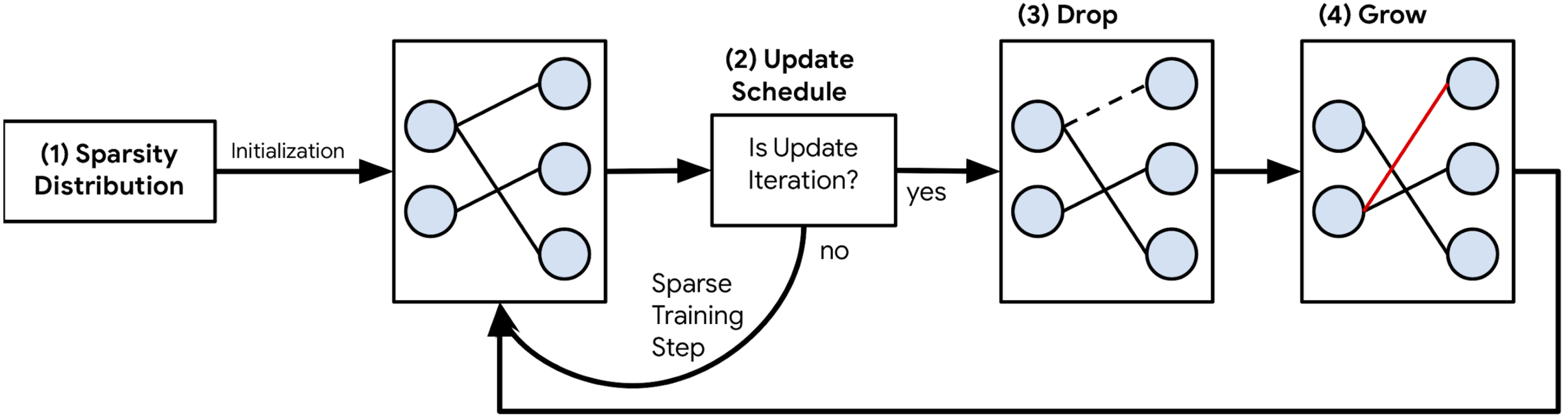
- Layer-wise vs global
- Sparsity level
- Unstructured vs structured, etc.
- One-shot vs iterative pruning
- Followed by: finetuning, reinitialization, rewinding
- …

Sparse2Spar



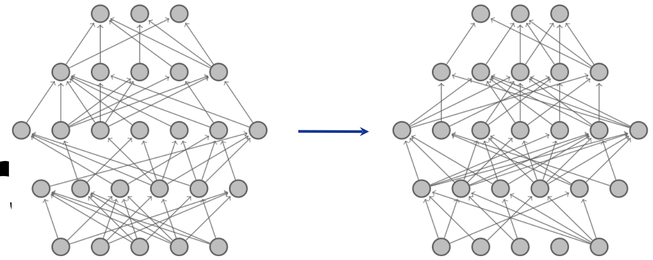
Start with a sparse network to reduce memory and training FLOPs

Example: **RiGL**



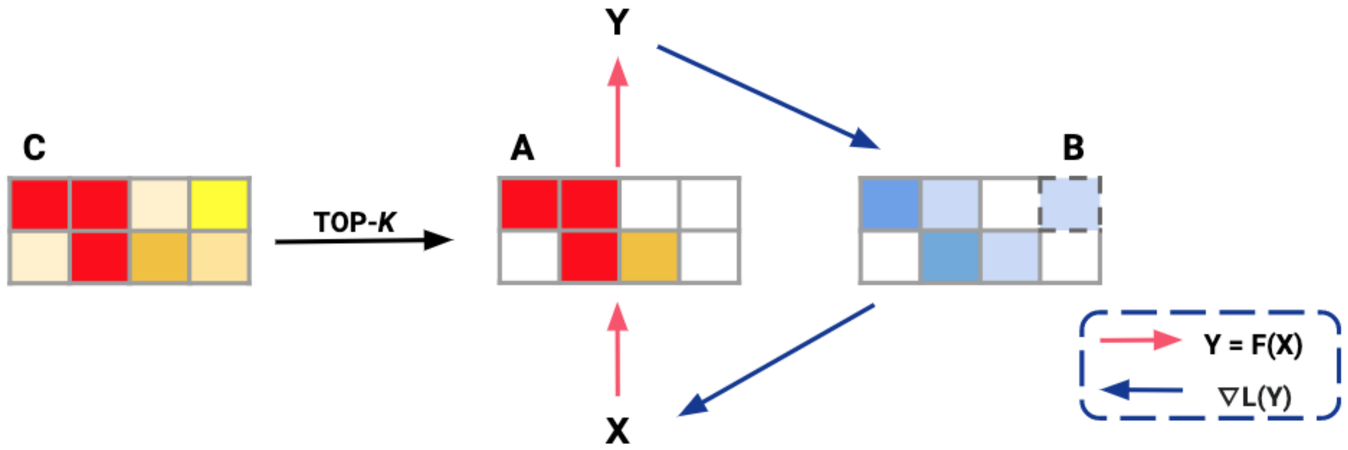
Issue: every N iterations, need to compute Top-K over gradient. Efficient implementation?

Sparse2Spar



Start with a sparse network to reduce memory and training FLOPs

Example: **Top-KAST**



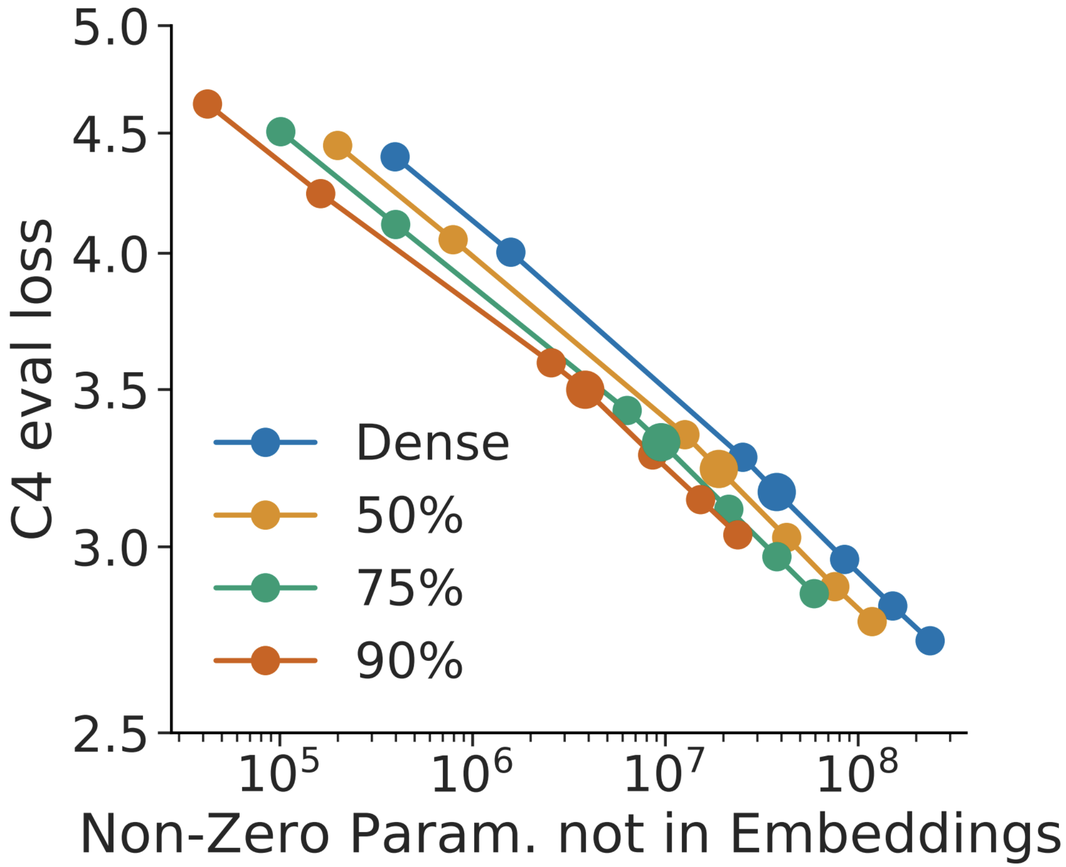
Start at desired final sparsity with a top-k over weight norms and keep around an extra set of connections that are only updated in the backward pass. If these grow large enough, they turn “on” in the forward.

Sparsifying LSLMs

"While we show modest success in the compression of these models, resulting in small shifts in the scaling curves, on the whole, none of the methods we explore are remarkably successful. The general finding is that whilst compressing models for a particular application has seen success, it is difficult to compress them for the objective of language modelling over a diverse corpus."

90% sparsity requires ~2.5× fewer params for a given eval loss. However:

- unstructured sparsity is hard to take advantage of on most accelerators → 2.5x reduction in #params is not sufficient to offset the decrease in efficiency of doing sparse computations.
- required instantiating 10x larger dense model
- sparsifying without appreciable accuracy drop only leads to 20-30% sparsity



See "Routing Nets" for most successful example of sparsity in scaling LSLMs!

Interpretability

How do these models really work?

- Mechanistic interpretability:
  - Induction heads
  - Superposition (see other Anthropic results)
- Decoding of vision+language
-

Towards understanding LSLMs

Mechanistic Interpretability: Approach to interpretability focused on understanding the underlying physical mechanisms that correspond to higher level observed behavior in the system
→ find mechanisms that generate the patterns we see

- Examine the phenomena in a network like in a physical system (empirical approach)
- Concepts of cause and effect, interventions
- Ability to probe the system, interact with it, test hypotheses
- Assumes modularity of behavior
- Analogous to *mechanics* in physics (statics + dynamics)
- Start from “what?” and get to “how?”

A Mathematical Framework for Transformer Circuits

AUTHORS

Nelson Elhage*†, Neel Nanda*, Catherine Olsson*, Tom Henighan†, Nicholas Joseph†, Ben Mann†, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, Chris Olah‡

AFFILIATION

Anthropic

PUBLISHED

Dec 22, 2021

https://transformer-circuits.pub/2021/framework/index.html

In-context Learning and Induction Heads

AUTHORS

Catherine Olsson*, Nelson Elhage*, Neel Nanda*, Nicholas Joseph†, Nova DasSarma†, Tom Henighan†, Ben Mann†, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, Chris Olah‡

AFFILIATION

Anthropic

PUBLISHED

Mar 8, 2022

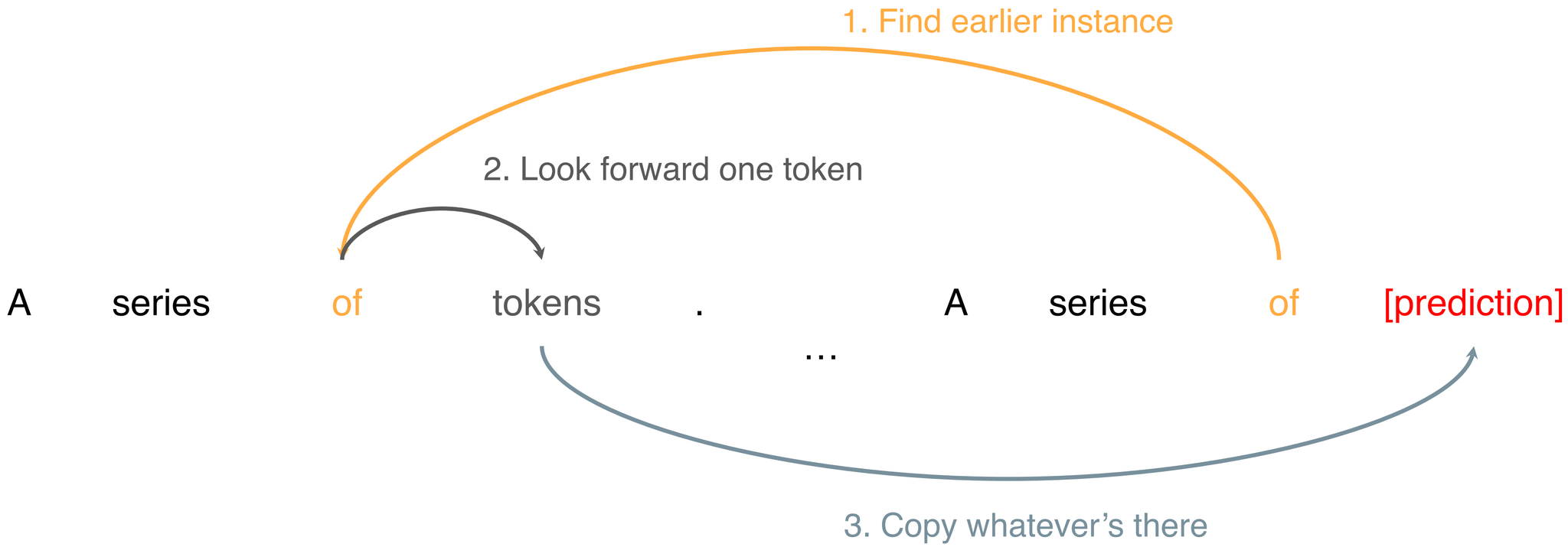
https://transformer-circuits.pub/2022/in-context-learning-and-induction-heads/index.html

Visualization of Induction Heads in our 2-Layer Attention-Only Transformers

Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr Dursley was the director of a firm called Grunnings, which made drills. He was a big, beefy man with hardly any neck, although he did have a very large moustache. Mrs Dursley was thin and blonde and had nearly twice the usual amount of neck, which came in very useful as she spent so much of her time craning over garden fences, spying on the neighbours. The Dursleys had a small son called Dudley and in their opinion there was no finer boy anywhere. The Dursleys had everything they wanted, but they also had a secret, and their greatest fear was that somebody would discover it. They didn't think they could bear it if anyone found out about the Potters. Mrs Potter was Mrs Dursley's sister, but they hadn't met for several years; in fact, Mrs Dursley pretended she didn't have a sister, because her sister and her good-for-nothing husband were as unDursleyish as it was possible to be. The Dursleys shuddered to think what the neighbours would say if the Potters arrived in the street. The Dursleys knew that the Potters had a small son, too, but they had never even seen him. This boy was another good reason for keeping the Potters away; they didn't want Dudley mixing with a child like that.

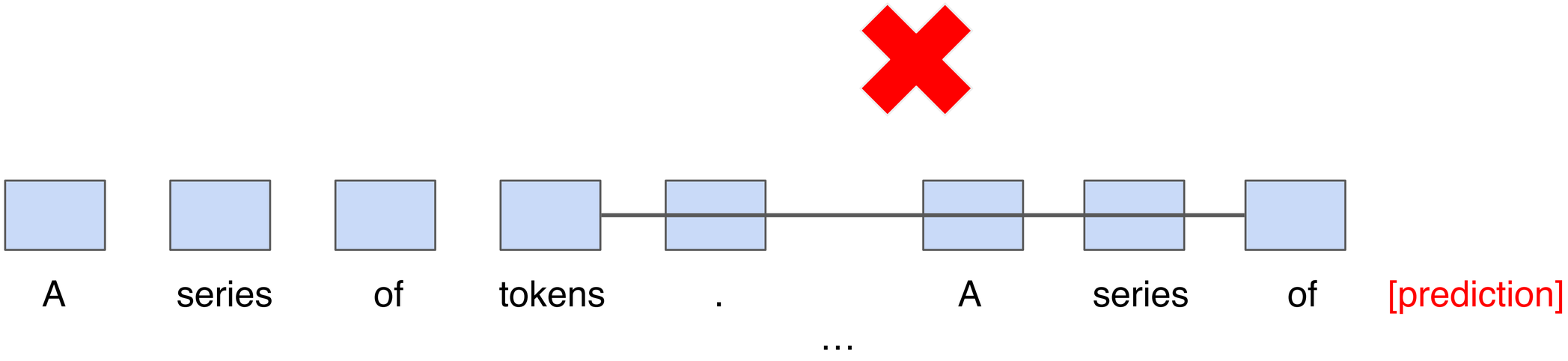
Mr and Mrs Dursley, of number four, Privet Drive, were proud to say that they were perfectly normal, thank you very much. They were the last people you'd expect to be involved in anything strange or mysterious, because they just didn't hold with such nonsense. Mr Dursley was the director of a firm called Grunnings, which made drills.

Copying arbitrary sequences



Let’s think about this mechanistically

Can’t expect a copying head to do it - the trigram isn’t that likely!



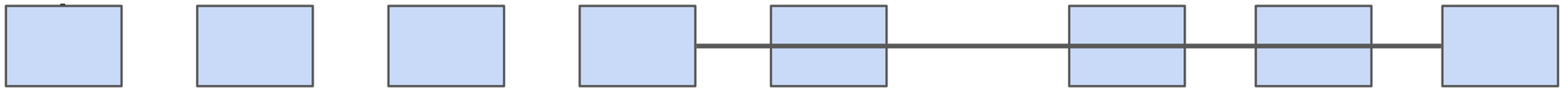
Let's think about this mechanistically

(but there are some "copying heads" that do this for more likely sequences)

Examples from Elhage et al.:

`[b]` `[a]` `[b or b']`

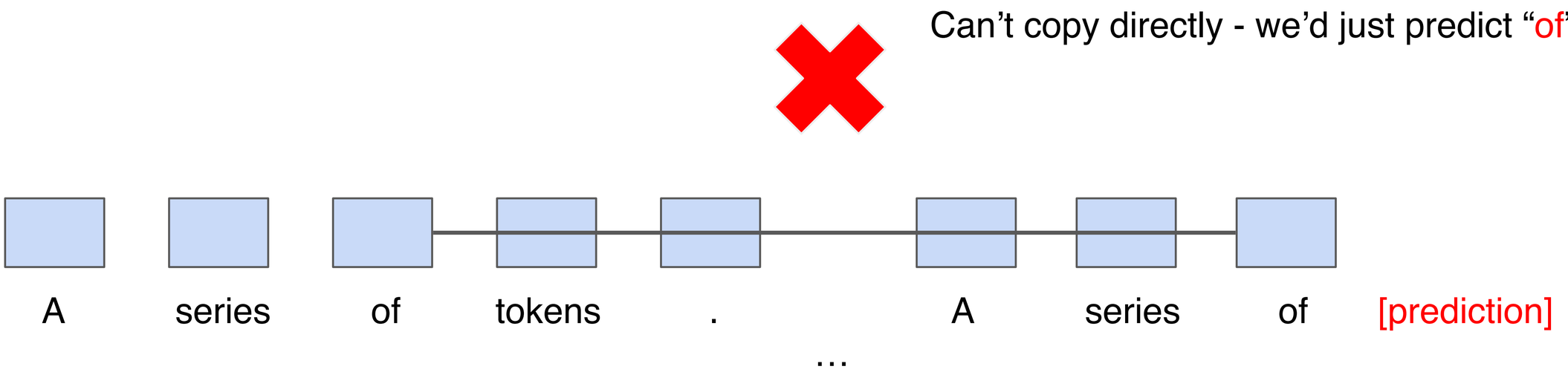
Source Token	Destination Token	Out Token	Example Skip Tri-grams
" perfect"	" are", " looks", " is", " provides"	" perfect", " super", " absolute", " pure"	" perfect... are perfect", " perfect... looks super"
" large"	" contains", " using", " specify", " contain"	" large", " small", " very", " huge"	" large... using large", " large... contains small"
" two"	" One", "\n ", " has", "\r\n ", "One"	" two", " three", " four", " five", " one"	" two... One two", " two... has three"
"lambda"	" $\\", "}{\\", "+\\", "(\\", " ${\\"	"lambda", "sorted", " lambda", "operator"	"lambda... $\\lambda", "lambda... +\\lambda"
"nbsp"	"&", " \"&", "}&", ">&", "=&"	"nbsp", "01", "gt", "00012", "nbs", "quot"	"nbsp... ", "nbsp... > "
"Great"	"The", " The", " the", " contains", " /"	" Great", " great", " poor", " Every"	"Great... The Great", "Great... the great"



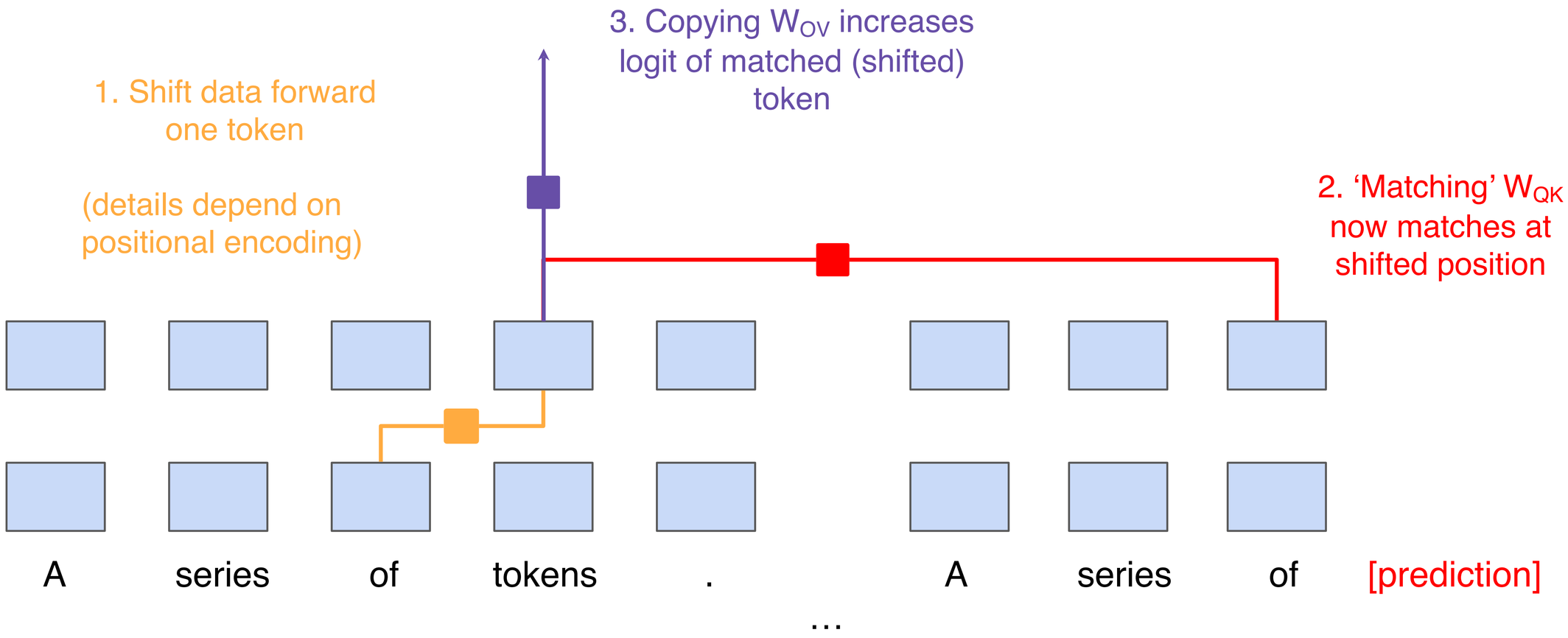
series of tokens . A series of [prediction]

...

Let’s think about this mechanistically



Let’s think about this mechanistically



Induction in action

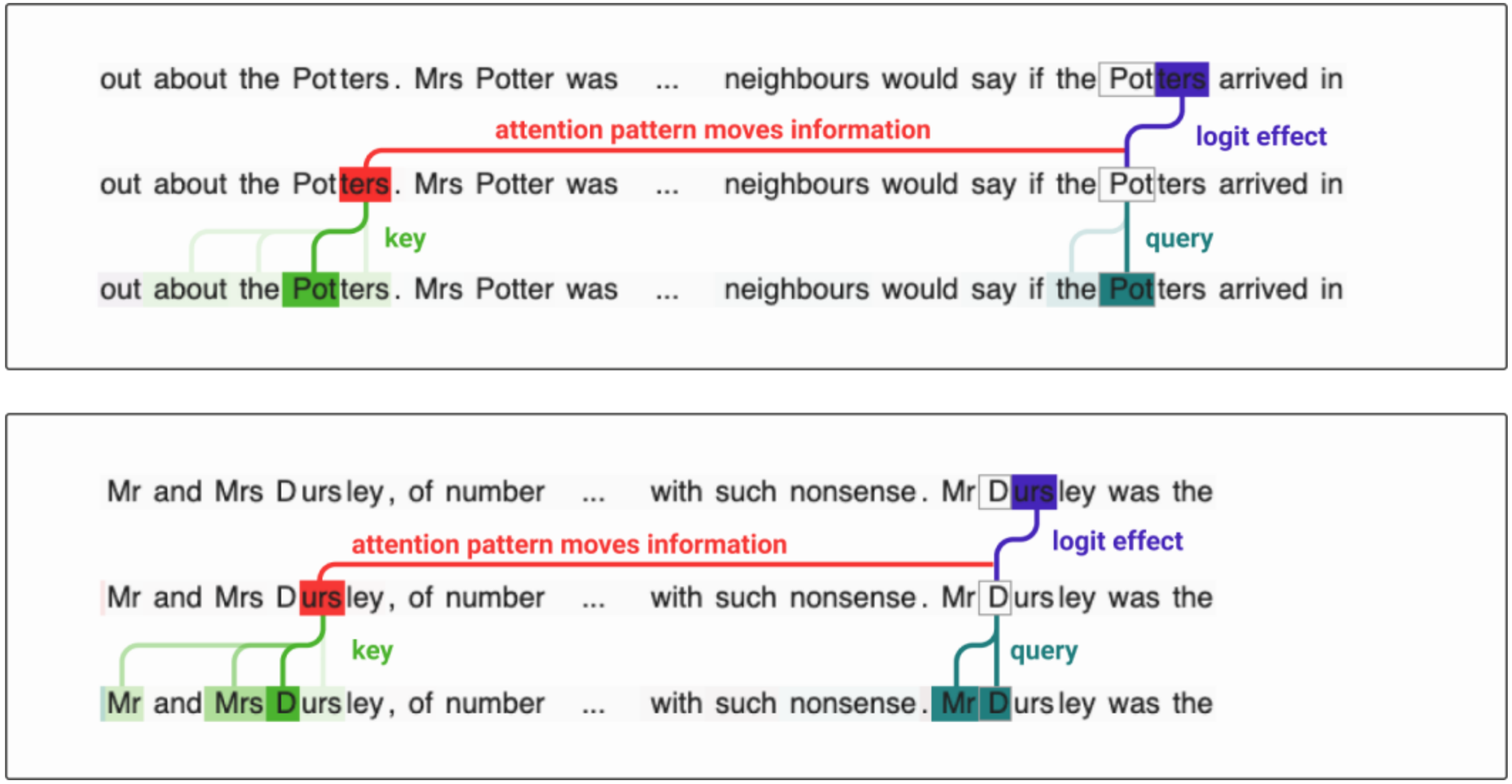
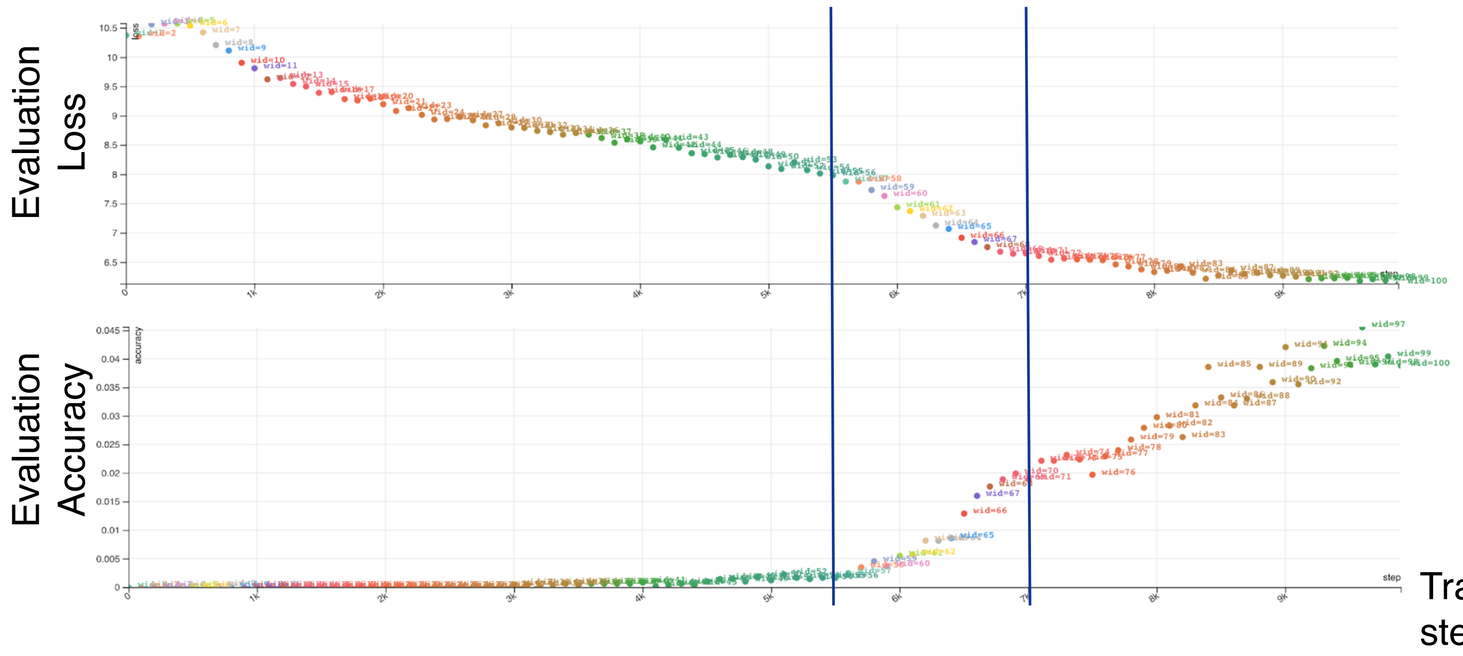


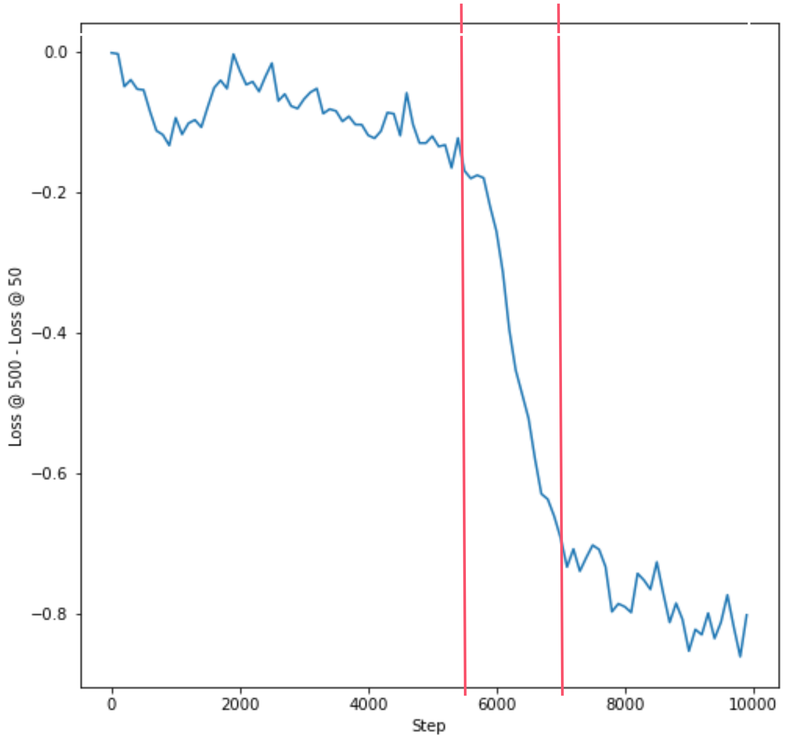
Image from Elhage et al., 2021

Induction heads phase change

Replicating Olsson et al. 2022 and Elhange et al. 2021

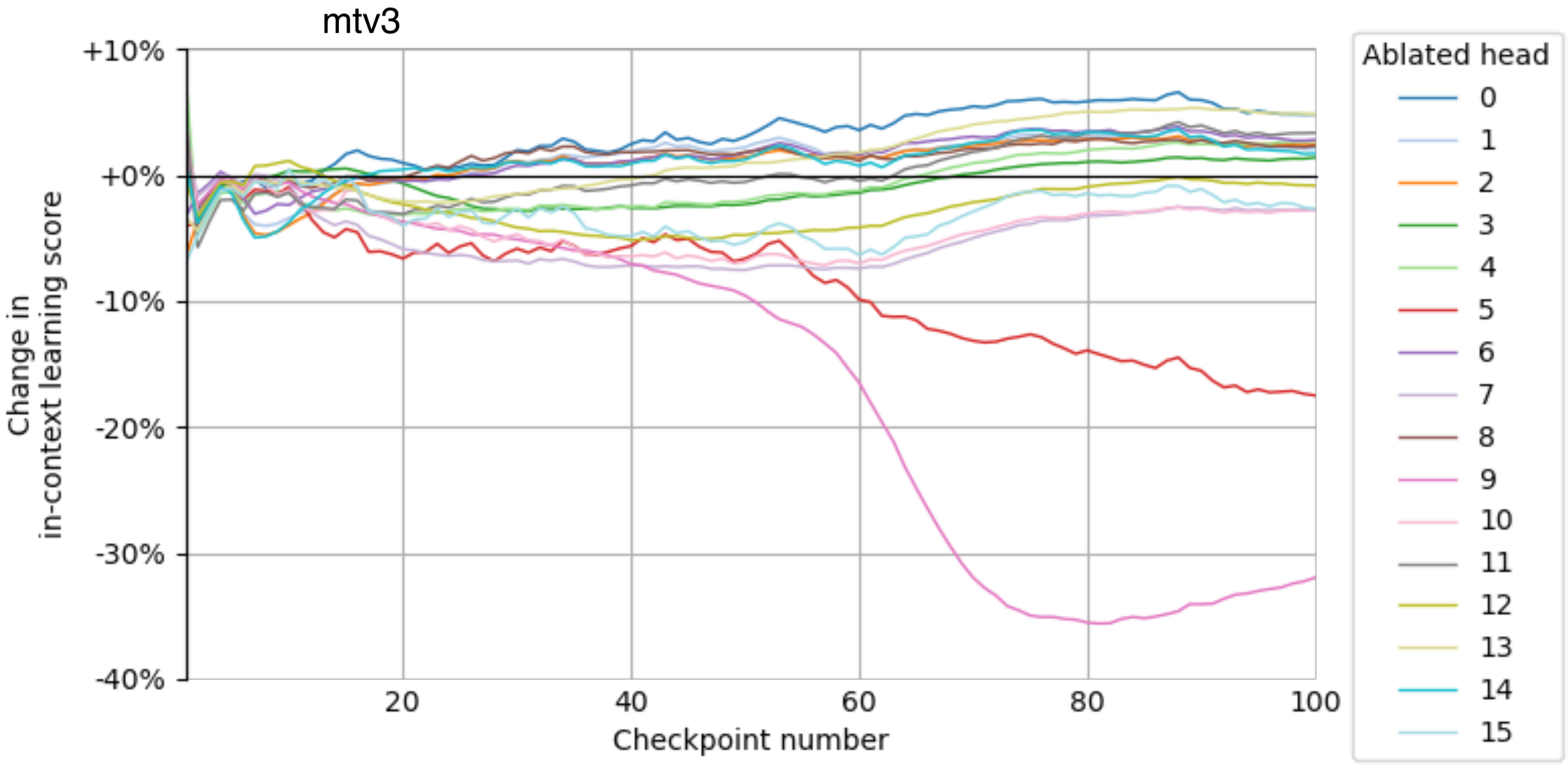
The occurrence of the phase-transition bump, observed in the learning curve, corresponds to a clear phase transition and behavioral change in the model: the model picks up the ability to more easily generate correct tokens later on in the context





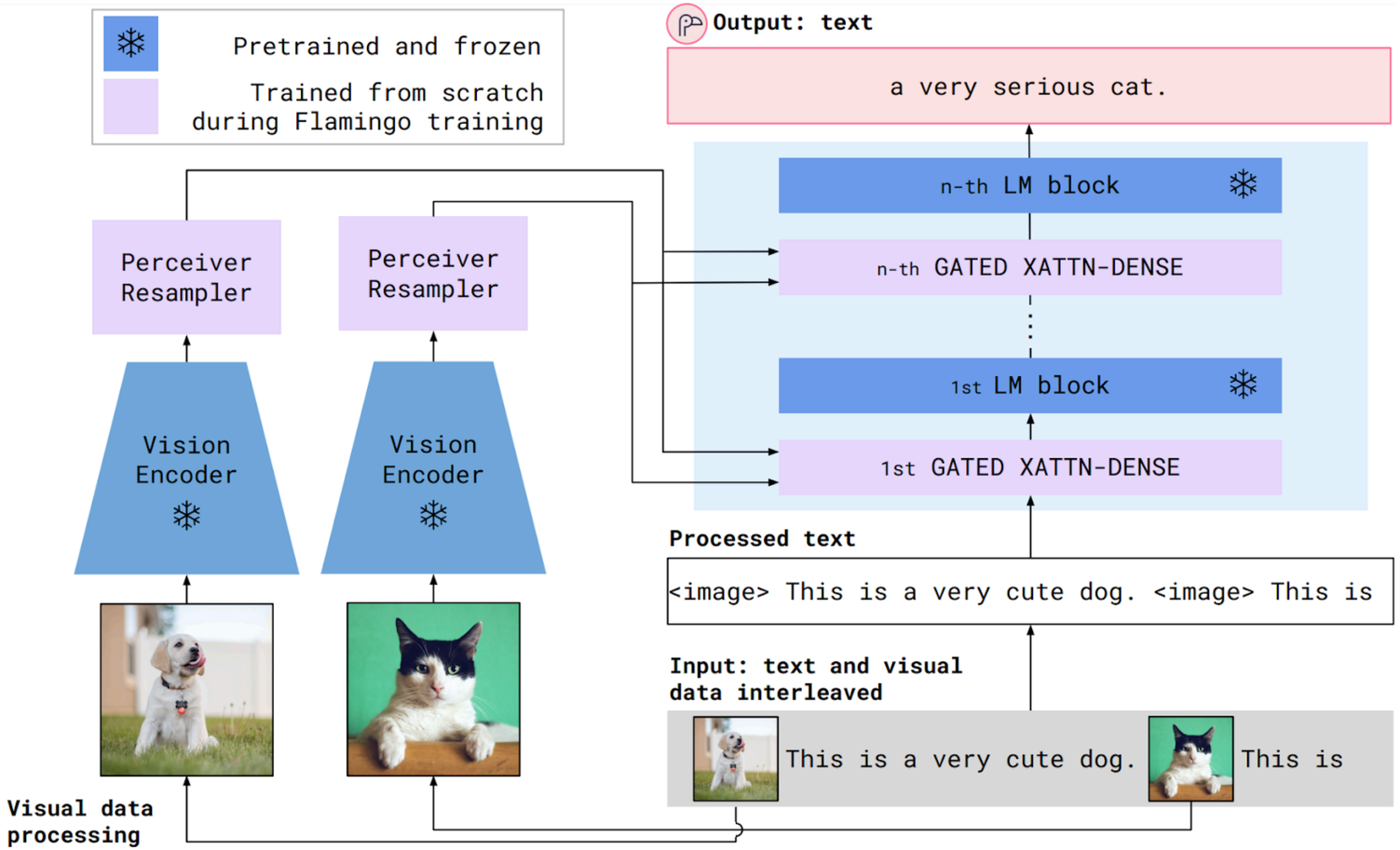
A "poswamah" is a type of Elvish sword. An example of a sentence that uses the word poswamah is: "The elf took out his poswamah and started swinging it around."

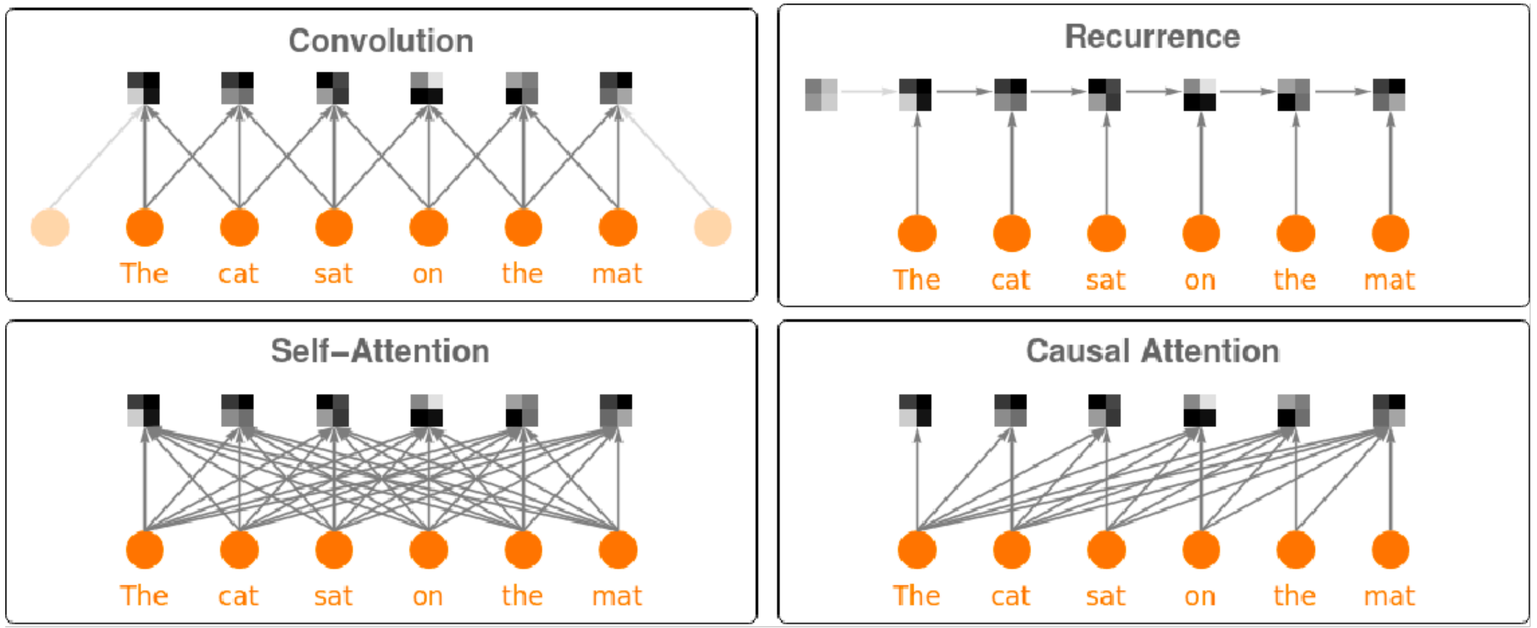
A "saosihd" is a colorful flower that grows on Alpine glaciers. An example of a sentence that uses the word saosihd is: "A saosihd bloomed, and then it was gone again."



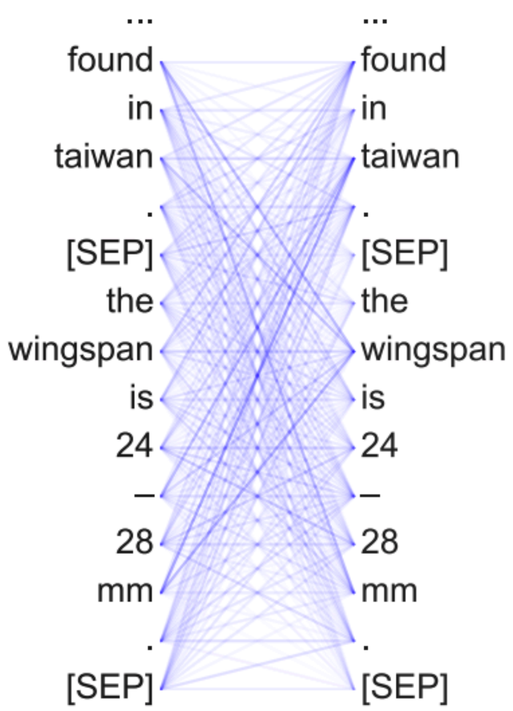
Negative change means that the ability of the model to do in-context learning decreases as that head is ablated

Interpreting VLMs?

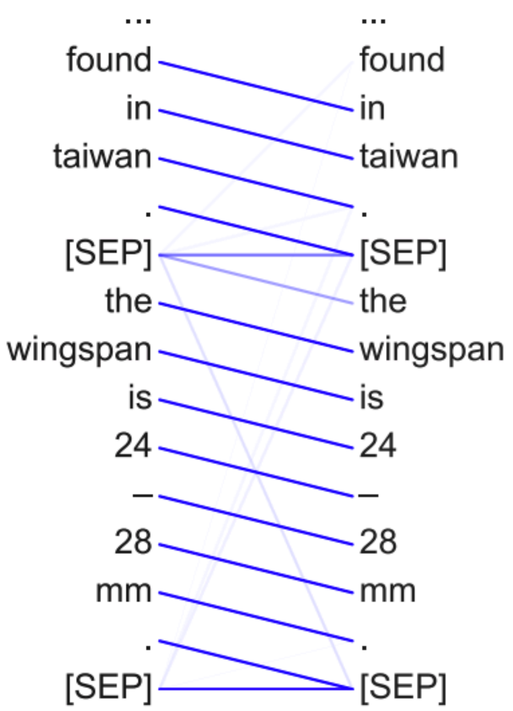



Convolution
The cat sat on the mat
Recurrence
The cat sat on the mat
Self–Attention
The cat sat on the mat
Causal Attention
The cat sat on the mat

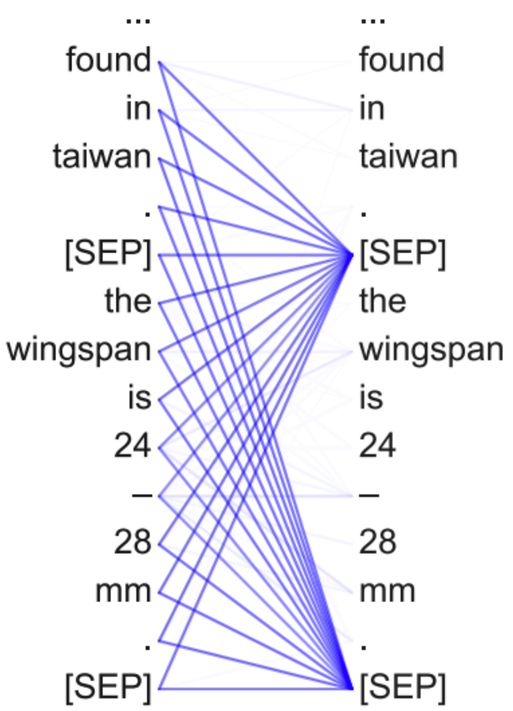
Head 1-1
Attends broadly



Head 3-1
Attends to next token



Head 8-7
Attends to [SEP]



Things to discuss if I have time:

- What are the missing ingredients? Can scale get us all the way? Or are we missing some building blocks for reasoning? (Memory, RL, inductive biases, invariances, guarantees, explicit knowledge graphs, …)
- External memory (memorizing transformers, etc.)