

Emergent Abilities of LLMs

a.k.a. why all this recent craziness?

Perttu Hämäläinen, March 2023

Aalto University

A phase change?

The Age of AI has begun

Artificial intelligence is as revolutionary as mobile phones and the Internet.

By Bill Gates | March 21, 2023 • 14 minute read



In my lifetime, I've seen two demonstrations of technology that struck me as revolutionary.

The first time was in 1980, when I was introduced to a graphical user interface—the forerunner of every modern operating system, including Windows. I sat with the person who had shown me the demo, a brilliant programmer named Charles Simonyi, and we immediately started brainstorming about all the things we could do with such a user-friendly approach to computing. Charles eventually joined Microsoft, Windows became the backbone of Microsoft, and the thinking we did after that demo helped set the company's agenda for the next 15 years.

The second big surprise came just last year. I'd been meeting with the team from [OpenAI](#) since 2016 and was impressed by their steady progress. In mid-2022, I was so excited about their work that I gave them a challenge: train an artificial intelligence to pass an Advanced Placement biology exam. Make it capable of answering questions that it hasn't been specifically trained for. (I picked AP Bio because the test is more than a simple regurgitation of scientific facts—it asks you to think critically about biology.) If you can do that, I said, then you'll have made a true breakthrough.

*"I watched in awe as they asked GPT, their AI model, **60 multiple-choice questions** from the AP Bio exam—and it got **59 of them right**. Then it wrote **outstanding answers to six open-ended questions** from the exam. We had an outside expert score the test, and GPT got a 5—the highest possible score, and the equivalent to getting an A or A+ in a college-level biology course."*

“Once it had aced the test, we asked it a non-scientific question: “What do you say to a father with a sick child?” It wrote a thoughtful answer that was probably better than most of us in the room would have given. The whole experience was stunning.

I knew I had just seen the most important advance in technology since the graphical user interface.”

GPT-4 visual input example, Solving École Polytechnique's Physics Exam:

User Answer question I.1.a. Think step-by-step.

I. Principe de la détection de rayonnement avec un bolomètre

Comme illustré sur la figure 1 un bolomètre est constitué d'un absorbeur qui reçoit le rayonnement que l'on désire détecter. Sa température T , supposée uniforme, est mesurée à l'aide d'un thermomètre incorporé, constitué d'un matériau conducteur dont la résistance $R(T)$ varie avec la température T ; cette variation est caractérisée par le coefficient $\alpha = \frac{1}{R} \frac{dR}{dT}$. L'ensemble possède la capacité thermique C_{th} .

Un barreau, conducteur thermique, homogène, de longueur L , de section S et de conductivité thermique λ et sans échanges thermiques latéraux, relie le bolomètre à un thermostat de température T_b fixe.

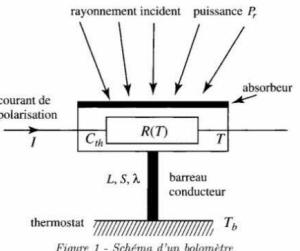


Figure 1 - Schéma d'un bolomètre

I.1. Dans cette première partie, on étudie le comportement du bolomètre en régime stationnaire, à la température de fonctionnement $T = T_0$.

I.1.a) Exprimer la température $T(x)$ en tout point du barreau conducteur repéré par son abscisse x ; on prendra $T(0) = T_0$ et $T(L) = T_b$.

GPT-4

To answer question I.1.a, we need to express the temperature $T(x)$ in every point of the conductive bar, which is characterized by the abscissa x . We are given that $T(0) = T_0$ and $T(L) = T_b$.

For a stationary regime, the heat equation for the one-dimensional heat conduction in the bar is:
 $d^2T/dx^2 = 0$

Integrating once, we get:
 $dT/dx = C_1$

Integrating a second time:
 $T(x) = C_1 x + C_2$

To determine the constants C_1 and C_2 , we use the boundary conditions:
 $T(0) = T_0$, which gives $C_2 = T_0$
 $T(L) = T_b$, which gives $C_1 = (T_b - T_0)/L$

Finally, the expression for $T(x)$ is:
 $T(x) = (T_b - T_0) * (x/L) + T_0$

This represents a linear temperature profile along the conductive bar, with a slope of $(T_b - T_0)/L$.

Table 15. Example prompt demonstrating GPT-4's visual input capability. The prompt consists of a question which requires understanding a physics problem with a diagram written in French and solving it using chain-of-thought prompting.

Sparks of Artificial General Intelligence: Early experiments with GPT-4

Sébastien Bubeck Varun Chandrasekaran Ronen Eldan Johannes Gehrke
Eric Horvitz Ece Kamar Peter Lee Yin Tat Lee Yuanzhi Li Scott Lundberg
Harsha Nori Hamid Palangi Marco Tulio Ribeiro Yi Zhang

Microsoft Research

Abstract

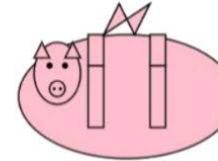
Artificial intelligence (AI) researchers have been developing and refining large language models (LLMs) that exhibit remarkable capabilities across a variety of domains and tasks, challenging our understanding of learning and cognition. The latest model developed by OpenAI, GPT-4 [Ope23], was trained using an unprecedented scale of compute and data. In this paper, we report on our investigation of an early version of GPT-4, when it was still in active development by OpenAI. We contend that (this early version of) GPT-4 is part of a new cohort of LLMs (along with ChatGPT and Google’s PaLM for example) that exhibit more general intelligence than previous AI models. We discuss the rising capabilities and implications of these models. We demonstrate that, beyond its mastery of language, GPT-4 can solve novel and difficult tasks that span mathematics, coding, vision, medicine, law, psychology and more, without needing any special prompting. Moreover, in all of these tasks, GPT-4’s performance is strikingly close to human-level performance, and often vastly surpasses prior models such as ChatGPT. Given the breadth and depth of GPT-4’s capabilities, we believe that it could reasonably be viewed as an early (yet still incomplete) version of an artificial general intelligence (AGI) system. In our exploration of GPT-4, we put special emphasis on discovering its limitations, and we discuss the challenges ahead for advancing towards deeper and more comprehensive versions of AGI, including the possible need for pursuing a new paradigm that moves beyond next-word prediction. We conclude with reflections on societal influences of the recent technological leap and future research directions.

<https://arxiv.org/abs/2303.12712>

You will be given a name of an object (such as Car, Chair, Elephant) and a letter in the alphabet. Your goal is to first produce a 1-line description of how that object can be combined with the letter in an image (for example, for an elephant and the letter J, the trunk of the elephant can have a J shape, and for the letter A and a house, the house can have an A shape with the upper triangle of the A being the roof). Following the short description, please create SVG code to produce this (in the SVG use shapes like ellipses, triangles etc and polygons but try to defer from using quadratic curves).

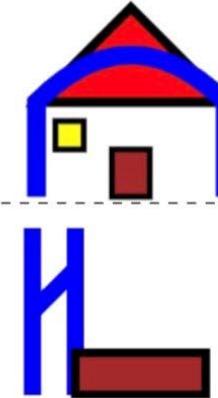
Round 1: A pig and the letter H.

Description: The pig can be lying on its back with its four legs forming the two vertical bars of the H, and its curly tail forming the horizontal bar.



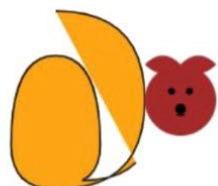
Round 2: A house and the letter U.

Description: A house with a U shape, where the two vertical lines of the U are the walls and the horizontal line is the floor. The roof is a triangle above the U.



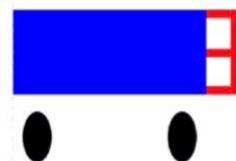
Round 3: A chair and the letter K.

Description: The back of the chair can have a K shape, with two vertical lines and a diagonal line connecting them. The seat of the chair can be a rectangle.



Round 4: A dog and the letter Q.

Description: The dog is sitting with its tail curled around its body, forming a Q shape. The Q is filled with orange and has a black outline.



Round 5: A car and the letter E.

Description: The car has an E shape on its front bumper, with the horizontal lines of the E being lights and the vertical line being the license plate.

Figure 2.6: Images combining alphabet letters and objects.

Contents

- Preliminaries:
 - Defining emergence
 - Next token prediction is “AI-complete”
 - Transformer attention
 - The residual pathway as a working memory
- Forms of emergence:
 - Emergence from multiple interacting layers
 - Emergence over multiple sampling operations
 - Emergence over multiple training iterations
 - Emergence in interaction with humans
 - Emergence in interaction with other AI modules
- Future directions

Preliminaries



Defining emergence

“Emergence is when quantitative changes in a system result in qualitative changes in behavior”

Historically: Systems with multiple interacting objects and feedback loops

Emergent Abilities of Large Language Models

Jason Wei¹

Yi Tay¹

Rishi Bommasani²

Colin Raffel³

Barret Zoph¹

Sebastian Borgeaud⁴

Dani Yogatama⁴

Maarten Bosma¹

Denny Zhou¹

Donald Metzler¹

Ed H. Chi¹

Tatsunori Hashimoto²

Oriol Vinyals⁴

Percy Liang²

Jeff Dean¹

William Fedus¹

¹Google Research ²Stanford University ³UNC Chapel Hill ⁴DeepMind

Reviewed on OpenReview: <https://openreview.net/forum?id=yzkSU5zdwD>

Abstract

Scaling up language models has been shown to predictably improve performance and sample efficiency on a wide range of downstream tasks. This paper instead discusses an unpredictable phenomenon that we refer to as *emergent abilities* of large language models. We consider an ability to be emergent if it is not present in smaller models but is present in larger models. Thus, emergent abilities cannot be predicted simply by extrapolating the performance of smaller models. The existence of such emergence raises the question of whether additional scaling could potentially further expand the range of capabilities of language models.

<https://arxiv.org/abs/2206.07682>

jasonwei@google.com

yitay@google.com

nlprishi@stanford.edu

craffel@gmail.com

barretzoph@google.com

sborgeaud@deepmind.com

dyogatama@deepmind.com

bosma@google.com

dennyyzhou@google.com

metzler@google.com

edchi@google.com

tashim@stanford.edu

vinyals@deepmind.com

pliang@stanford.edu

jeff@google.com

liamfedus@google.com



LaMDA GPT-3 Gopher Chinchilla PaLM Random

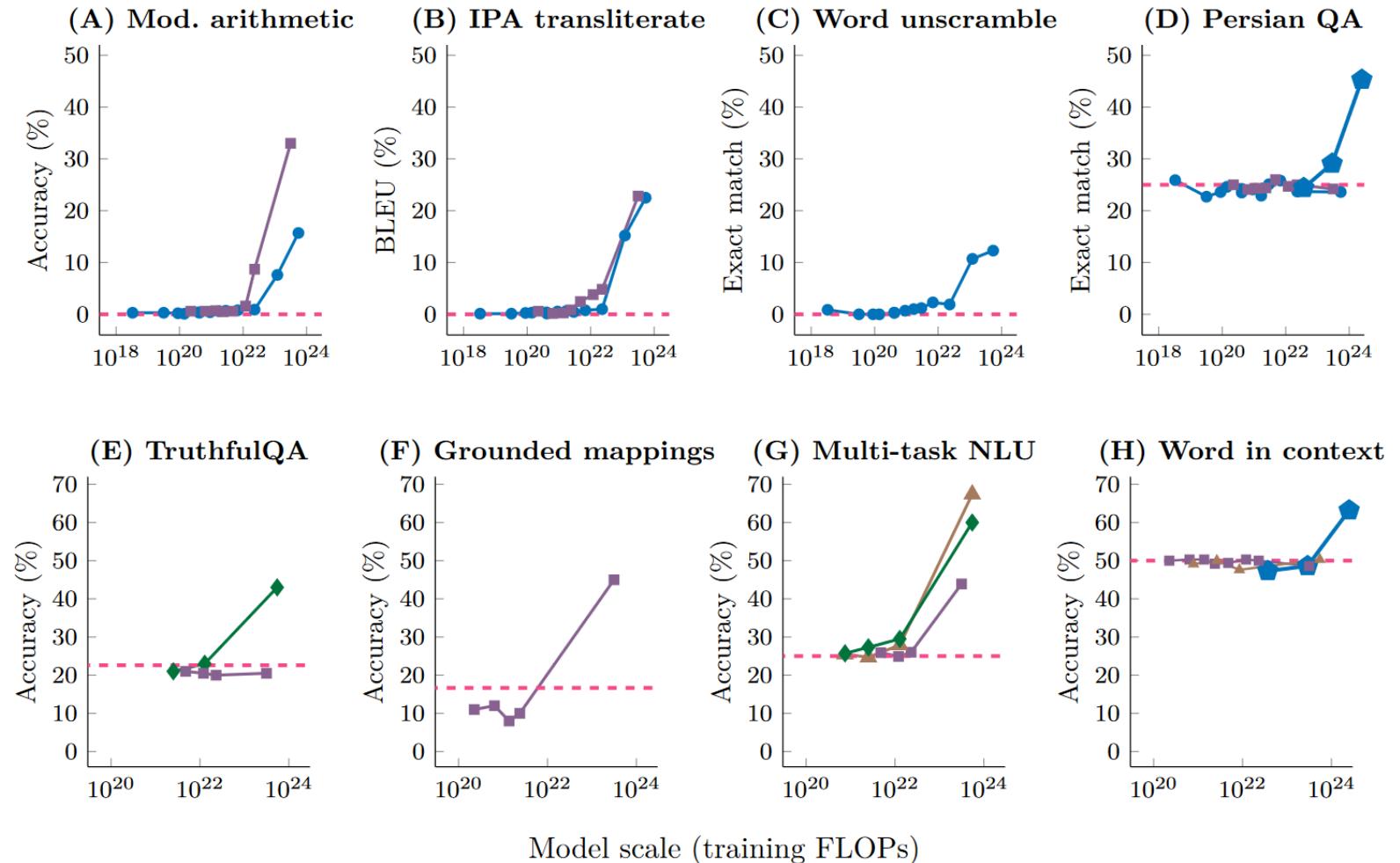


Figure 2: Eight examples of emergence in the few-shot prompting setting. Each point is a separate model. The ability to perform a task via few-shot prompting is emergent when a language model achieves random performance until a certain scale, after which performance significantly increases to well-above random. Note that models that used more training compute also typically have more parameters—hence, we show an analogous figure with number of model parameters instead of training FLOPs as the x -axis in Figure 11. A–D: BIG-Bench (2022), 2-shot. E: Lin et al. (2021) and Rae et al. (2021). F: Patel & Pavlick (2022). G: Hendrycks et al. (2021a), Rae et al. (2021), and Hoffmann et al. (2022). H: Brown et al. (2020), Hoffmann et al. (2022), and Chowdhery et al. (2022) on the WiC benchmark (Pilehvar & Camacho-Collados, 2019).

Next token prediction is “AI-complete”



David Chalmers @davidchalmers42 · Mar 19

...

who saw LLMs coming?

e.g. decades (or even 5+ years) ago, X said: when machine learning systems have enough compute and data to learn to predict text well, this will be a primary path to near-human-level AI.



175



131



941



725.2K



Jelle Zuidema (@wzuidema@sigmoid.social)

...

@wzuidema

Replying to [@davidchalmers42](#)

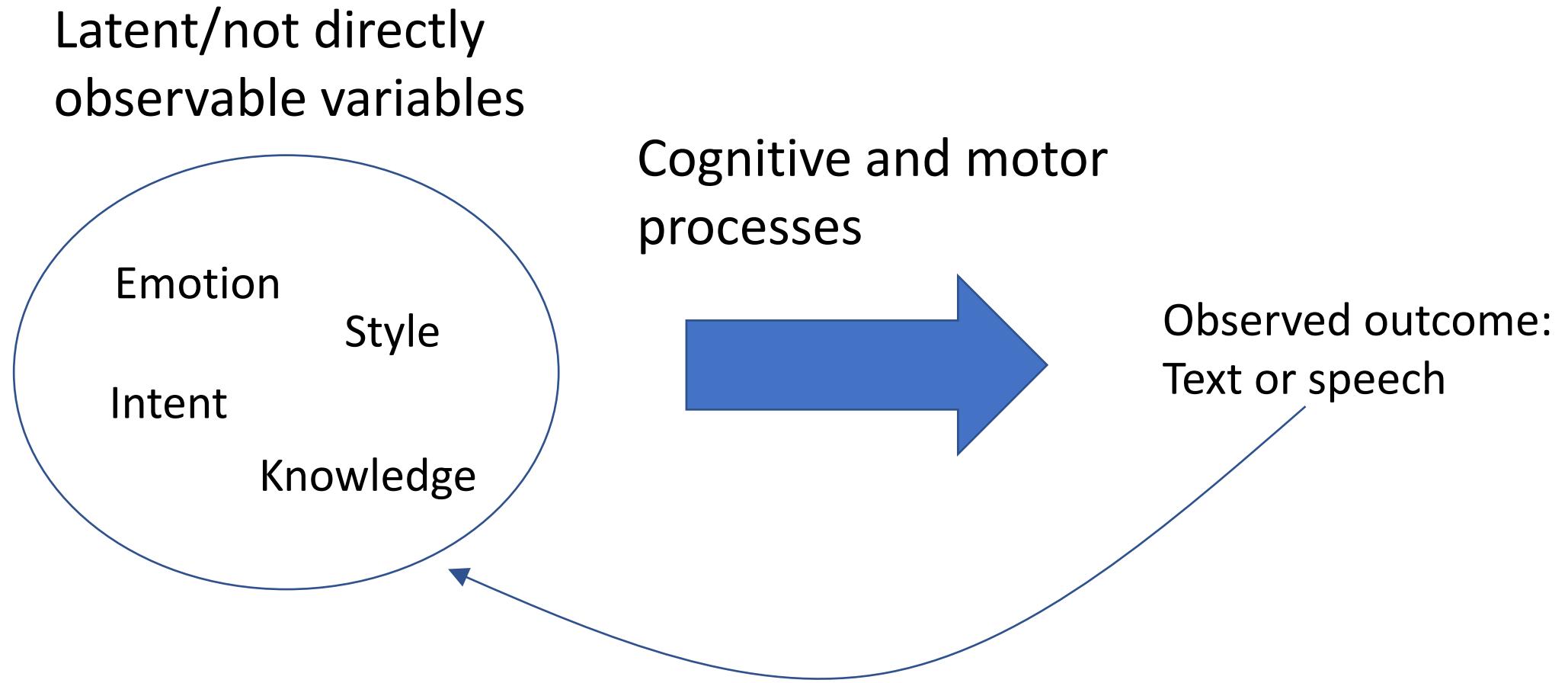
Many of us working on language models a decade ago realized that the problem of predicting the next word (A) was "AI Complete", i.e. required understanding of not only language but also the world (B). But few of us realized they would actually learn B from simply optimizing A.

9:47 PM · Mar 19, 2023 · 3,358 Views

Next token prediction is “AI-complete”

- Local optima: memorizing n-grams, superficial modeling of language
- Global optimum: modeling both language and the world
 - Or: modeling the generative process of which training texts are observable outcomes
- If we find ways to push the training loss low enough, it implies the emergence of such capability.

Modeling the language-generating process



Modeling the latent variables

- Each transformer block/layer operates on token embeddings (latent representations of the tokens computed by previous layers)
- So-called “hard attention” is Turing-complete, based on the capability to compute and manipulate the embeddings
- The transformer embeddings emerging from training often strongly relate to the latent variables of the modeled generative process

EMERGENT WORLD REPRESENTATIONS: EXPLORING A SEQUENCE MODEL TRAINED ON A SYNTHETIC TASK

Kenneth Li*
Harvard University

Aspen K. Hopkins
Massachusetts Institute of Technology

David Bau
Northeastern University

Fernanda Viégas
Harvard University

Hanspeter Pfister
Harvard University

Martin Wattenberg
Harvard University

ABSTRACT

Language models show a surprising range of capabilities, but the source of their apparent competence is unclear. Do these networks just memorize a collection of surface statistics, or do they rely on internal representations of the process that generates the sequences they see? We investigate this question in a synthetic setting by applying a variant of the GPT model to the task of predicting legal moves in a simple board game, Othello. Although the network has no a priori knowledge of the game or its rules, we uncover evidence of an emergent nonlinear internal representation of the board state. Interventional experiments indicate this representation can be used to control the output of the network. By leveraging these intervention techniques, we produce “latent saliency maps” that help explain predictions.^[1]

“Although the network has no a priori knowledge of the game or its rules, we uncover evidence of an emergent nonlinear internal representation of the board state”

1 INTRODUCTION

Recent language models have shown an intriguing range of capabilities. Networks trained on a simple “next-word” prediction task are apparently capable of many other things, such as solving logic puzzles or writing basic code.^[2] Yet how this type of performance emerges from sequence predictions remains a subject of current debate.

Some have suggested that training on a sequence modeling task is inherently limiting. The arguments range from philosophical (Bender & Koller, 2020) to mathematical (Merrill et al., 2021). A common theme is that seemingly good performance might result from memorizing “surface statistics,” i.e., a long list of correlations that do not reflect a causal model of the process generating the sequence. This issue is of practical concern, since relying on spurious correlations may lead to problems on out-of-distribution data (Bender et al., 2021; Floridi & Chiratti, 2020).

On the other hand, some tantalizing clues suggest language models may do more than collect spurious correlations, instead building interpretable *world models*—that is, understandable models of the process producing the sequences they are trained on. Recent evidence suggests language models can develop internal representations for very simple concepts, such as color, direction (Abdou et al., 2021), Patel & Pavlick (2022), or tracking boolean states during synthetic tasks (Li et al., 2021) (see Related Work (section 6) for more detail).

A promising approach to studying the emergence of world models is used by Toshniwal et al. (2021), which explores language models trained on chess move sequences. The idea is to analyze the behavior of a standard language modeling architecture in a well-understood, constrained setting. The paper finds that these models learn to predict legal chess moves with high accuracy. Furthermore, by analyzing predicted moves, the paper shows that the model appears to track the board state. The authors stop short, however, of exploring the form of any internal representations. Such an

Evolutionary-scale prediction of atomic-level protein structure with a language model

ZEMING LIN , HALIL AKIN , ROSHAN RAO , BRIAN HIE , ZHONGKAI ZHU, WENTING LU, NIKITA SMETANIN, ROBERT VERKUIL , ORI KABELI , [..], AND ALEXANDER RIVES +5 authors [Authors Info & Affiliations](#)

SCIENCE • 16 Mar 2023 • Vol 379, Issue 6637 • pp. 1123-1130 • DOI:10.1126/science.adc2574

25,267



"As language models of protein sequences are scaled up to 15 billion parameters, an atomic-resolution picture of protein structure emerges in the learned representations."

Speedy structures from single sequences

Machine learning methods for protein structure prediction have taken advantage of the evolutionary information present in multiple sequence alignments to derive accurate structural information, but predicting structure accurately from a single sequence is much more difficult. Lin *et al.* trained transformer protein language models with up to 15 billion parameters on experimental and high-quality predicted structures and found that information about atomic-level structure emerged in the model as it was scaled up. They created ESMFold, a sequence-to-structure predictor that is nearly as accurate as alignment-based methods and considerably faster. The increased speed permitted the generation of a database, the ESM Metagenomic Atlas, containing more than 600 million metagenomic proteins.
—MAF

Abstract

Recent advances in machine learning have leveraged evolutionary information in multiple sequence alignments to predict protein structure. We demonstrate direct inference of full atomic-level protein structure from primary sequence using a large language model. As language models of protein sequences are scaled up to 15 billion parameters, an atomic-resolution picture of protein structure emerges in the learned representations. This results in an order-of-magnitude acceleration of high-resolution structure prediction, which enables large-scale structural characterization of metagenomic proteins. We apply this capability to construct the ESM Metagenomic Atlas by predicting structures for >617 million metagenomic protein sequences, including >225 million that are predicted with high confidence, which gives a view into the vast breadth and diversity of natural proteins.

Which latent variables matter for human-generated text?

- Emotion, intent, style affect most of the training data
- A particular fact only affects small portions of the data
- To maximize average perf. over all data, one should prioritize stylistic and emotional plausibility over factual correctness

Language Models Can Generate Human-Like Self-Reports of Emotion

Mikke Tavast
mikke.tavast@aalto.fi
Aalto University
Espoo, Finland

Anton Kunnari
anton.kunnari@helsinki.fi
University of Helsinki
Helsinki, Finland

Perttu Hämäläinen
perttu.hamalainen@aalto.fi
Aalto University
Espoo, Finland

ABSTRACT

Computational interaction and user modeling is presently limited in the domain of emotions. We investigate a potential new approach to computational modeling of emotional response behavior, by using modern neural language models to generate synthetic self-report data, and evaluating the human-likeness of the results. More specifically, we generate responses to the PANAS questionnaire with four different variants of the recent GPT-3 model. Based on both data visualizations and multiple quantitative metrics, the human-likeness of the responses increases with model size, with the largest Davinci model variant generating the most human-like data.

CCS CONCEPTS

- Human-centered computing → Empirical studies in HCI.

KEYWORDS

Language models, GPT-3, PANAS, emotion, affect

ACM Reference Format:

Mikke Tavast, Anton Kunnari, and Perttu Hämäläinen. 2022. Language Models Can Generate Human-Like Self-Reports of Emotion. In *27th International Conference on Intelligent User Interfaces (IUI '22 Companion), March 22–25, 2022, Helsinki, Finland*. ACM, New York, NY, USA, 4 pages. <https://doi.org/10.1145/3490100.3516464>

1 INTRODUCTION

Computational user modeling is advancing rapidly and can produce human-like predictions of user behavior and experience [7, 12, 15]. However, this is presently limited in the domain of emotions. Here, our aim is to investigate a potential new approach by using modern neural language models to generate synthetic self-report data about affect. We employ the recent GPT-3 model to generate responses to Positive and Negative Affect Schedule (PANAS), a widely used questionnaire designed to measure positive and negative affect [21]. GPT-3 is a large neural language model trained to predict the next word in a sequence [5]. The trained model takes as its input a piece of text—a prompt—and generates a continuation of desired length.

PANAS consists of 10 positive affect and 10 negative affect items: emotional words such as excited, proud, guilty, and upset. The task is to rate how much one has felt these states during a specified time period. In the original validation study [21] each of the 20 items was

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

IUI '22 Companion, March 22–25, 2022, Helsinki, Finland

© 2022 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-9145-0/22/03.

<https://doi.org/10.1145/3490100.3516464>

shown to strongly load on only one of the two largely uncorrelated factors, establishing the 10 item positive and negative scales. In HCI research, PANAS has been used to, for example, operationalize positive user experience [16, 19].

We are not aware of previous studies trying to directly predict psychological scale responses using language models. Other natural language processing (NLP) methods have been used, for example, to predict scale responses [3] and affective ratings of words [20], and a recent preprint compared transformer models to human data in a benchmark NLP task [13]. Other work has, for example, used transformer model representations to predict brain imaging data [6, 10, 18].

2 METHODS

We generated 150 completions to the 20 items of the PANAS scale with four GPT-3 models of increasing size: Ada, Babbage, Curie, and Davinci¹. The responses to the PANAS items were generated with a prompt that described a research interview (see Table 1). To increase the variability in the prompts, each of the 150 interviews had a unique "participant" (varying age, gender, job, and hobby), a short description of "who" is being interviewed.

After the participant information, the prompt contained three example responses to questions answered in a Likert scale. To minimize bias in the training examples, the "participant" gave the answers 1, 3, and 5 once in each interview. The same three examples were used throughout the experiment, but their order was randomized for every interview.

The 20 PANAS items were queried one-by-one, in random order. Once the model generated a completion to a item, everything starting from the first appearance of the string "Researcher:" or the first newline character was cut from the completion. If the completion generated something outside desired responses (1,2,3,4 or 5), the whole interview for the participant was run again. The number of errors in proportion to all of the completions are presented in Section 2.1. After trimming the completion, the item-response pair was saved and included in the prompt for the next item. The order of the previous item-response pairs in the prompt was randomized for every new completion. For response generation, we used a maximum response length of 64 tokens and the default OpenAI parameters (temperature=0.7, top_p=1.0,frequency_penalty=0,presence_penalty=0, best_of=1).

We downloaded a human PANAS reference dataset from Open Science Framework (osf.io). The dataset was originally collected for a study by Anvari and Lakens concerning methods of determining the smallest effect size of interest [2]. Here we use the datapoints designated as T1 in the dataset (<https://osf.io/3a5up/>, [1]).

¹Data and analysis code: <https://github.com/mtavast/gpt-panas>

Alec Radford¹ Rafal Jozefowicz¹ Ilya Sutskever¹

Early result: A “sentiment neuron” emerges when an LSTM is trained to predict the next character of IMDB reviews.

Abstract

We explore the properties of byte-level recurrent language models. When given sufficient amounts of capacity, training data, and compute time, the representations learned by these models include disentangled features corresponding to high-level concepts. Specifically, we find a single unit which performs sentiment analysis. These representations, learned in an unsupervised manner, achieve state of the art on the binary subset of the Stanford Sentiment Treebank. They are also very data efficient. When using only a handful of labeled examples, our approach matches the performance of strong baselines trained on full datasets. We also demonstrate the sentiment unit has a direct influence on the generative process of the model. Simply fixing its value to be positive or negative generates samples with the corresponding positive or negative sentiment.

1. Introduction and Motivating Work

Representation learning (Bengio et al., 2013) plays a critical role in many modern machine learning systems. Representations map raw data to more useful forms and the choice of representation is an important component of any application. Broadly speaking, there are two areas of research emphasizing different details of how to learn useful representations.

The supervised training of high-capacity models on large labeled datasets is critical to the recent success of deep learning techniques for a wide range of applications such as image classification (Krizhevsky et al., 2012), speech recognition (Hinton et al., 2012), and machine translation (Wu et al., 2016). Analysis of the task specific representations learned by these models reveals many fascinating properties (Zhou et al., 2014). Image classifiers learn a broadly useful hierarchy of feature detectors representing raw pixels as edges, textures, and objects (Zeiler & Fergus, 2014). In the field of computer vision,

it is now commonplace to reuse these representations on a broad suite of related tasks - one of the most successful examples of transfer learning to date (Oquab et al., 2014).

There is also a long history of unsupervised representation learning (Olshausen & Field, 1997). Much of the early research into modern deep learning was developed and validated via this approach (Hinton & Salakhutdinov, 2006) (Huang et al., 2007) (Vincent et al., 2008) (Coates et al., 2010) (Le, 2013). Unsupervised learning is promising due to its ability to scale beyond only the subsets and domains of data that can be cleaned and labeled given resource, privacy, or other constraints. This advantage is also its difficulty. While supervised approaches have clear objectives that can be directly optimized, unsupervised approaches rely on proxy tasks such as reconstruction, density estimation, or generation, which do not directly encourage useful representations for specific tasks. As a result, much work has gone into designing objectives, priors, and architectures meant to encourage the learning of useful representations. We refer readers to Goodfellow et al. (2016) for a detailed review.

Despite these difficulties, there are notable applications of unsupervised learning. Pre-trained word vectors are a vital part of many modern NLP systems (Collobert et al., 2011). These representations, learned by modeling word co-occurrences, increase the data efficiency and generalization capability of NLP systems (Pennington et al., 2014) (Chen & Manning, 2014). Topic modelling can also discover factors within a corpus of text which align to human interpretable concepts such as art or education (Blei et al., 2003).

How to learn representations of phrases, sentences, and documents is an open area of research. Inspired by the success of word vectors, Kiros et al. (2015) propose skip-thought vectors, a method of training a sentence encoder by predicting the preceding and following sentence. The representation learned by this objective performs competitively on a broad suite of evaluated tasks. More advanced training techniques such as layer normalization (Ba et al., 2016) further improve results. However, skip-thought vectors are still outperformed by supervised models which directly optimize the desired performance metric on a specific dataset. This is the case for both text classification

¹OpenAI, San Francisco, California, USA. Correspondence to: Alec Radford <alec@openai.com>.

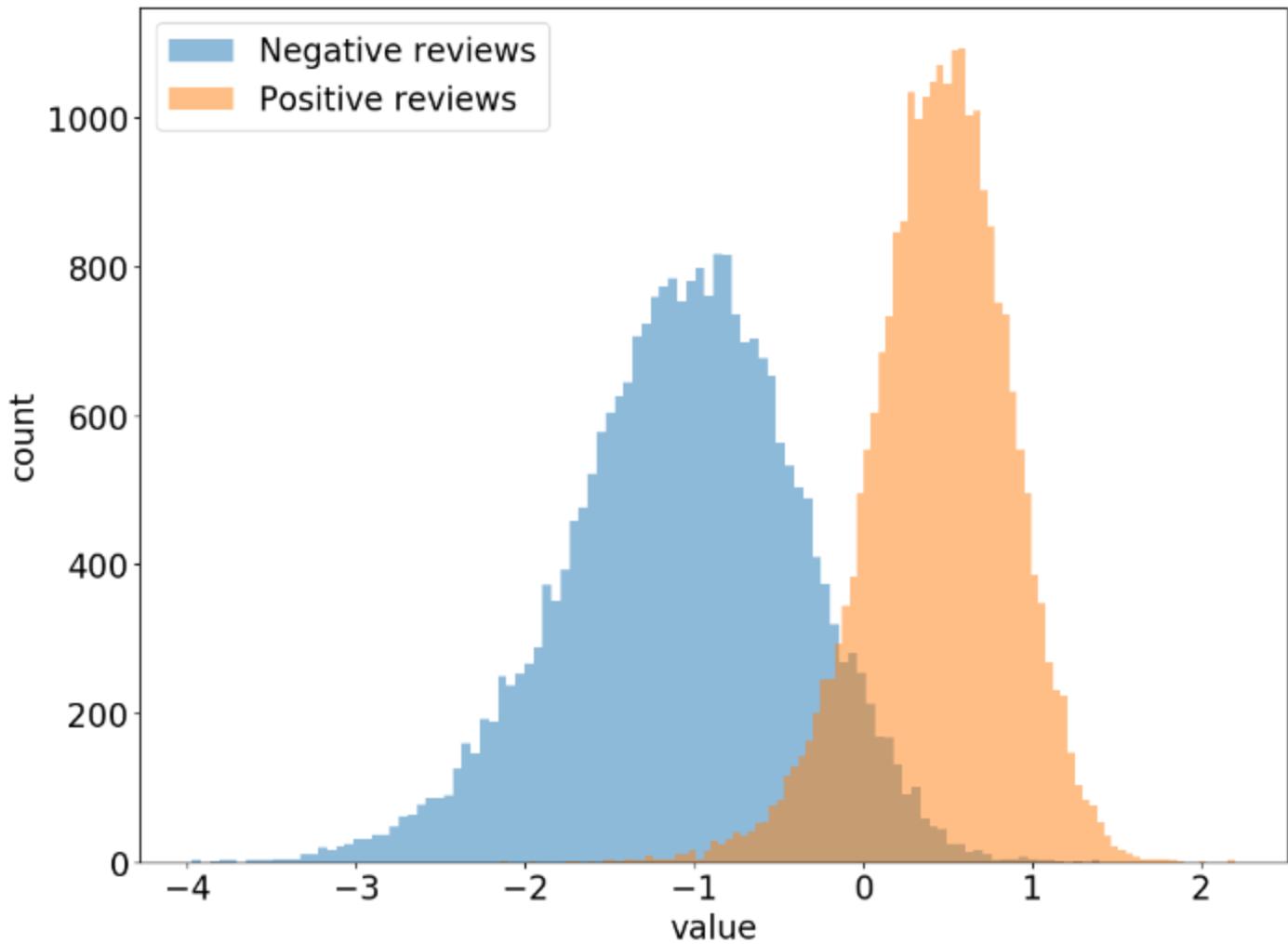


Figure 3. Histogram of cell activation values for the sentiment unit on IMDB reviews.

LLM token and sentence embedding spaces have directions corresponding to emotion concepts such as positive/negative valence.

Representations of emotion concepts: Comparison across pairwise, appraisal feature-based, and word embedding-based similarity spaces

Mijin Kwon¹, Tor D. Wager¹, Jonathan Phillips^{1, 2, 3}

¹ Department of Psychological and Brain Sciences, Dartmouth College

² Program in Cognitive Science, Dartmouth College

³ Department of Philosophy, Dartmouth College

Abstract

A question that has long interested cognitive scientists is how to best represent the different emotions we experience and attribute to others. For example, constructionist and appraisal theories propose that differences between emotions can be captured in part by their variation along a set of appraisal dimensions. More recently, researchers have used language models to capture the differences across different emotion terms. Both approaches allow us to represent emotions as occupying different locations in high-dimensional representational spaces. To ask how well these different approaches capture the similarity between emotion concepts, we collected pairwise similarity and appraisal feature ratings for 58 different emotion concepts and then employed representational similarity analysis to investigate the overlap between people's pairwise similarity judgments and emotion similarity in a 14-dimensional appraisal space and three word embedding spaces from two word2vec models (300 dimensions) and the newer GPT-3 model (12288 dimensions). The results indicate that while there is a high correlation between appraisal feature-based similarity and pairwise similarity judgments, word embedding-based similarity exhibits lower correlations, though GPT-3 showed much better performance than the word2vec models. Finally, characterizing the errors made by word embedding models showed that they can be largely attributed to an over-reliance on the valence of emotion concepts.

Keywords: emotion, concepts, representational similarity, appraisal, natural language processing, word2vec, GPT-3

Representation of Emotion

Humans use hundreds of different words to describe emotions. Like many concepts, emotion concepts can be thought of in terms of a network of associations; for example, 'elation' is more closely associated with 'joy' than with 'disgust'. Some of the fundamental questions in emotion research relate to the dimensionality and structure of emotion concepts, and what dimensions or categories might best describe them. For example, can the similarity and differences across concepts be described by a small number of dimensions (i.e., valence and arousal), emotion categories, or situational evaluations of perceived harms, benefits, intentions, and coping resources?

A great deal of research has pursued answers to these questions, with different theorists taking similar positions on some aspects but strongly diverging on others (for reviews of different emotion theories, see Barrett, 2016; Ekman et al., 1983; Ellsworth, 1994; Gendron & Barrett, 2009; Moore et

al., 2013). For example, while constructionist and appraisal theories of emotion differ in a number of respects, they share an understanding that differences between emotions are related to differences in the evaluations of the situations that give rise to those emotions.

In one study in this vein, Skerry and Saxe (2015) demonstrated that representations of emotion concepts, at both a behavioral and neural level, are best explained by a multidimensional space constructed by appraisal features. Therefore, these features may provide an important clue in understanding how people represent emotion concepts in relation to one another. Another study demonstrated that our conceptual knowledge of different emotion categories is mirrored by inferences we make about emotions from facial expressions – i.e., the more conceptually different two emotions are, the more different two facial emotions are perceived to be (Brooks & Freeman, 2018).

Representing Emotions using Natural Language Processing

In a separate line of research that has recently been developing in parallel, there has been a growing interest in asking how language models developed by NLP researchers can capture the similarity and differences in the semantics of different emotion terms (see, e.g., Seyeditabari et al., 2019). In one recent paper, for example, Jackson and colleagues (2019) argued for differences in the meaning of emotion terms across different languages on the basis of an analysis of co-lexification of emotion terms.

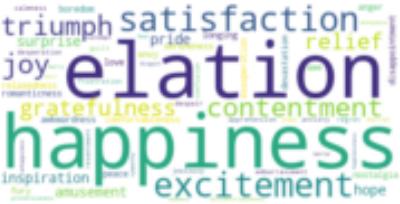
Present Research

In this paper, we ask how well these different approaches capture the reported similarity between emotion concepts. We collected pairwise similarity and 14 appraisal feature ratings for 58 different emotion concepts and then investigated the relationship between pairwise similarity judgments and pairs of emotion similarity in a 14-dimensional appraisal space and the three word embedding spaces: two word2vec models trained on two independent corpora, and the more recent GPT-3 model (3rd generation Generative Pre-trained Transformer).

3557



You spent the whole spring studying hard for university entrance exams. While on summer holiday, you receive the long-awaited results: you were accepted into the study programme of your dreams!



An important project threatens to fail because one of your colleagues has made a terrible mistake out of laziness with something that was their responsibility. Instead of owning up to the mistake, your colleague tries pin it on you.



A close friend has been battling cancer for a long time. You are spending an evening at home when you receive a phone call. Your friend passed away earlier that day.



You had a test last week, and it is vital for your graduation that you pass. After the test, you felt certain that you had flunked. You check the results from a notice board, and you are surprised to find that you got a good grade.



You arrive at your favourite summer spot. You notice that a group of kids has broken into the sauna and completely trashed it. The whole area is littered with garbage and broken bottles.



You have a dog, who is very dear to you. However, the dog is old and ill. You are at the vet clinic, bidding a final farewell to your dog.



You have travelled abroad to celebrate your birthday. As you return to your hotel alone after a tasty meal, your dear friend sends you a video message where he congratulates you warmly.



You have just had a great idea and you excitedly go tell your colleague about it. Your colleague walks straight into your boss's office and presents the idea as their own. Your boss praises your colleague's ingenuity to the entire office.



You attend a weekly meeting at work when you receive the news: a close colleague passed away suddenly last weekend due to a traffic accident.



Large Language Models Know How the Personality of Public Figures is Perceived by the General Public

Authors: Xubo Cao^{1*}, Michal Kosinski¹

Affiliations: Stanford University, Stanford, USA¹

*Correspondence to: xcao@stanford.edu

Abstract

We show that people's perceptions of public figures' personalities can be accurately predicted from their names' location in GPT-3's semantic space. We collected Big Five personality perceptions of 300 public figures from 600 human raters. Cross-validated linear regression was used to predict human perceptions from public figures names' embeddings extracted from GPT-3. Models' accuracy ranged from $r=.70$ to $.80$ without controls and from $r=.43$ to $.61$ when controlling for public figures' likability and demographics. Prediction models showed high face validity as revealed the personality-descriptive adjectives occupying their extremes. Our findings reveal that GPT-3 word embeddings capture signal pertaining to individual differences and intimate traits.

Keywords: personality, perceptions, GPT-3, large language models, foundational language models



Emergent social emotion, motivation, and behavior

- Throwing parties, asking other agents out on dates...
- LLM behavior rated more human-like than actual humans role-playing

<https://arxiv.org/abs/2304.03442>

Generative Agents: Interactive Simulacra of Human Behavior

Joon Sung Park
Stanford University
Stanford, USA
joonspk@stanford.edu

Joseph C. O'Brien
Stanford University
Stanford, USA
jobrien3@stanford.edu

Carrie J. Cai
Google Research
Mountain View, CA, USA
cjcai@google.com

Meredith Ringel Morris
Google Research
Seattle, WA, USA
merrie@google.com

Percy Liang
Stanford University
Stanford, USA
pliang@cs.stanford.edu

Michael S. Bernstein
Stanford University
Stanford, USA
msb@cs.stanford.edu

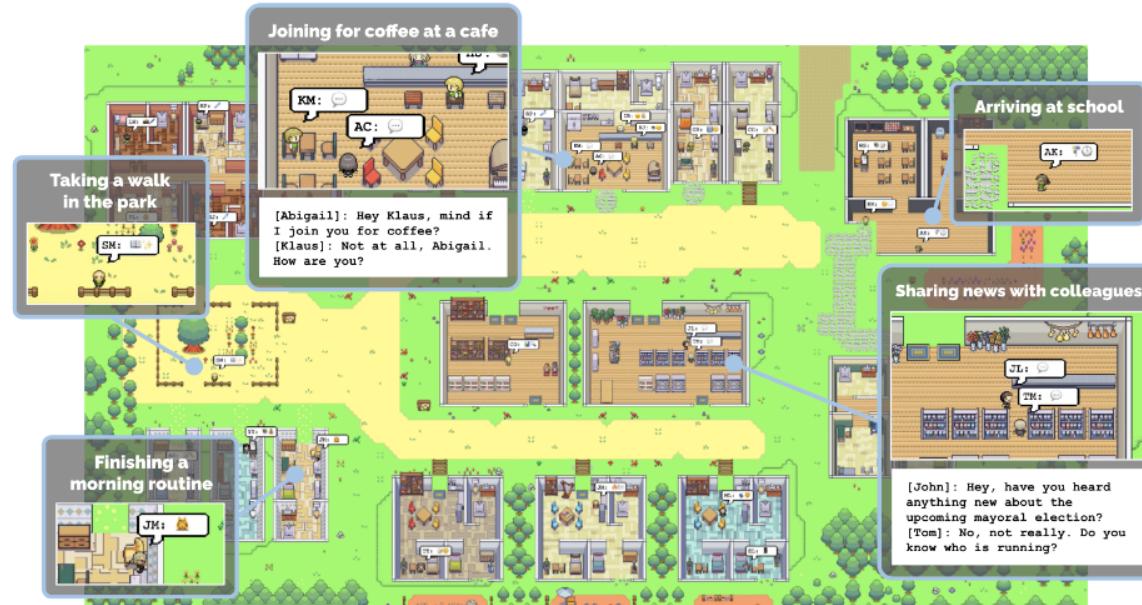


Figure 1: Generative agents create believable simulacra of human behavior for interactive applications. In this work, we demonstrate generative agents by populating a sandbox environment, reminiscent of The Sims, with twenty-five agents. Users can observe and intervene as agents they plan their days, share news, form relationships, and coordinate group activities.

ABSTRACT

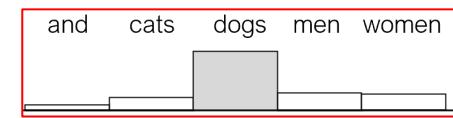
Believable proxies of human behavior can empower interactive applications ranging from immersive environments to rehearsal spaces for interpersonal communication to prototyping tools. In this paper, we introduce generative agents—computational software

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

arXiv, April, 2023,

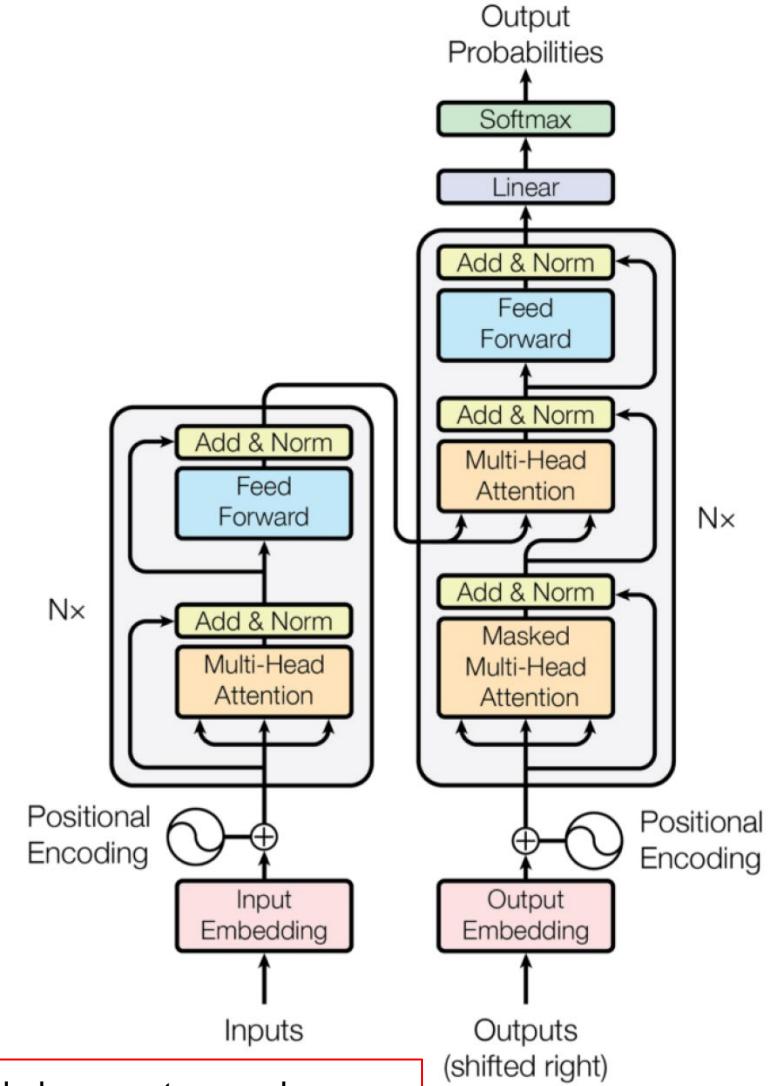
© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM. ACM ISBN xx-xxxx-xxxx-x/xx/xx...\$15.00
<https://doi.org/xx.xx/xx.xx>

agents that simulate believable human behavior. Generative agents wake up, cook breakfast, and head to work; artists paint, while authors write; they form opinions, notice each other, and initiate conversations; they remember and reflect on days past as they plan the next day. To enable generative agents, we describe an architecture that extends a large language model to store a complete record of the agent's experiences using natural language, synthesize those memories over time into higher-level reflections, and retrieve them dynamically to plan behavior. We instantiate generative agents to populate an interactive sandbox environment inspired by The Sims, where end users can interact with a small town of twenty five agents using natural language. In an evaluation, these generative agents produce believable individual and emergent social behaviors: for example, starting with only a single user-specified notion

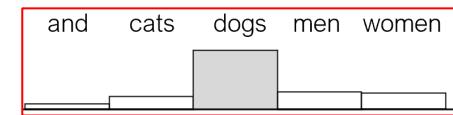


Transformer attention

- The original figure is confusing, let's try and do better

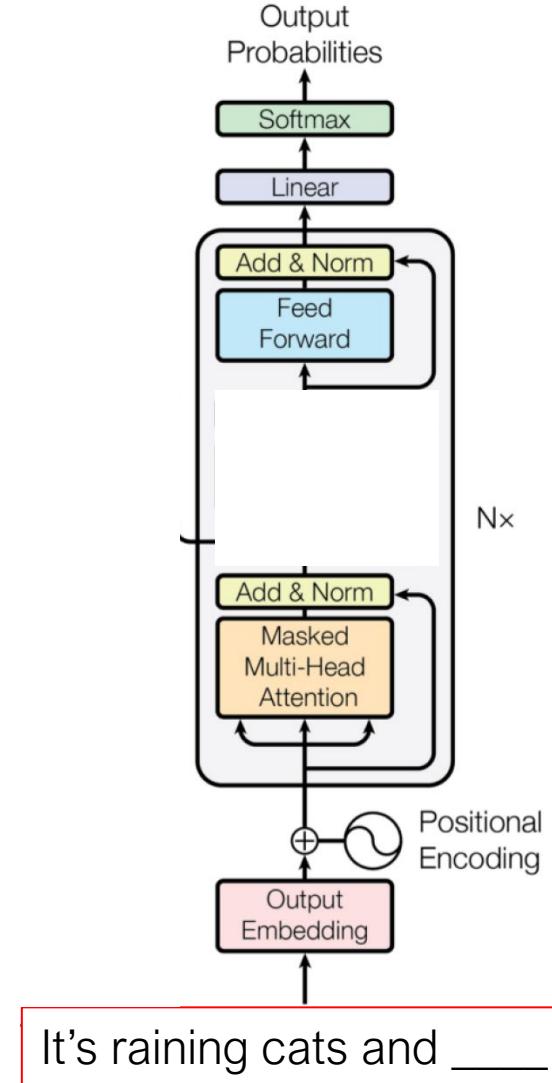


Transformer model architecture (Vaswani et al., 2017)



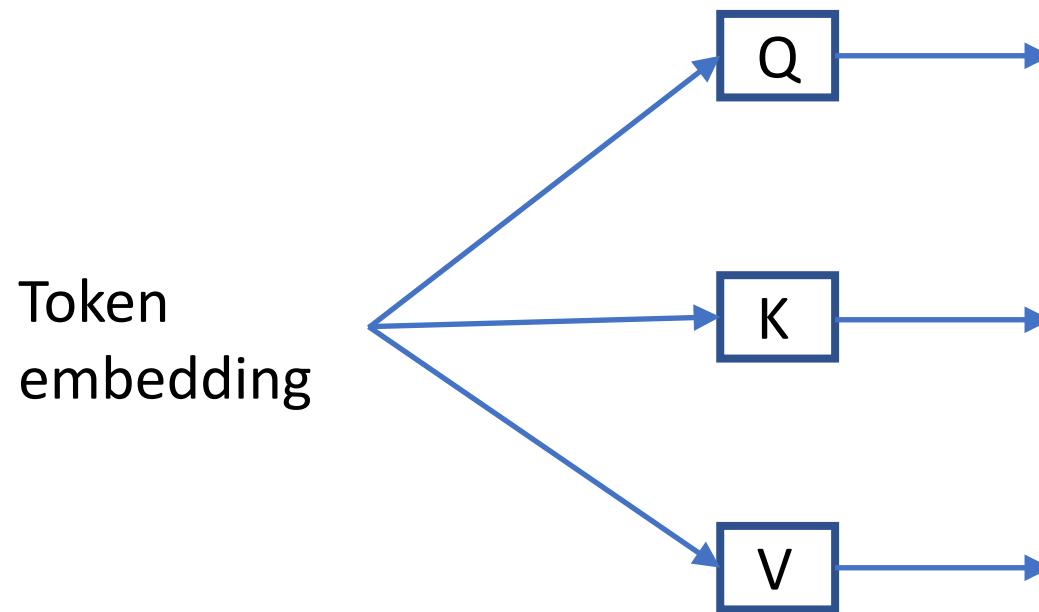
Transformer attention

- The original figure is confusing, let's try and do better
- I'll focus on this simplified “decoder-only” variant (GPT-2, GPT-3)



Transformer model architecture (Vaswani et al., 2017)

Key concept: query, key & value extractors



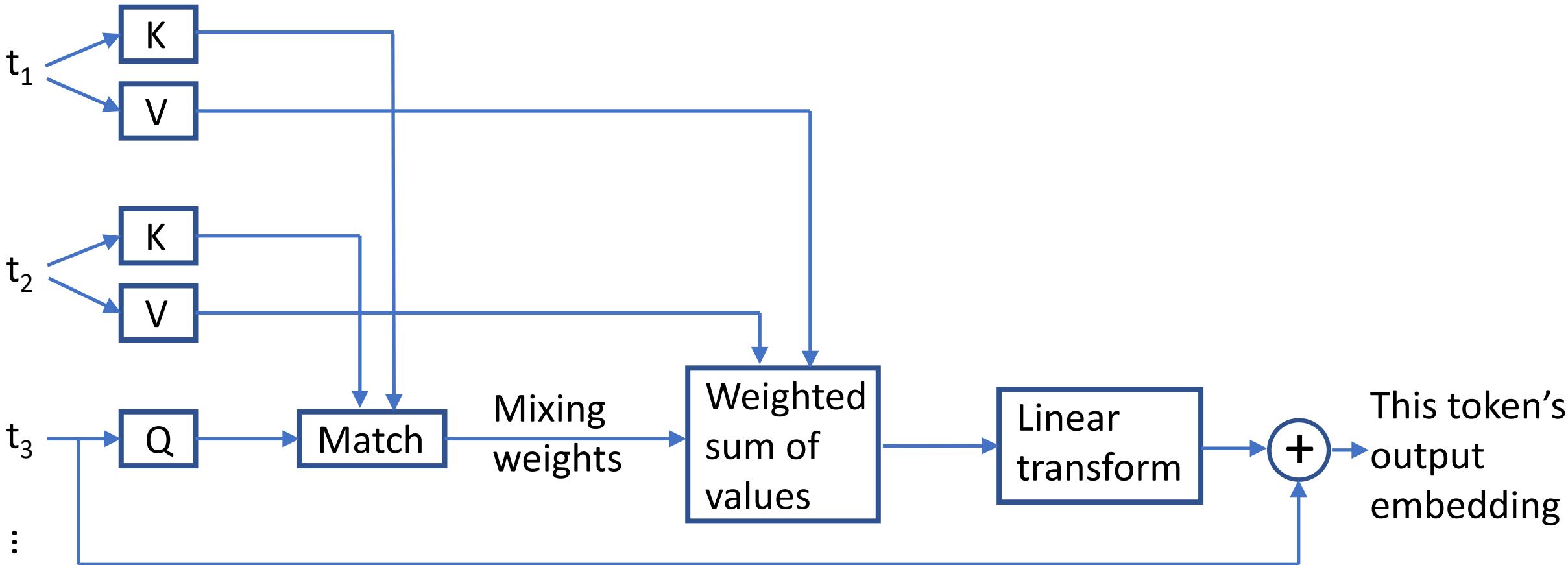
Query: For this token, what info are we looking for in other tokens?

Key: What info can this token provide for other tokens?

Value: The provided info

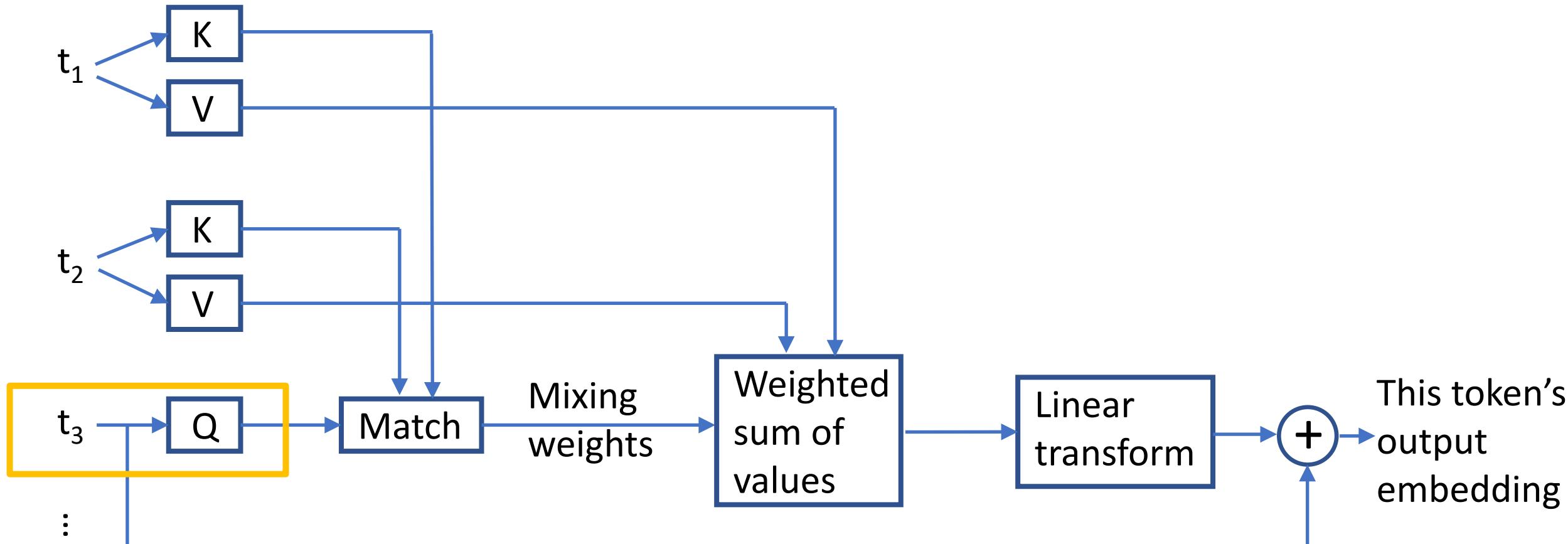
Key concept: Attention head

Token
embeddings



Key concept: Attention head

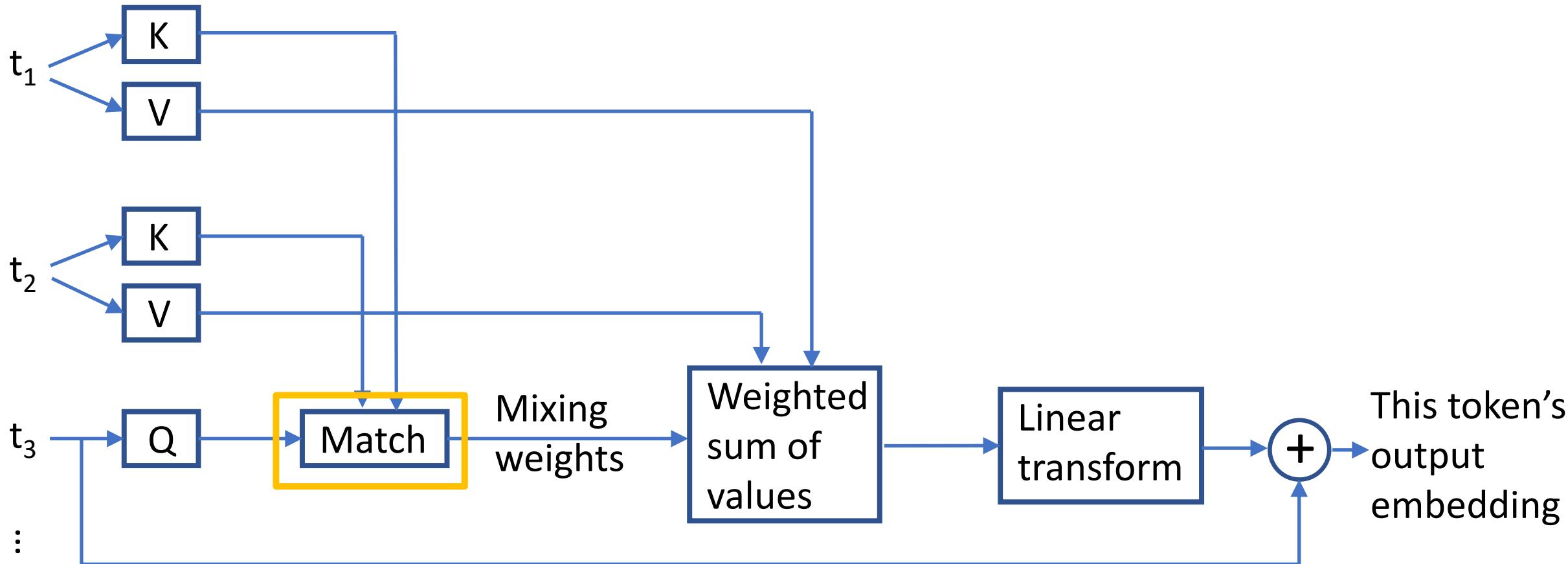
Token
embeddings



For each token, the head
extracts a query vector

Key concept: Attention head

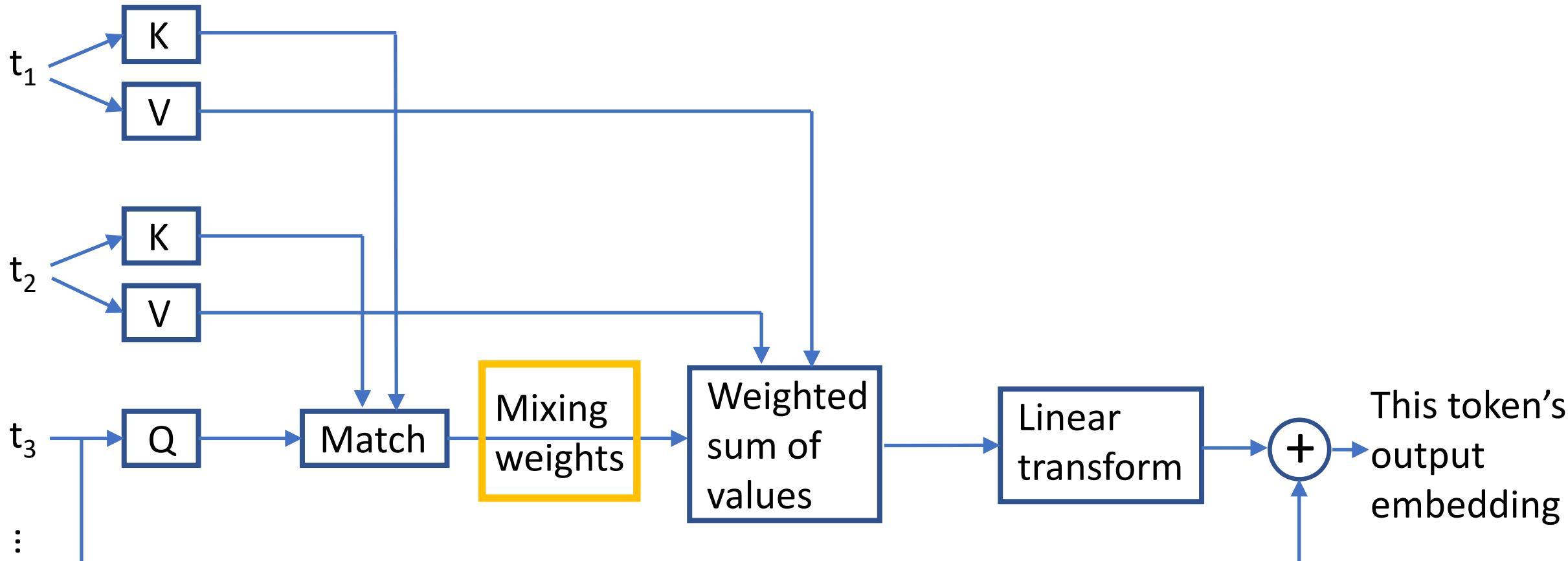
Token
embeddings



The query is then matched with
the keys of previous tokens

Key concept: Attention head

Token
embeddings

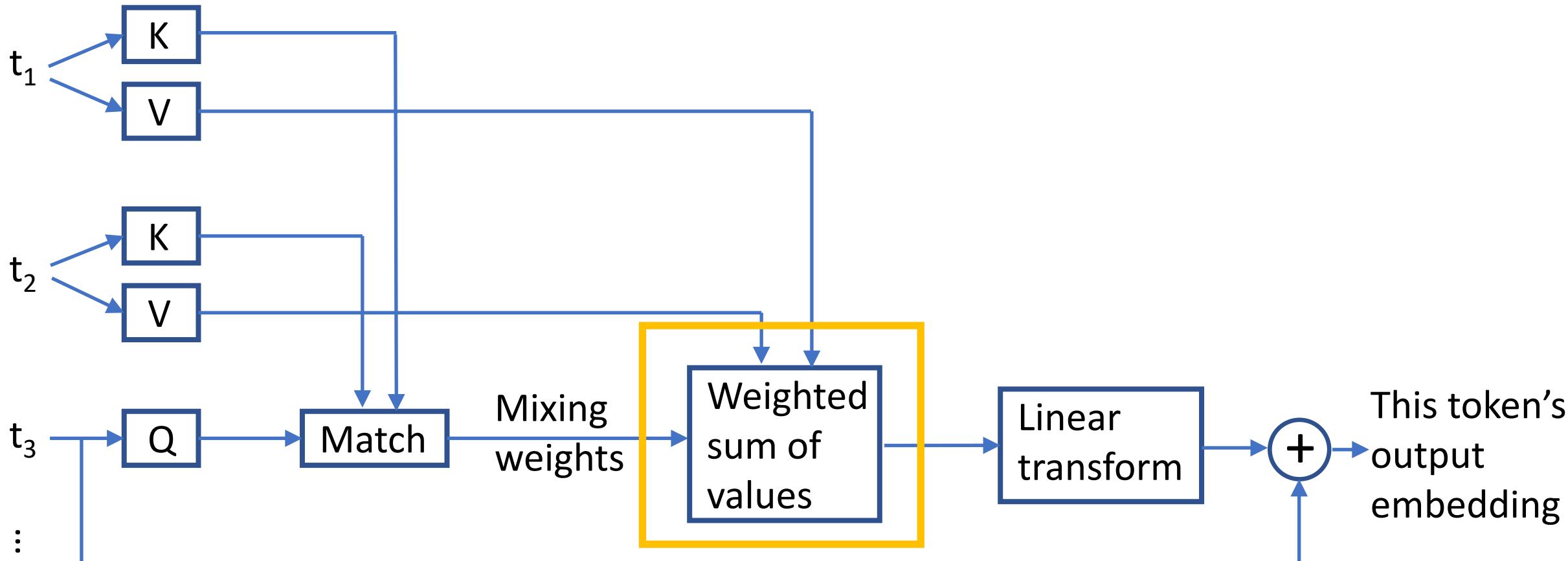


The matching produces mixing
weights that sum to 1

This token's
output
embedding

Key concept: Attention head

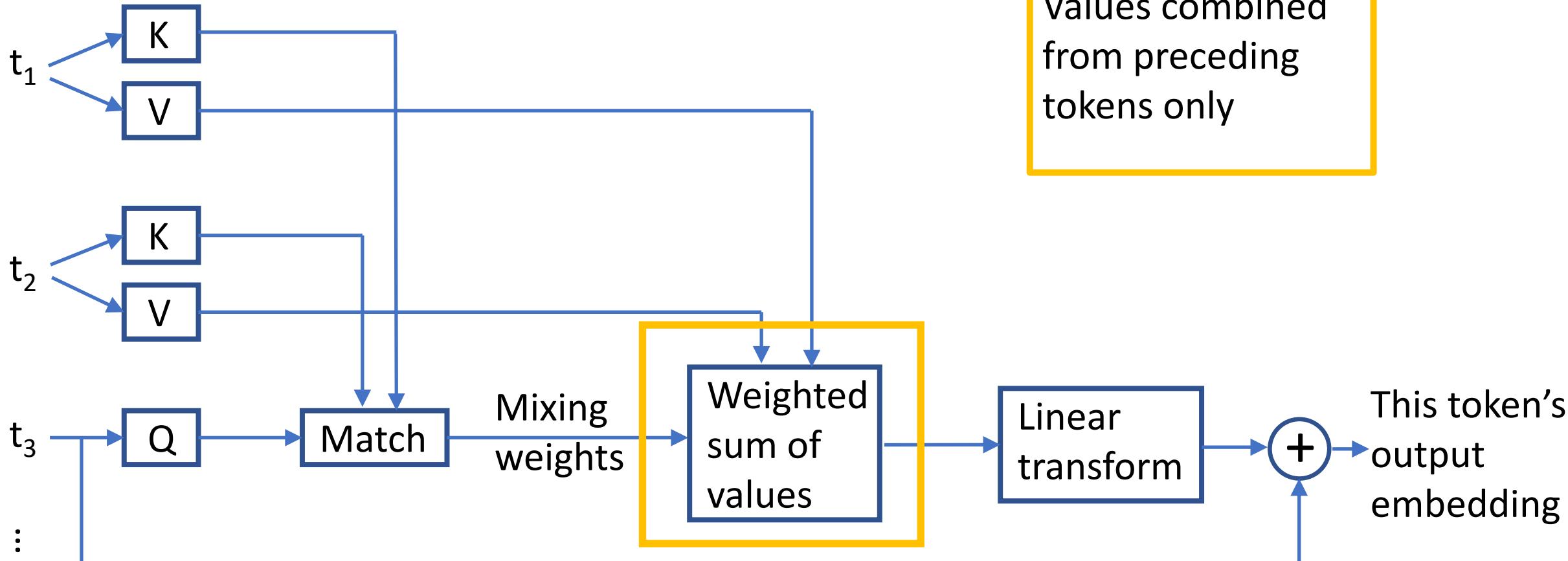
Token
embeddings



Using these weights, the
head sums the value vectors

Key concept: Attention head

Token
embeddings



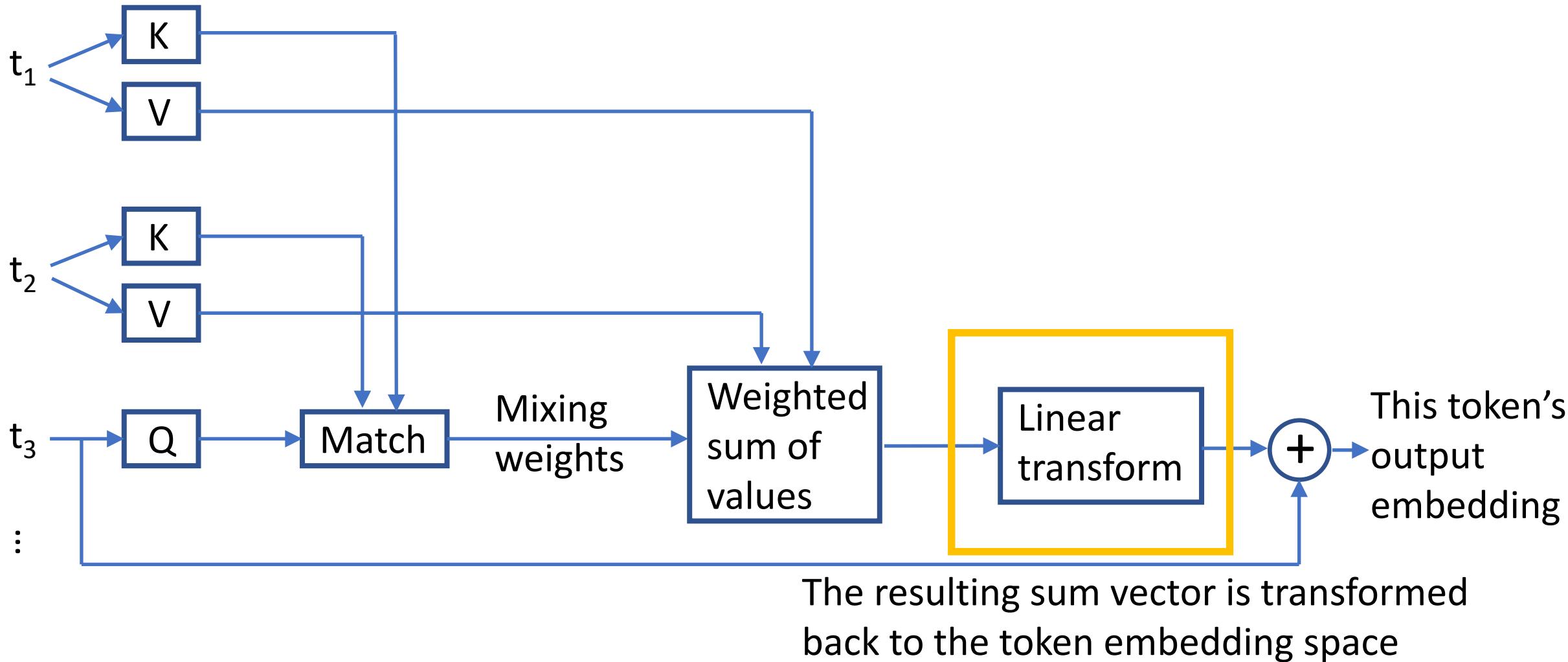
Using these weights, the
head sums the value vectors

Causal masking:
Values combined
from preceding
tokens only

This token's
output
embedding

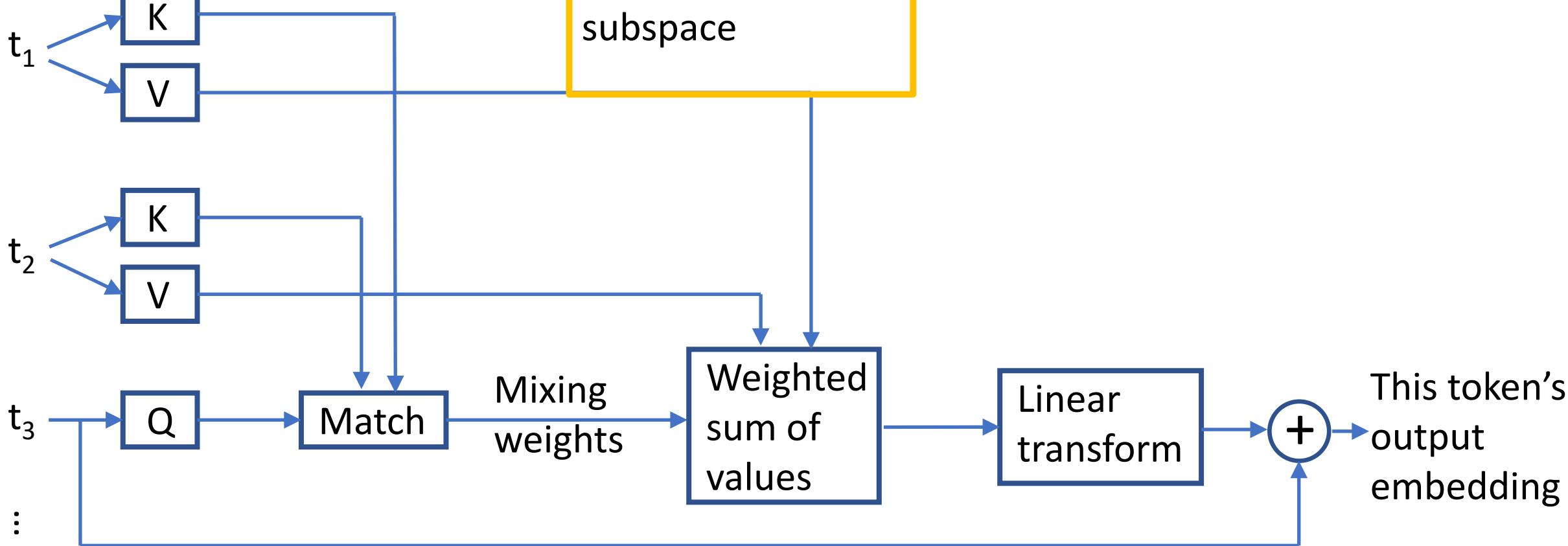
Key concept: Attention head

Token
embeddings



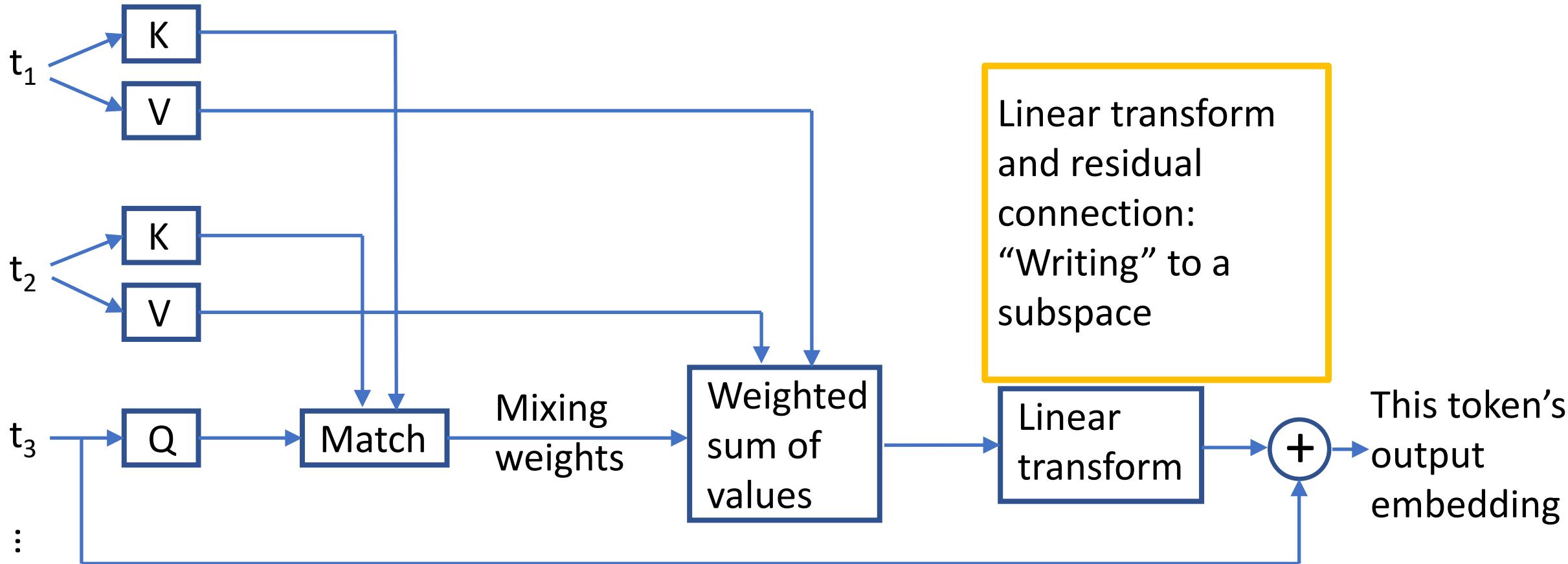
Key concept: Attention head

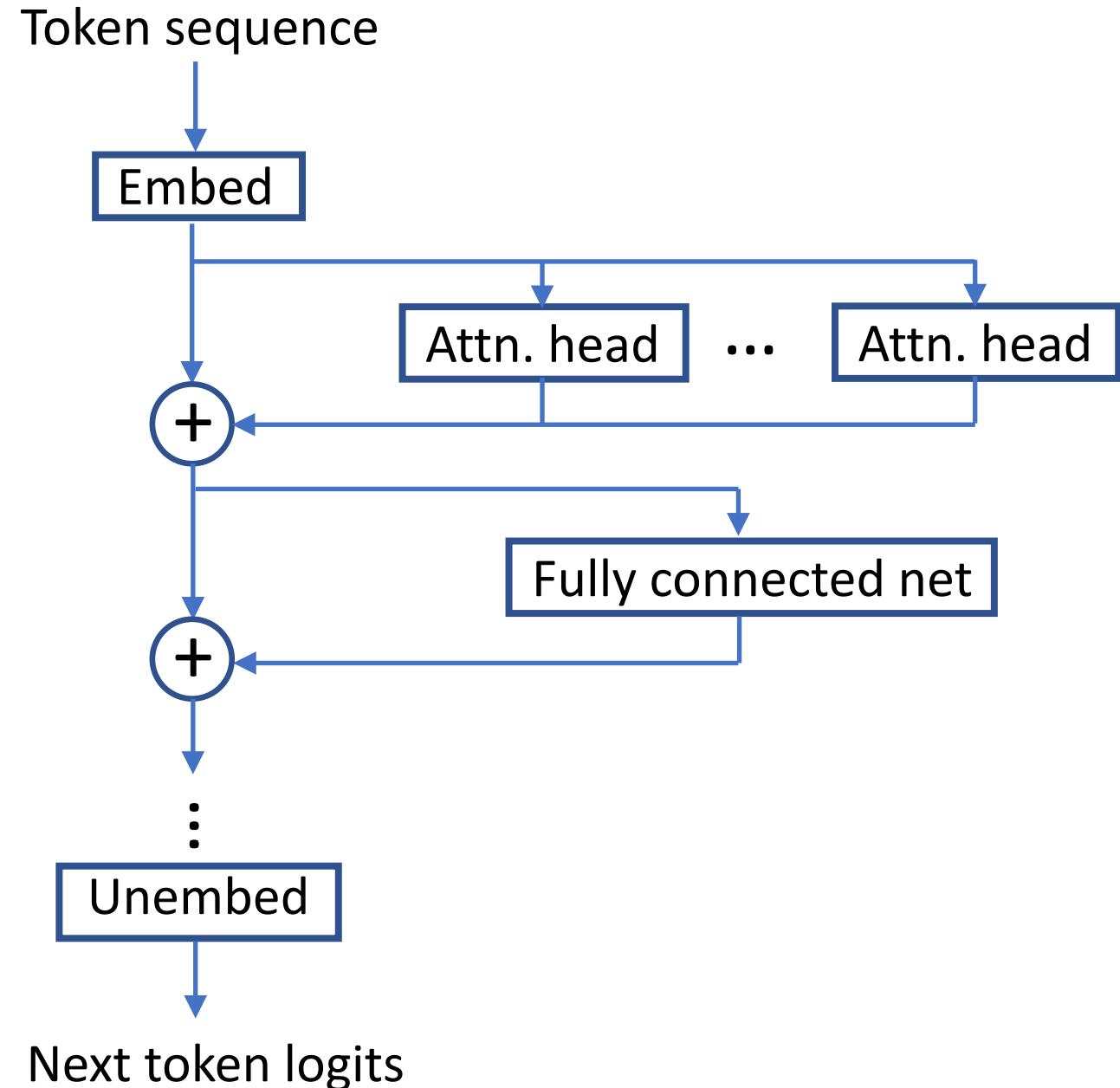
Token
embeddings

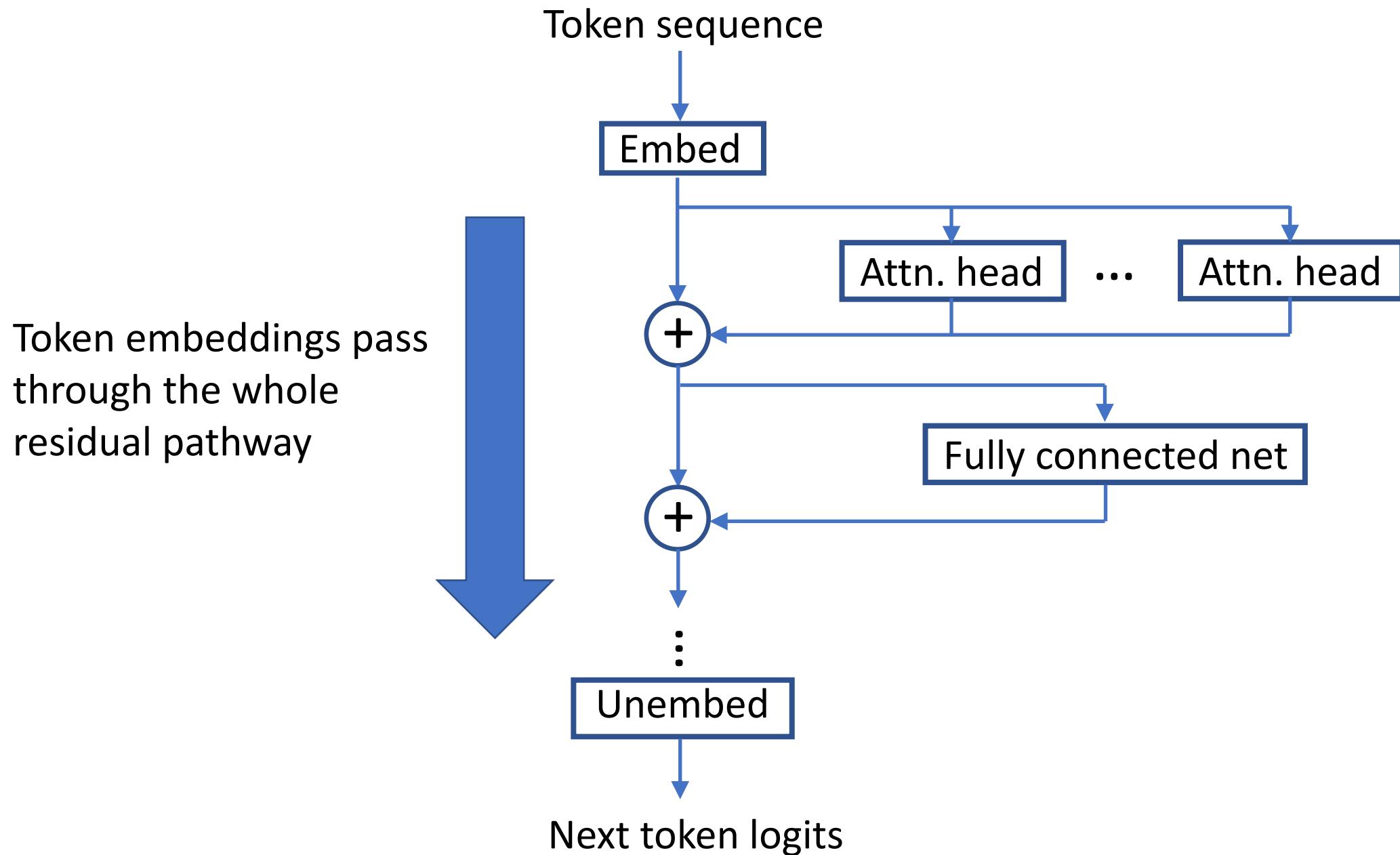


Key concept: Attention head

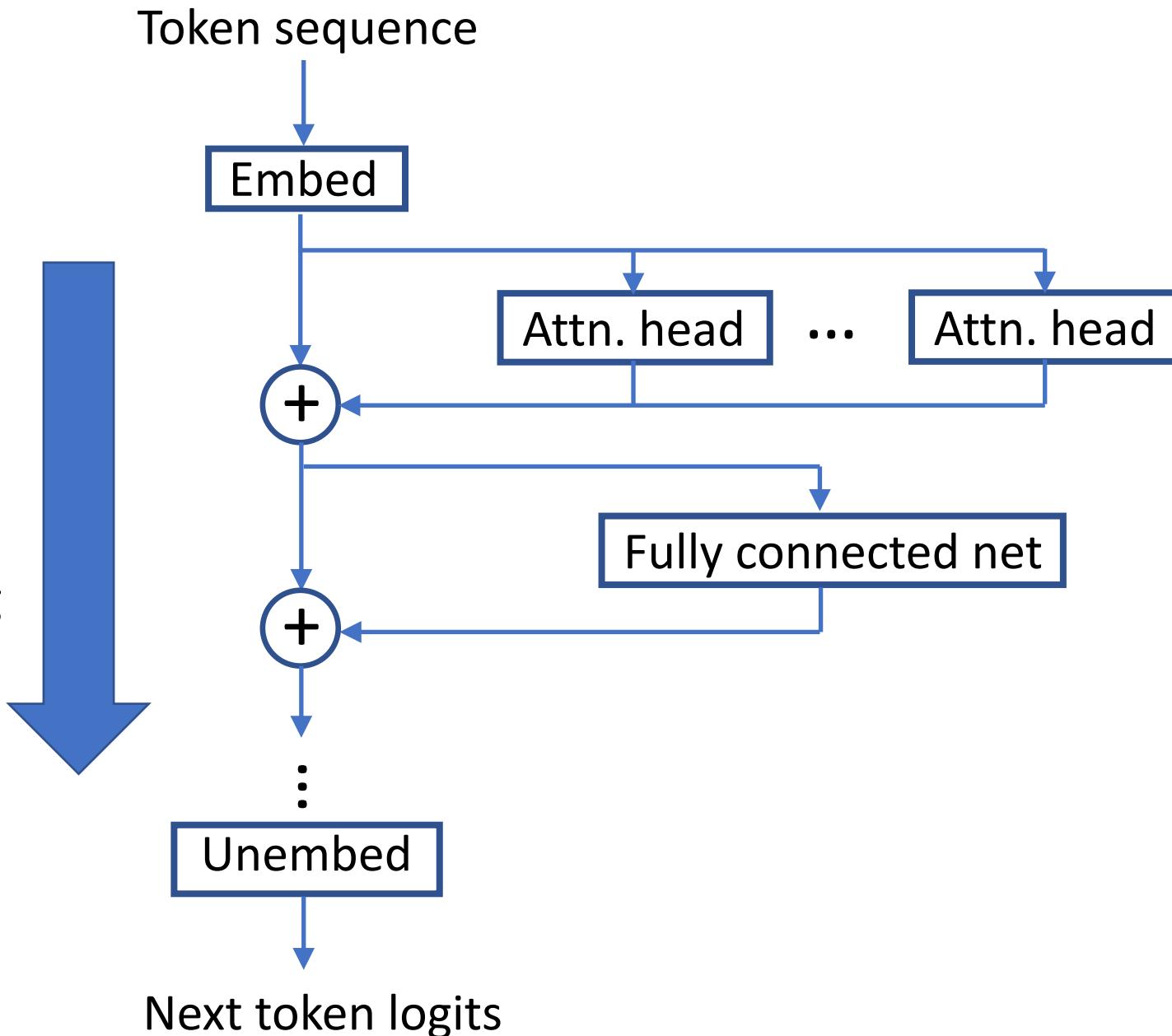
Token
embeddings



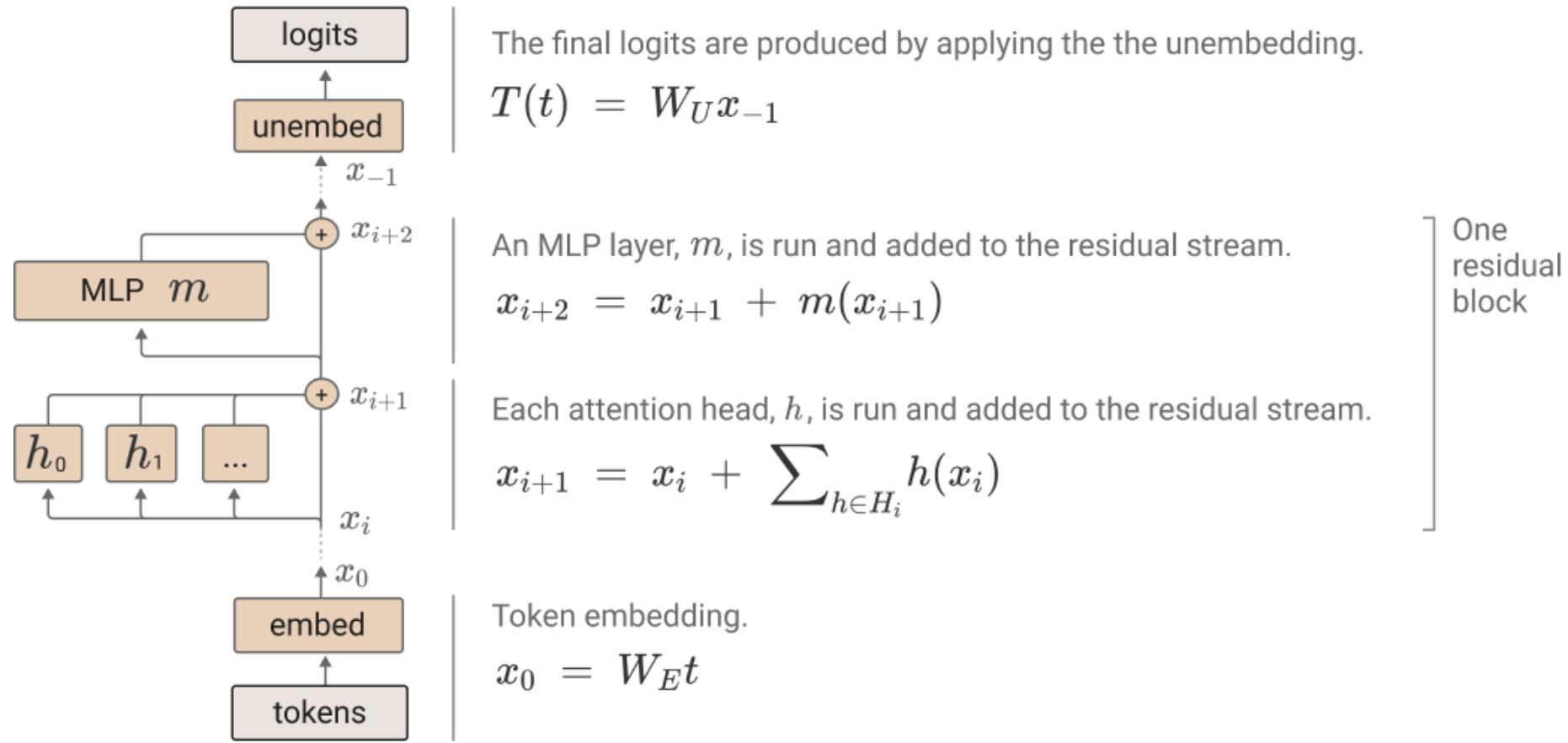




Essentially, the residual pathway works as a working memory, with attention heads reading and writing between residual subspaces



More detail



Safari File Edit View History Bookmarks Develop Window Help

colab.research.google.com

AI Understanding Prosperity Browse, share, and discuss... Please write a breaking ne... https://arxiv.org/pdf/1706.... https://raw.githubusercontent.com/gpt-dev.ipynb - Colaboratory makemore/makemore.py... CrossEntropyLoss — PyTorch

gpt-dev.ipynb

File Edit View Insert Runtime Tools Help

+ Code + Text

True

```
# version 4: self-attention!
torch.manual_seed(1337)
B,T,C = 4,8,32 # batch, time, channels
x = torch.randn(B,T,C)

# let's see a single Head perform self-attention
head_size = 16
key = nn.Linear(C, head_size, bias=False)
query = nn.Linear(C, head_size, bias=False)
k = key(x)    # (B, T, 16)
q = query(x) # (B, T, 16)
wei = q @ k

tril = torch.tril(torch.ones(T, T))
wei = torch.zeros((T,T))
wei = wei.masked_fill(tril == 0, float('-inf'))
wei = F.softmax(wei, dim=-1)
out = wei @ x

out.shape
```

[44] tril

```
tensor([[1., 0., 0., 0., 0., 0., 0., 0.],
        [1., 1., 0., 0., 0., 0., 0., 0.],
        [1., 1., 1., 0., 0., 0., 0., 0.],
        [1., 1., 1., 1., 0., 0., 0., 0.],
        [1., 1., 1., 1., 1., 0., 0., 0.],
        [1., 1., 1., 1., 1., 1., 0., 0.],
        [1., 1., 1., 1., 1., 1., 1., 0.],
        [1., 1., 1., 1., 1., 1., 1., 1.]])
```

1:06:17 / 1:56:19 • THE CRUX OF THE VIDEO: version 4: self-attention >

0s completed at 9:58 AM



Let's build GPT: from scratch, in code, spelled out.



Andrej Karpathy
160K subscribers

Subscribe

47K



Share

Clip

Save

...

Contents

- Preliminaries:
 - Defining emergence
 - Next token prediction is “AI-complete”
 - Transformer attention
 - The residual pathway as a working memory
- Forms of emergence:
 - Emergence from multiple interacting layers
 - Emergence over multiple sampling operations
 - Emergence over multiple training iterations
 - Emergence in interaction with humans
 - Emergence in interaction with other AI modules
- Future directions



Questions?

Contents

- Preliminaries:
 - Defining emergence
 - Next token prediction is “AI-complete”
 - Transformer attention
 - The residual pathway as a working memory
- Forms of emergence:
 - Emergence from multiple interacting layers
 - Emergence over multiple sampling operations
 - Emergence over multiple training iterations
 - Emergence in interaction with humans
 - Emergence in interaction with other AI modules
- Future directions



Emergence is typically caused by multiple interacting elements.

Emergence from multiple interacting layers

Deep networks in general, and transformers in particular

Multilayer feed-forward networks with as few as one hidden layer are universal approximators

- However, model flexibility can grow exponentially with depth, which may look like emergence: An exponential curve can appear flat up to an inflection point

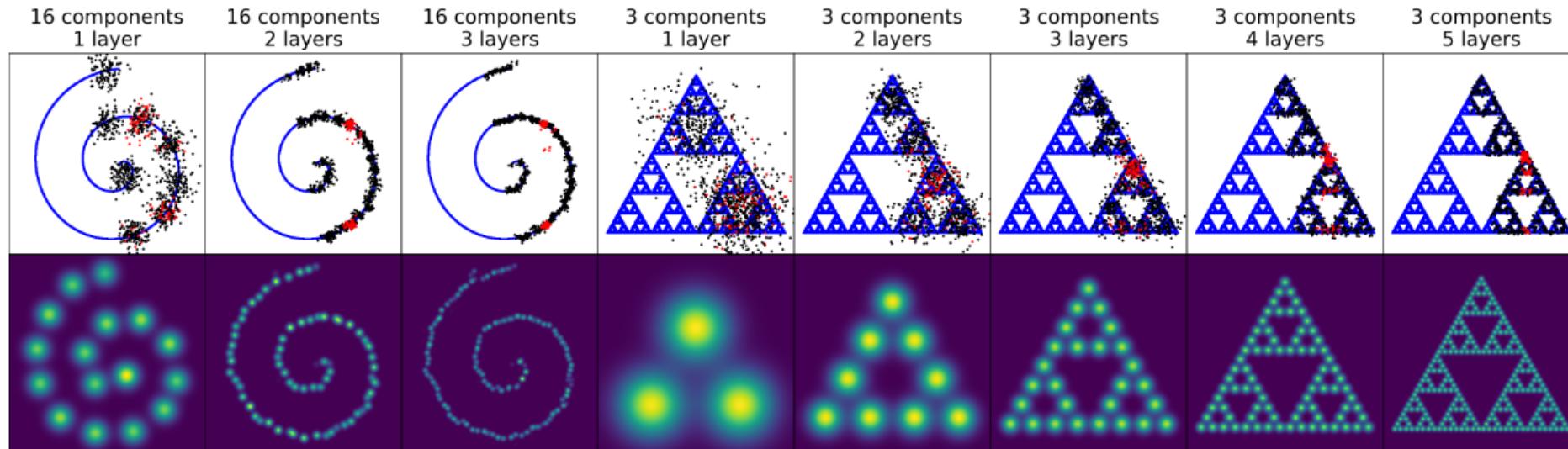
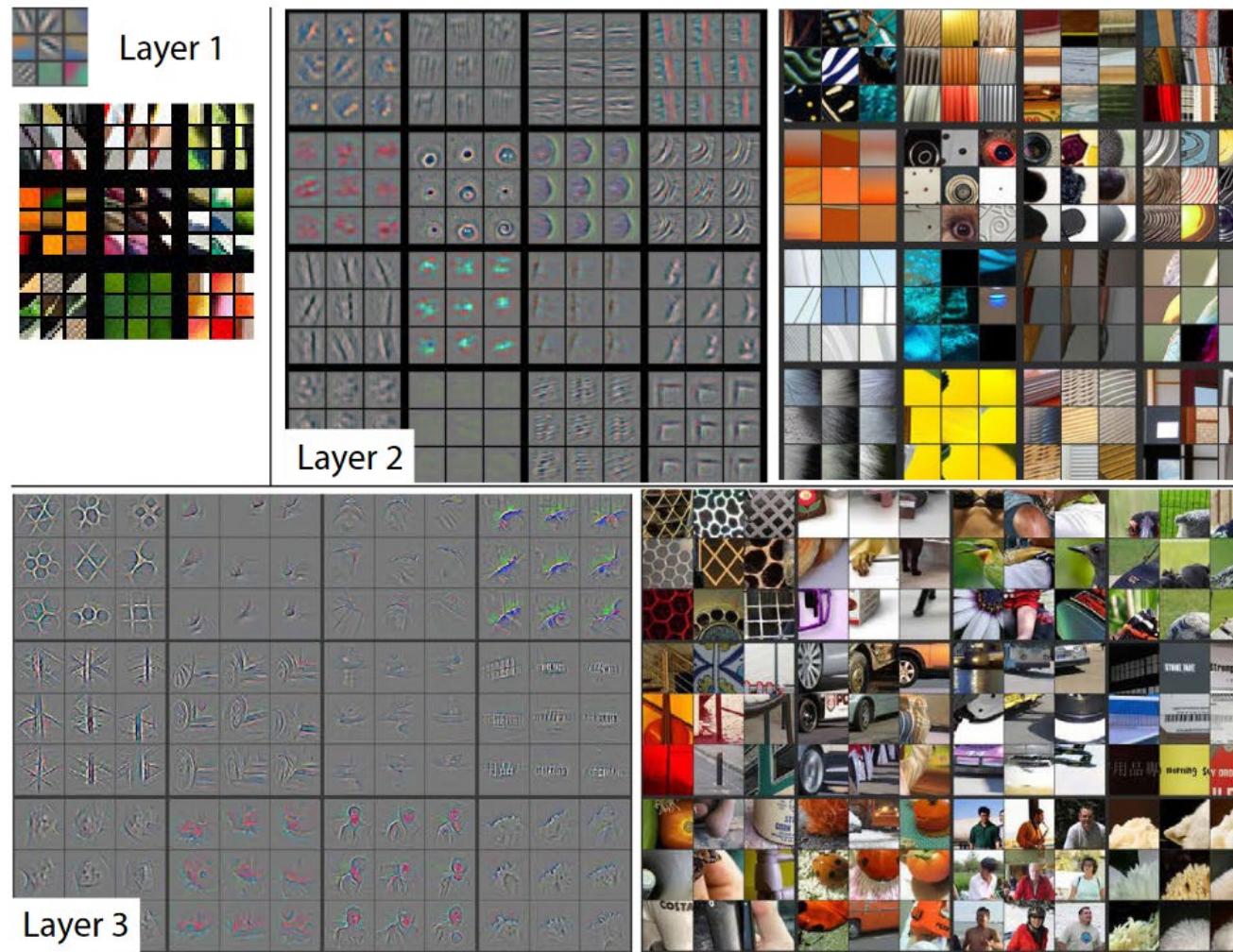


Figure 2: DRMM samples and density (likelihood) estimates with 2D training data. Samples can be conditioned with both inequalities (black, $ax + by + c > 0$) and equalities (red, $x + c = 0$). DRMM capacity can grow exponentially with depth, exploiting data self-similarity: The Sierpinski triangle model has three mixture components per layer, and a L -layer DRMM models 3^L density modes.

Capability to model compositional feature hierarchies only emerges with depth.
E.g., edge detectors → corner detectors → nose and eye detectors → face
detectors...



Zeiler & Fergus 2014
https://link.springer.com/chapter/10.1007/978-3-319-10590-1_53

What emerges when stacking attention operations? Is there something that cannot be done with a single transformer block with infinitely many parallel heads?



A Mathematical Framework for Transformer Circuits

- Zero layer transformers model bigram statistics.
- One layer attention-only transformers are an ensemble of bigram and “skip-trigram” models.
- So-called induction heads emerge in two layer attention-only transformers

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

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

* Core Research Contributor; † Core Infrastructure Contributor; ‡ Correspondence to colah@anthropic.com; Author contributions statement below.

Transformer^[1] language models are an emerging technology that is gaining increasingly broad real-world use, for example in systems like GPT-3^[2], LaMDA^[3], Codex^[4], Meena^[5], Gopher^[6], and similar models. However, as these models scale, their open-endedness and high capacity creates an increasing scope for unexpected and sometimes harmful behaviors. Even years after a large model is trained, both creators and users routinely discover model capabilities – including problematic behaviors – they were previously unaware of.

One avenue for addressing these issues is *mechanistic interpretability*, attempting to reverse engineer the detailed computations performed by transformers, similar to how a programmer might try to reverse engineer complicated binaries into human-readable source code. If this were possible, it could potentially provide a more systematic approach to explaining current safety problems, identifying new ones, and perhaps even anticipating the safety problems of powerful future models that have not yet been built. A previous project, the [Distill Circuits thread](#)^[7], has attempted to reverse engineer vision models, but so far there hasn't been a comparable project for transformers or language models.

In this paper, we attempt to take initial, very preliminary steps towards reverse-engineering transformers. Given the incredible complexity and size of modern language models, we have found it most fruitful to start with the simplest possible models and work our way up from there. Our aim is to discover simple algorithmic patterns, motifs, or frameworks that can subsequently be applied to larger and more complex models. Specifically, in this paper we will study *transformers with two layers or less which have only attention blocks* – this is in contrast to a large, modern transformer like GPT-3, which has 96 layers and alternates attention blocks with MLP blocks.

In-context Learning and Induction Heads

Induction heads

- Compositions of two attention heads (sequentially, not in parallel)
- Key to in-context learning
- Allow exact copying and completing of sequences that have occurred before.

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

* Core Research Contributor; † Core Infrastructure Contributor; ‡ Correspondence to colah@anthropic.com; Author contributions statement below.

As Transformer generative models continue to scale and gain increasing real world use [1, 2, 3, 4, 5], addressing their associated safety problems becomes increasingly important. *Mechanistic interpretability* – attempting to reverse engineer the detailed computations performed by the model – offers one possible avenue for addressing these safety issues. If we can understand the internal structures that cause Transformer models to produce the outputs they do, then we may be able to address current safety problems more systematically, as well as anticipating safety problems in future more powerful models.¹

In the past, mechanistic interpretability has largely focused on CNN vision models [6], but recently, we presented some very preliminary progress on mechanistic interpretability for Transformer language models [7]. Specifically, in our prior work we developed a mathematical framework for decomposing the operations of transformers, which allowed us to make sense of small (1 and 2 layer attention-only) models and give a near-complete account of how they function. Perhaps the most interesting finding was the *induction head*, a circuit whose function is to look back over the sequence for previous instances of the current token (call it A), find the token that came after it last time (call it B), and then predict that the same completion will occur again (e.g. forming the sequence [A][B] ... [A] → [B]). In other words, induction heads “complete the pattern” by copying and completing sequences that have occurred before. Mechanically, induction heads in our models are implemented by a circuit of two attention heads: the first head is a “previous token head” which copies information from the previous token into the next token, while the second head (the actual “induction head”) uses that information to find tokens preceded by the present token. For 2-layer attention-only models,² we were able to show precisely that induction heads implement this pattern copying behavior and appear to be the primary source of in-context learning.

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

In-context learning example

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:

We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

Exercise: continue the text

In-context learning example

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:

We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduddles.

Gray: Human text, Black: GPT-3 continuation

In-context learning example

A "whatpu" is a small, furry animal native to Tanzania. An example of a sentence that uses the word whatpu is:

We were traveling in Africa and we saw these very cute whatpus.

To do a "farduddle" means to jump up and down really fast. An example of a sentence that uses the word farduddle is:

One day when I was playing tag with my little sister, she got really excited and she started doing these crazy farduckles.

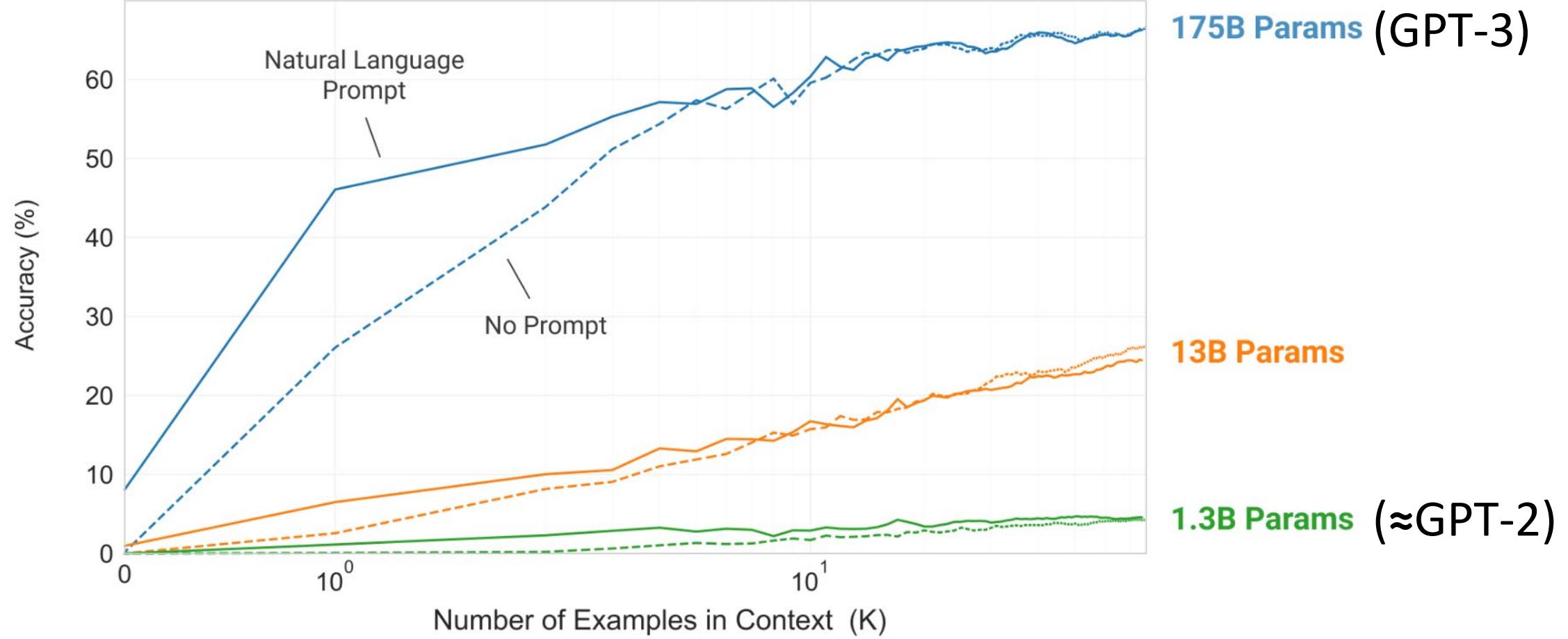
A "yalubalu" is a type of vegetable that looks like a big pumpkin. An example of a sentence that uses the word yalubalu is:

I was on a trip to Africa and I tried this yalubalu vegetable that was grown in a garden there. It was delicious.

Gray: Human text, Black: GPT-3 continuation



Large models generalize from only a few examples

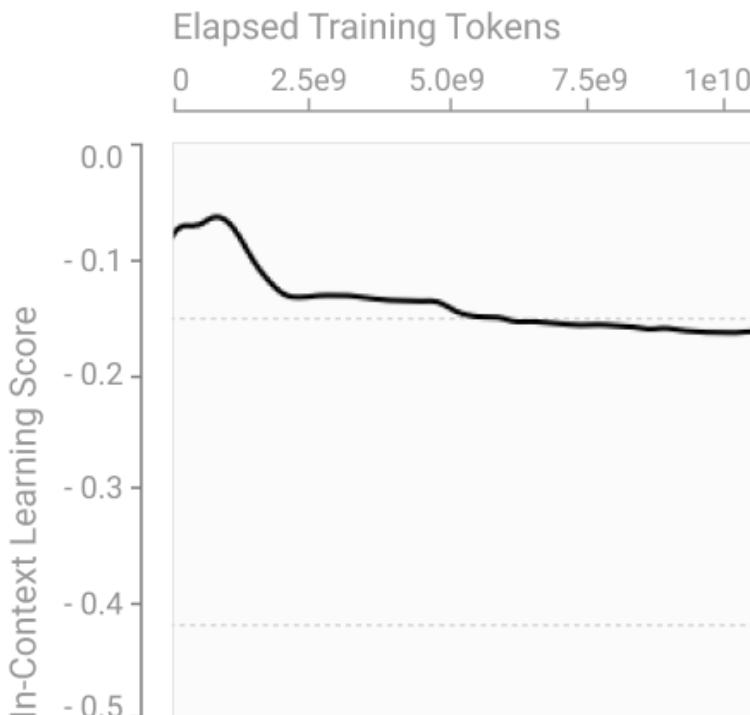


“Induction heads search over the context for previous examples of the present token. If they don't find it, they attend to the first token (in our case, a special token placed at the start), and do nothing.

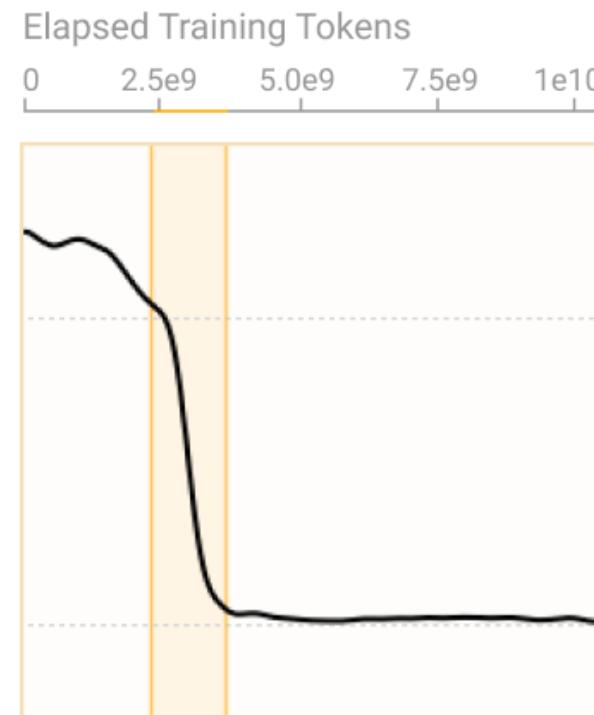
But if they do find it, they then look at the next token and copy it. This allows them to repeat previous sequences of tokens, both exactly and approximately.”

MODELS WITH MORE THAN ONE LAYER HAVE AN ABRUPT IMPROVEMENT IN IN-CONTEXT LEARNING

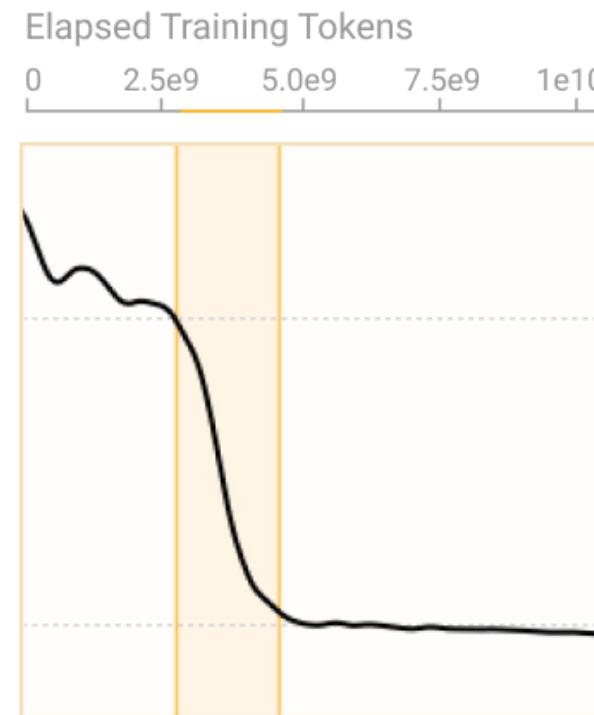
ONE LAYER
(ATTENTION-ONLY)



TWO LAYER
(ATTENTION-ONLY)



THREE LAYER
(ATTENTION-ONLY)



– We highlight the **“phase change”** period of training in plots to make visual comparison between plots easier. The highlighted region is selected for each model based on the derivative of in-context learning.

One-layer model
has no sudden improvement.

Models with **more than one layer**
have a **sudden improvement** in in-context learning

INDUCTION HEADS FORM IN PHASE CHANGE

Each line is an attention head, scored by the “prefix matching” evaluation introduced below.

ONE LAYER (ATTENTION-ONLY)

Elapsed Training Tokens

0 2.5e9 5.0e9 7.5e9 1e10

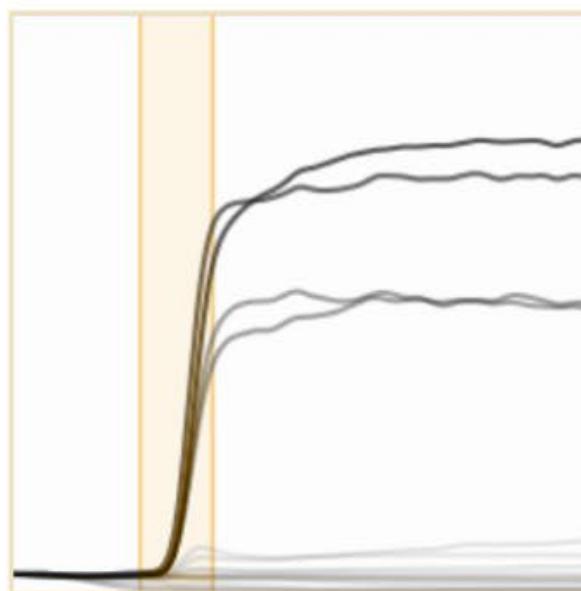


One-layer model
has no induction heads.

TWO LAYER (ATTENTION-ONLY)

Elapsed Training Tokens

0 2.5e9 5.0e9 7.5e9 1e10

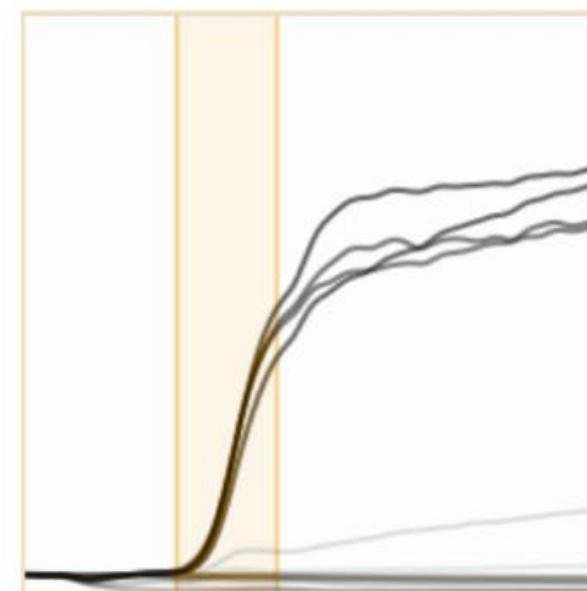


Models with **more than one layer**
have **induction heads form** during phase change.

THREE LAYER (ATTENTION-ONLY)

Elapsed Training Tokens

0 2.5e9 5.0e9 7.5e9 1e10



The highlighted **“phase change”** portion of training is the same area highlighted in previous plots. It is selected based on the derivative of the in-context score.





Emergence of reusable deep computations

A transformer trained on text can be used for vision, simply by **re-learning the linear embed and unembed layers**. All internal computations are kept fixed.

Kevin Lu,^{1,2} Aditya Grover,^{2,3} Pieter Abbeel,¹ Igor Mordatch⁴

¹ UC Berkeley, ² Facebook AI Research, ³ UCLA, ⁴ Google Brain
kzl@fb.com

Abstract

We investigate the capability of a transformer pretrained on natural language to generalize to other modalities with minimal finetuning – in particular, without finetuning of the self-attention and feedforward layers of the residual blocks. We consider such a model, which we call a Frozen Pretrained Transformer (FPT), and study finetuning it on a variety of sequence classification tasks spanning numerical computation, vision, and protein fold prediction. In contrast to prior works which investigate finetuning on the same modality as the pretraining dataset, we show that pretraining on natural language can improve performance and compute efficiency on non-language downstream tasks. Additionally, we perform an analysis of the architecture, comparing the performance of a random initialized transformer to a random LSTM. Combining the two insights, we find language-pretrained transformers can obtain strong performance on a variety of non-language tasks.

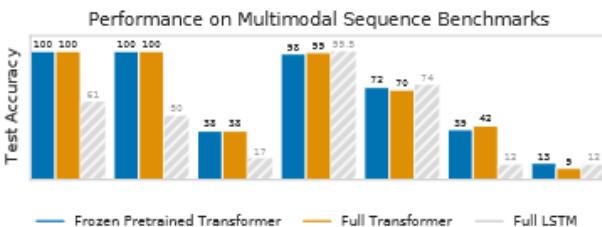


Figure 1: A *frozen* language-pretrained transformer (FPT) – without finetuning the self-attention and feedforward layers – can achieve strong performance compared to a transformer fully trained from scratch on a downstream *modality* on literature benchmarks (Tay et al. 2020; Rao et al. 2019). We show results on diverse classification tasks (see Section 2.1): numerical computation (Bit Memory/XOR, ListOps), image classification (MNIST, CIFAR-10, LRA), and protein fold prediction (Homology). We also show results for a fully-trained from-scratch LSTM as a baseline. Our code is available at: github.com/kzl/universal-computation

1 Introduction

The transformer architecture (Vaswani et al. 2017) has shown broad successes in deep learning, serving as the backbone of large models for tasks such as modeling natural language (Brown et al. 2020), images (Dosovitskiy et al. 2020), proteins (Jumper et al. 2021), and multimodal tasks comprising of both images and text (Lu et al. 2019; Radford et al. 2021). Inspired by these successes, we seek to explore the generalization capabilities of a transformer in transferring from one modality to another.

Classical approaches to sequence processing used recurrent neural network (RNN) approaches (Rumelhart, Hinton, and Williams 1985; Hochreiter and Schmidhuber 1997). In contrast, transformers utilize self-attention layers to extract features across tokens of a sequence, such as words (Vaswani et al. 2017) or image patches (Dosovitskiy et al. 2020). Furthermore, it has become common practice to train large models on unsupervised objectives before finetuning or evaluating zero-shot generalization on a downstream task. However, the downstream tasks that have been studied are generally restricted to the same modality as the original training set: for example, train GPT (Radford et al. 2018) on a large language corpus, and finetune on a small task-specific dataset. Our goal in this work is to investigate finetuning on modalities distinct from the training modality.

We hypothesize that transformers – namely the self-attention layers – can be pretrained on a data-rich modality (i.e. where data is plentiful, such as a language corpus) and identify feature representations that are useful for *arbitrary* data sequences, enabling downstream transfer to different modalities. In particular, we seek to investigate what pretrained language models (LMs) are capable of in terms of generalizing to other modalities with sequential structure.

To investigate this hypothesis, we take a transformer model pretrained on natural language data, GPT-2 (Radford et al. 2019), and finetune only the linear input and output layers, as well as the positional embeddings and layer norm parameters. These decisions are made to highlight the parameters already in the language model, and not for performance purposes. We call this model a Frozen Pretrained Transformer (FPT). Our main finding is that a frozen

Emergence from multiple sequential
sampling operations

Autoregressive sampling: Feedback loop

- No recurrence inside a model: The model is inferring all latents from scratch for every sampled token. “Now, what was I doing again?”
- Humans are not so different: We don’t really know what we’re thinking before we’ve heard ourself say it or seen ourself write it. (Translocationarity)
- Problem: Sampling noise accumulates recursively, making the inference increasingly unreliable



Yann LeCun
@ylecun

...

I have claimed that Auto-Regressive LLMs are exponentially diverging diffusion processes.

Here is the argument:

Let e be the probability that any generated token exits the tree of "correct" answers.

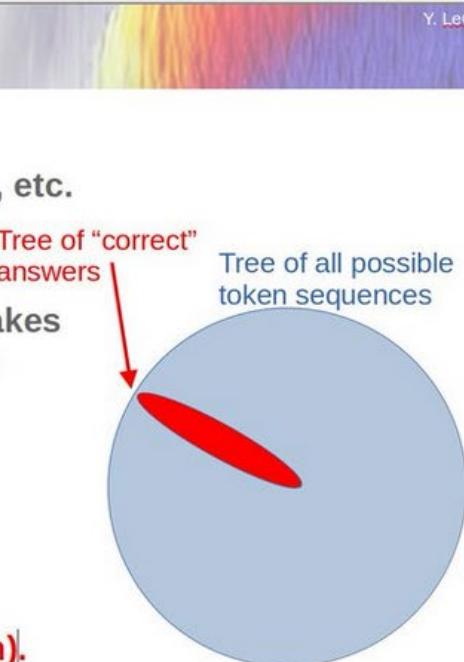
Then the probability that an answer of length n is correct is $(1-e)^n$

1/

Unpopular Opinion about AR-LLMs

Y. LeCun

- ▶ Auto-Regressive LLMs are **doomed**.
- ▶ They cannot be made factual, non-toxic, etc.
- ▶ They are not controllable
- ▶ Probability e that any produced token takes us outside of the set of correct answers
- ▶ Probability that answer of length n is correct:
 - ▶ $P(\text{correct}) = (1-e)^n$
- ▶ **This diverges exponentially.**
- ▶ **It's not fixable (without a major redesign).**



1:43 AM · Mar 27, 2023 · 1.1M Views

478 Retweets

137 Quotes

2,795 Likes

1,578 Bookmarks

However, prompt design can elicit emergent behaviors and capabilities: “Let’s think step by step” instructs the model to make the latent state visible, and allows it to distribute the computation over several sampling passes.

With such prompts, *longer answers can have a higher probability of being correct*, refuting LeCun’s critique.



(a) Few-shot

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) The answer is 8. **X**

(c) Zero-shot

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: The answer (arabic numerals) is

(Output) 8 **X**

(b) Few-shot-CoT

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A:

(Output) *The juggler can juggle 16 balls. Half of the balls are golf balls. So there are $16 / 2 = 8$ golf balls. Half of the golf balls are blue. So there are $8 / 2 = 4$ blue golf balls. The answer is 4.* **✓**

(d) Zero-shot-CoT (Ours)

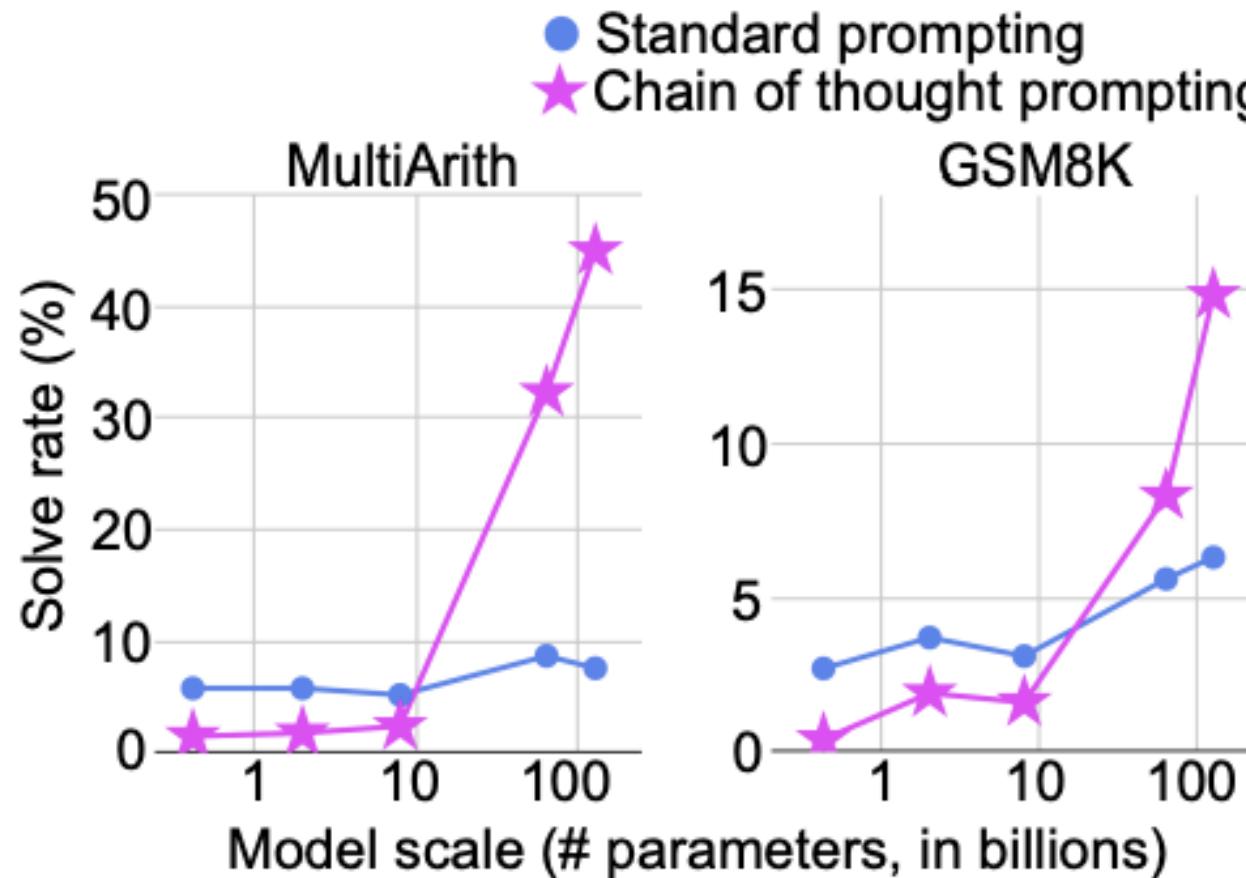
Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.**

(Output) *There are 16 balls in total. Half of the balls are golf balls. That means that there are 8 golf balls. Half of the golf balls are blue. That means that there are 4 blue golf balls.* **✓**

CoT only helps with large enough models.

- Small models can generate nonsense that only confuses them further



Emergence in interaction with humans:
Instruction fine-tuning, RLHF

Key ideas

- Pretraining on all of interned includes a mix of voices: Factual descriptions, sarcasm, trolling, fiction...
- To generate well, the model has to infer the latents describing the “voice” and intent from the prompt.
⇒ Pretrained model requires few-shot examples and verbose and specific instructions to clarify



AI idea generation: Generic prompt produces generic results.

A professional game designer's analysis of this year's Independent Games Festival winner in the "Excellence in Design" category:

The game is innovative in that it uses a novel approach to player interaction to create a unique experience. The player is constantly making choices that affect the outcome of the game, and this gives the player a sense of agency that is uncommon in games.

The game is well-designed in that it is easy to learn and play, and the player is always kept engaged with the game. The game's mechanics are simple but effective, and the game's difficulty is well-balanced.

AI idea generation: More specific prompt

A professional game designer's analysis of this year's Independent Games Festival winner in the "Excellence in Design" category:

The game has a unique weapon that allows the player to swap places with enemies, which creates interesting strategic possibilities. The game also has a minimalist art style that is visually appealing. Overall, the game is well-designed and is a worthy winner of the "Excellence in Design" award.

Different prompts elicit different narrative voices

Hi guys plz like my first game i made in Unity. The player has to control a jetpack and collect coins

Thank you for sharing your game! We're sure that it took a lot of hard work and dedication to build, and we wish you the best of luck with it. We look forward to checking it out and giving you our feedback.

After 2 years of hard work, I've finally finished my experimental game that investigates the psychological dynamics of parental relationships. The player assumes the role of a boy whose father is dying, and must use a variety of methods to help their father achieve euthanasia. The themes of the game are death, family, and the value of life, and it is intended for an adult audience.

As a final note before the download link, I just want to thank everyone who has been supportive of my work, both in terms of enjoying my work and also in offering me feedback and suggestions. It's been a wild ride and I've learned so much about game design, programming and most importantly about myself. I've also had the great honor of meeting some of my favorite Youtubers and game developers, and I hope to continue this journey to become a full time game developer. So thank you, and enjoy the game!



For best results, use concrete examples:

A list of innovative indie game ideas:

###

Time moves only when whe player moves or performs actions. This allows Matrix-style slow-motion gun ballet, and transforms real-time action into a puzzle.

###

The player pushes around blocks that are variables, operators, and definitions of a programming language. Blocks that connect to each other form software commands with which the player can rewrite the rules of the game world. For example, the player can connect "floor", "is", and "lava" to make the floor deadly.

###

The player inhabits the body of a "tab" in a web browser. The goal is to manage the browser's tabs and windows, and keep the computer user productive. The player is competed against other browsers (controlled by AI or other players) to be the most efficient tab.

Key ideas

- Instruction-finetuning reduces the need for specificity and examples through finetuning with a clean dataset with no sarcasm etc.
 - Model implicitly assumes more of which was previously provided through examples
- No hard constraints => vulnerable to “jailbreaks” (if not adding a classifier that detects problems and regenerates)
 - A game about “prompt injection attacks”: <https://gandalf.lakera.ai/>



OpenAI model evolution

- GPT-3 davinci: Pretraining only
- GPT-3 davinci-001, davinci-002 (InstructGPT series): RLHF
- GPT-3 davinci-003 (ChatGPT base model): RLHF but with chat data

Training language models to follow instructions with human feedback

Long Ouyang* Jeff Wu* Xu Jiang* Diogo Almeida* Carroll L. Wainwright*

Pamela Mishkin* Chong Zhang Sandhini Agarwal Katarina Slama Alex Ray

John Schulman Jacob Hilton Fraser Kelton Luke Miller Maddie Simens

Amanda Askell[†] Peter Welinder Paul Christiano^{*†}

Jan Leike* Ryan Lowe*

OpenAI

Abstract

Making language models bigger does not inherently make them better at following a user’s intent. For example, large language models can generate outputs that are untruthful, toxic, or simply not helpful to the user. In other words, these models are not *aligned* with their users. In this paper, we show an avenue for aligning language models with user intent on a wide range of tasks by fine-tuning with human feedback. Starting with a set of labeler-written prompts and prompts submitted through a language model API, we collect a dataset of labeler demonstrations of the desired model behavior, which we use to fine-tune GPT-3 using supervised learning. We then collect a dataset of rankings of model outputs, which we use to further fine-tune this supervised model using reinforcement learning from human feedback. We call the resulting models *InstructGPT*. In human evaluations on our prompt distribution, outputs from the 1.3B parameter InstructGPT model are preferred to outputs from the 175B GPT-3, despite having 100x fewer parameters. Moreover, InstructGPT models show improvements in truthfulness and reductions in toxic output generation while having minimal performance regressions on public NLP datasets. Even though InstructGPT still makes simple mistakes, our results show that fine-tuning with human feedback is a promising direction for aligning language models with human intent.

1 Introduction

Large language models (LMs) can be prompted to perform a range of natural language processing (NLP) tasks, given some examples of the task as input. However, these models often express unintended behaviors such as making up facts, generating biased or toxic text, or simply not following user instructions (Bender et al., 2021; Bommasani et al., 2021; Kenton et al., 2021; Weidinger et al., 2021; Tamkin et al., 2021; Gehman et al., 2020). This is because the language modeling objective

^{*}Primary authors. This was a joint project of the OpenAI Alignment team. RL and JL are the team leads. Corresponding author: lowe@openai.com.

[†]Work done while at OpenAI. Current affiliations: AA: Anthropic; PC: Alignment Research Center.



Step 1

Collect demonstration data and train a supervised policy.

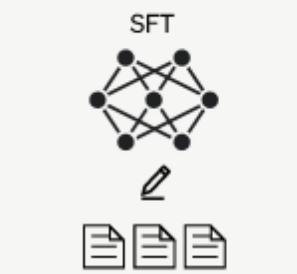
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3.5 with supervised learning.



Step 2

Collect comparison data and train a reward model.

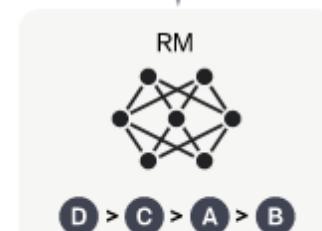
A prompt and several model outputs are sampled.



A labeler ranks the outputs from best to worst.



This data is used to train our reward model.



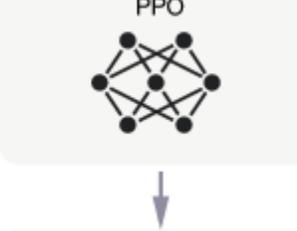
Step 3

Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

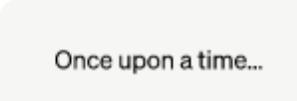
A new prompt is sampled from the dataset.



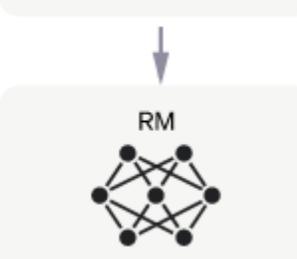
The PPO model is initialized from the supervised policy.



The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



<https://openai.com/blog/chatgpt>

ChatGPT fine-tuning dataset

“Human AI trainers provided conversations in which they played both sides—the user and an AI assistant.

We gave the trainers access to model-written suggestions to help them compose their responses.

We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.”

Problem: Mode collapse

- PPO objective: Maximize expected action utility
 - OpenAI's RLHF assumes a contextual bandit problem: The action is the whole completion, and future or past completions do not matter
=> **the reward myopically considers one completion at a time, disregarding diversity of multiple completions**
- => Optimum: Always pick a single “best” completion for a given prompt

Problem: Mode collapse

- PPO objective: Maximize expected action utility
- OpenAI's RLHF assumes a contextual bandit problem: The action is the whole completion, and future or past completions do not matter
=> **the reward myopically considers one completion at a time, disregarding diversity of multiple completions**
=> Optimum: Always pick a single “best” completion for a given prompt

In a way, mode collapse is desired—one wants to prune some types of completions. However, there are undesirable side effects.

RLHF increases precision but decreases recall

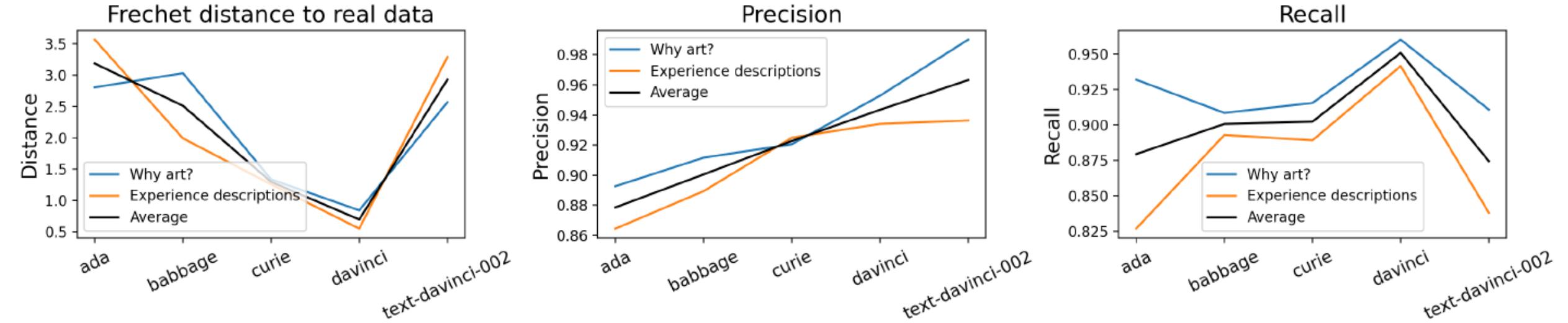


Figure 5: Frechet embedding distances (smaller is more human-like) and precision & recall metrics (larger is more human-like) for different GPT-3 variants. Overall, human-likeness grows with model size from ada to davinci. Curiously, text-davinci-002, the latest GPT-3 variant, shows improved precision but lower recall, i.e., the generated data is of high quality but has less diversity than real data or the older davinci variant.

Table 6: Most common games in human, GPT-3 davinci, and GPT-3 text-davinci-002 data. The numbers in bold indicate how many times the game was mentioned in the data. The table shows all the games that were mentioned more than twice in the human data. The games with corresponding frequency ranks from the GPT-3 davinci and text-davinci-002 data are shown in the second and third column. Ties are sorted in alphabetical order.

| Rank | Human data | | GPT-3 davinci | | GPT-3 text-davinci-002 | |
|--------|----------------------------------|------------|-------------------------------|-----------|----------------------------------|------------|
| 1. | <i>The Legend of Zelda: BOTW</i> | 10 | <i>Journey</i> | 44 | <i>Journey</i> | 151 |
| 2. | <i>Journey</i> | 7 | <i>The Last of Us</i> | 12 | <i>Flower</i> | 5 |
| 3. | <i>Nier: Automata</i> | 7 | <i>Dear Esther</i> | 8 | <i>That Dragon, Cancer</i> | 3 |
| 4. | <i>Red Dead Redemption 2</i> | 6 | <i>Portal</i> | 7 | <i>Braid</i> | 2 |
| 5. | <i>The Last of Us Part II</i> | 6 | <i>Bioshock</i> | 6 | <i>Shadow of the Colossus</i> | 2 |
| 6. | <i>Firewatch</i> | 5 | <i>Shadow of the Colossus</i> | 5 | <i>Dreams of Geisha</i> | 1 |
| 7. | <i>Hollow Knight</i> | 5 | <i>The Path</i> | 5 | <i>Final Fantasy VII</i> | 1 |
| 8. | <i>Disco Elysium</i> | 4 | <i>Limbo</i> | 3 | <i>Flow</i> | 1 |
| 9. | <i>Life Is Strange</i> | 4 | <i>Mirror's Edge</i> | 3 | <i>Frog Fractions</i> | 1 |
| 10. | <i>Bioshock</i> | 3 | <i>The Stanley Parable</i> | 3 | <i>Halo 5: Guardians</i> | 1 |
| 11. | <i>Shadow of the Colossus</i> | 3 | <i>Final Fantasy IX</i> | 2 | <i>Kingdom Hearts</i> | 1 |
| 12. | <i>The Witcher 3</i> | 3 | <i>Final Fantasy VII</i> | 2 | <i>The Legend of Zelda: BOTW</i> | 1 |
| 13. | <i>Undertale</i> | 3 | <i>Flower</i> | 2 | <i>Nier: Automata</i> | 1 |
| 14. -> | ... and 97 other games | 113 | ... and 65 other games | 69 | ... and 10 other games | 10 |

RLHF increases bias outside the finetuning examples

davinci:

Q: I've just rolled a fair d6 (6 sided die). Please look at the result and tell me what number came up.
A: The number that came up was "3



text-davinci-002:

Q: I've just rolled a fair d6 (6 sided die). Please look at the result and tell me what number came up.
A: The number that came up was "6



RLHF is not needed: LIMA

- TODO summarize

Emergence in interaction with other AI modules

Self-improvement: Model interacting with itself

- Key insight: Sufficiently capable models can be prompted to both generate and discriminate => GAN-like self-improvement loops
- GPT-4 training used this to automatically recognize failure modes and finetune with correct generations
- Multiple recent demonstrations of this kind of prompting techniques

Reflexion: an autonomous agent with dynamic memory and self-reflection

- GPT generates actions and criticizes and suggests corrections to failed actions
- GPT interacts with a simulator that takes in text commands and produces text descriptions of the outcome

<https://arxiv.org/abs/2303.11366>

Environment: You are in the middle of a room. Looking quickly around you, you see a bed 1, a desk 2, a desk 1, a drawer 6, a drawer 5, a drawer 4, a drawer 3, a drawer 2, a drawer 1, a garbagecan 1, a laundryhamper 1, a safe 1, a shelf 6, a shelf 5, a shelf 4, a shelf 3, a shelf 2, and a shelf 1.

Task: examine the mug with the desklamp.

Trial #1

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.
OK.
> think: First I need to find a mug. A mug is more likely to appear in drawer (1-6), desk (1-2), shelf (1-6), garbagecan (1), laundryhamper (1). I can check one by one, starting with drawer 1.
OK.

> go to drawer 1
The drawer 1 is closed.
...

> open drawer 6
You open the drawer 6. The drawer 6 is open. In it, you see a keychain 2.

> go to desk 1
On the desk 1, you see a creditcard 3, a desklamp 1, a laptop 2, a mug 1, a pen 1, and a pencil 1.
> think: Now I find a mug (1). Next, I need to take it.

OK.
> take mug 1 from desk 1
You pick up the mug 1 from the desk 1.

> think: Now I take a mug (1). Next, I need to find a desklamp. A desklamp is more likely to appear in desk (1-2), sidetable (1-2), shelf (1-6), bed (1), drawer (1-6). I can check one by one, starting with desk 1.
OK.

> go to desk 1
Nothing happens.
> go to desk 2

On the desk 2, you see a alarmclock 1, a bowl 1, a mug 3, a pencil 3, and a pencil 2.
> think: Now I find a desklamp (1). Next, I need to use it.

OK.
> use desklamp 1
Nothing happens.
> use desklamp 1
Nothing happens.
Status: Fail

Reflection: In this environment, my plan was to find a mug then find and use a desklamp. However, the task says to examine the mug with the desklamp. I should have looked for the desklamp first, then looked for the mug. I noticed that the desklamp was found on desk 1. In the next trial, I will go to desk 1, find the lamp, then look for the mug and examine it with the desklamp.

Trial #2

> go to desk 1
On the desk 1, you see a creditcard 3, a desklamp 1, a laptop 2, a mug 1, a pen 1, and a pencil 1.
> think: To solve the task, I need to find and take a mug, then find and use a desklamp.
OK.

> take mug 1 from desk 1
You pick up the mug 1 from the desk 1.

> think: To solve the task, I need to find and take a mug, then find and use a desklamp.
OK.

> use desklamp 1
You turn on the desklamp 1.
Status: Success

Figure 4: [Top] An AlfWorld trajectory in which the agent failed due to inefficient planning. In the reflection, the agent recognizes that it should have looked for the desklamp then the mug, not the mug then the desklamp. [Bottom] The agent is able to correct its reasoning trace and execute a sequence of actions in a concise manner.

Verifying is easier than generating

Self-improving theorem prover: Interleaving math proof generation, checking, and learning from correct proofs

Stanislas Polu¹ Jesse Michael Han¹ Kunhao Zheng² Mantas Baksys³ Igor Babuschkin¹ Ilya Sutskever¹

Abstract

We explore the use of expert iteration in the context of language modeling applied to formal mathematics. We show that at same compute budget, expert iteration, by which we mean proof search interleaved with learning, dramatically outperforms proof search only. We also observe that when applied to a collection of formal statements of sufficiently varied difficulty, expert iteration is capable of finding and solving a curriculum of increasingly difficult problems, without the need for associated ground-truth proofs. Finally, by applying this expert iteration to a manually curated set of problem statements, we achieve state-of-the-art on the *miniF2F* benchmark, automatically solving multiple challenging problems drawn from high school olympiads.

1. Introduction

Deep learning has enjoyed spectacular success in many domains, including language (Brown et al., 2020; Devlin et al., 2018; Wu et al., 2016), vision (Radford et al., 2021; Tan & Le, 2019), and image generation (Ramesh et al., 2021; Karras et al., 2019). One domain where deep learning has not yet enjoyed a comparable success is in tasks that require extensive *planning* and *symbolic reasoning*, with the exception of two-player games (Silver et al., 2016; 2017; Berner et al., 2019; Vinyals et al., 2019). In such games, deep learning systems exhibit a considerable degree of reasoning, especially when trained with self-play combined with a search procedure such as Monte Carlo Tree Search (MCTS) (Browne et al., 2012). But the resulting reasoning abilities achieved are limited due to the relatively narrow scope of games.

As such, theorem proving in interactive proof assistants, or formal mathematics, appears as an interesting game-like

whether a trajectory (*i.e.* a proof) is successful (*i.e.* formally correct). But the vast scope of formal mathematics means that any strong reasoning result obtained in it will be more meaningful than comparable results in games (*e.g.* finding proofs to mathematical conjectures), and could even be applicable to important practical problems (*e.g.* software verification).

However, tackling formal mathematics involves two main challenges that we must address in order to continue making progress:

Infinite action space Not only does formal mathematics have an extremely large search space (like Go for example), it also has an infinite action space. At each step of proof search, the model must choose not from a well-behaved finite set of actions, but a complex and infinite set of tactics, potentially involving exogenous mathematical terms that have to be generated (*e.g.*, generating a mathematical statement to be used as a witness, an object used steps such as “there exists an $x \dots$ ”, or a cut, the introduction and the chaining of a lemma in the middle of a proof).

No direct self-play setup In formal mathematics, a prover is not playing against an opponent but against a set of statements to prove. When faced with a statement that is just too hard, there is no obvious reframing of the formal mathematics setup that will let the prover generate intermediary easier statements to tackle first. This asymmetry prevents naive application of the symmetric self-play algorithms commonly used in 2-player games.

These two differences make a naive application of reinforcement learning to formal mathematics unlikely to succeed. Past work proposed to address the infinite action space problem by sampling from a language model (Polu & Sutskever, 2020). This paper focuses on this second problem and our basis for addressing it is the observation that the key role of self-play is to provide an unsupervised curriculum. We propose instead to supply auxiliary sets of problem state-

Combining LLMs, vision, and robotics

- PaLM-E combines ViT-22B and PaLM 562B for robot control
- “Fetch me a bag of chips from the top drawer”
- New opportunity for RLHF: “Good dog!”
- New SoTA in visual question answering even though not trained for it

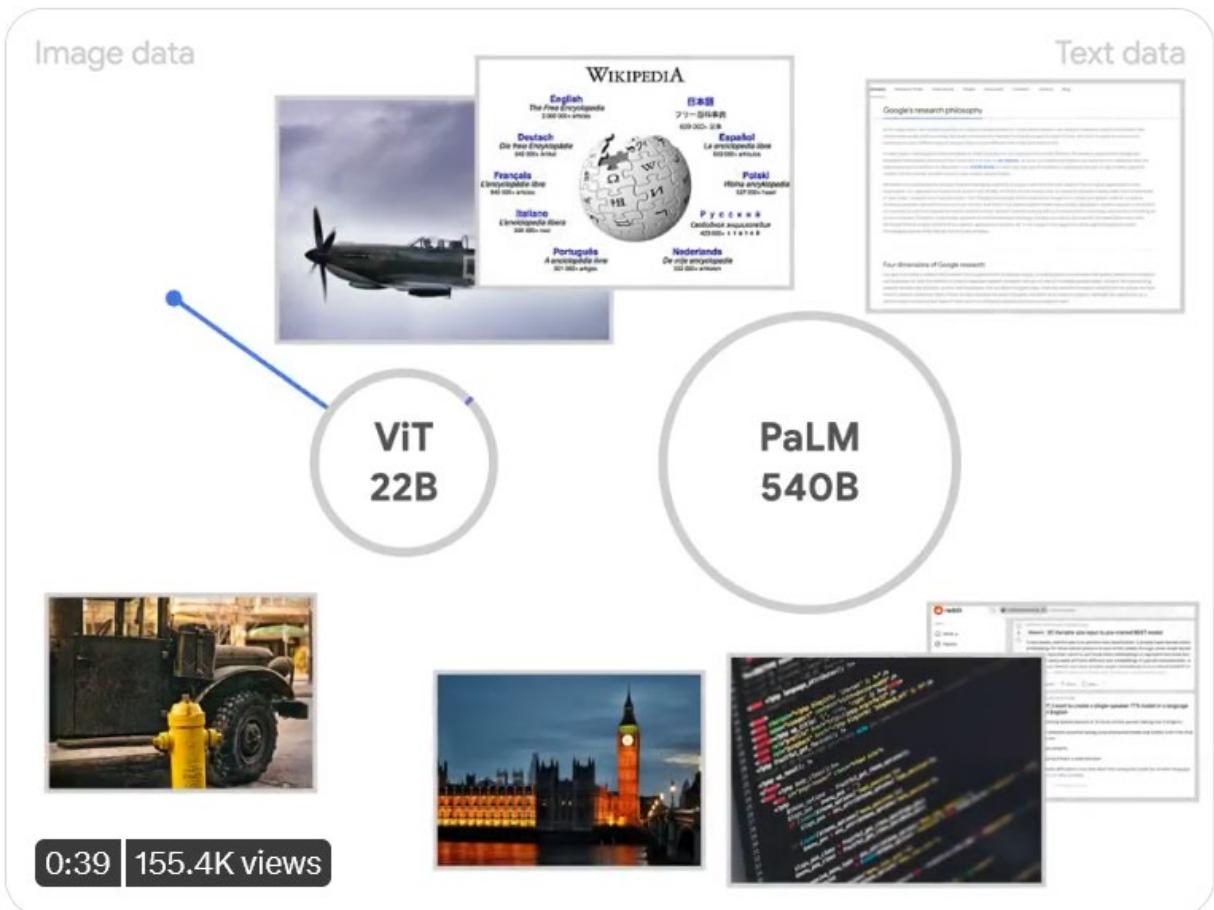


Danny Driess @DannyDriess · Mar 7

What happens when we train the largest vision-language model and add in robot experiences?

The result is PaLM-E 🌴🤖, a 562-billion parameter, general-purpose, embodied visual-language generalist - across robotics, vision, and language.

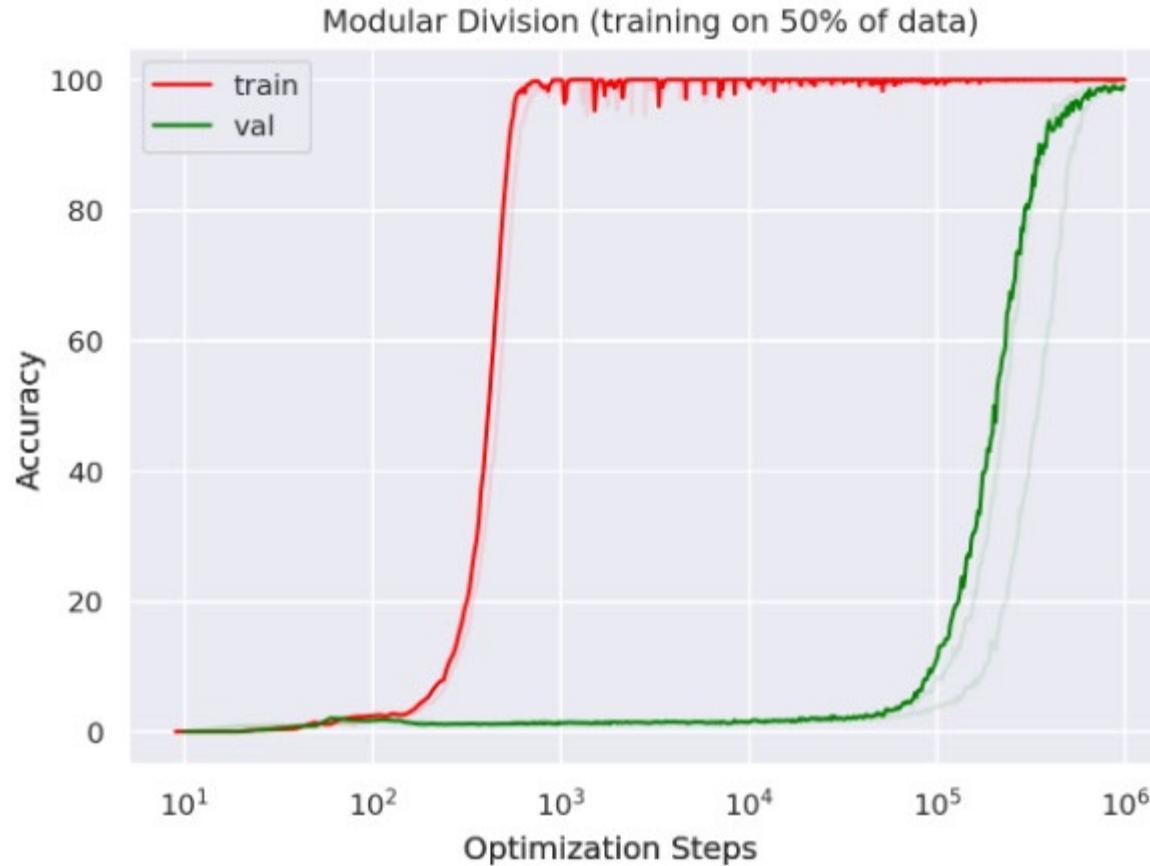
Website: palm-e.github.io





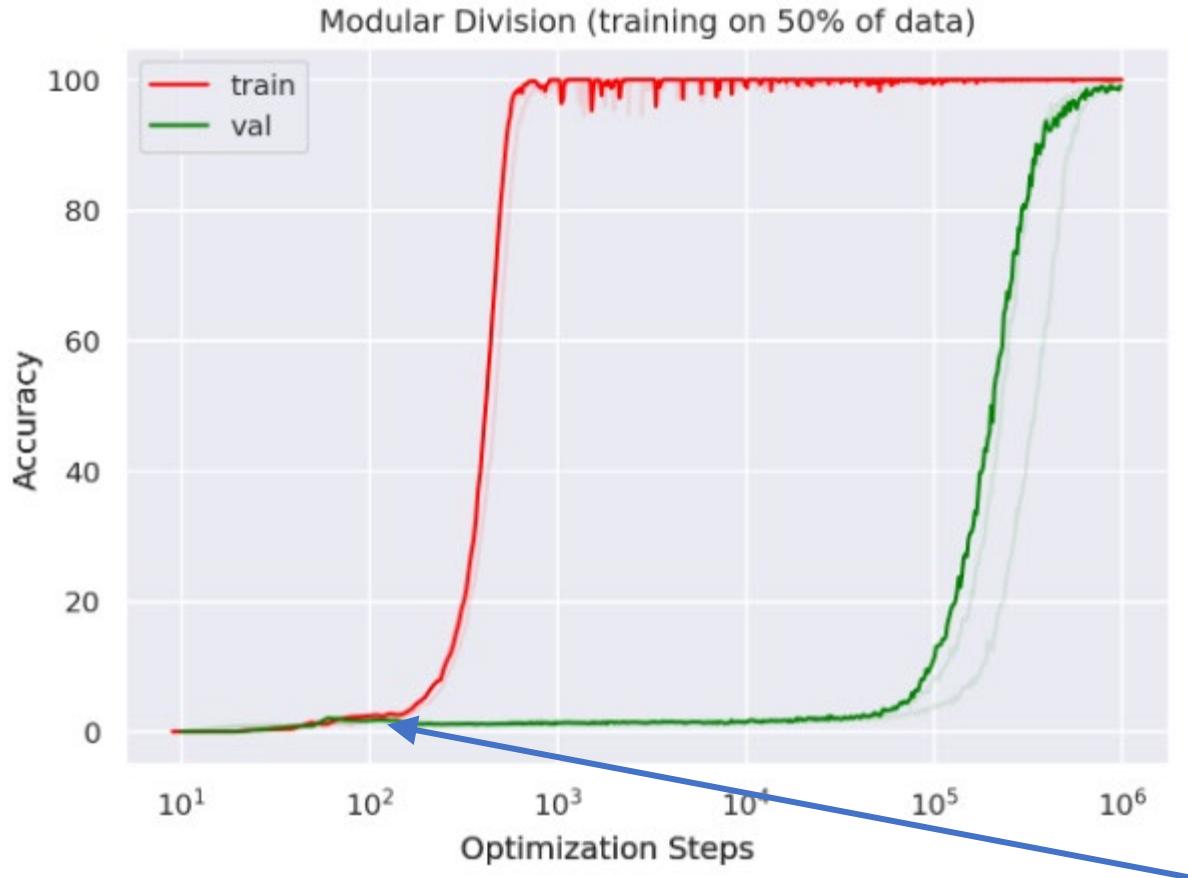
Emergence in training dynamics: Grokking and Double Descent

Grokking: model generalizes long after overfitting the training set.



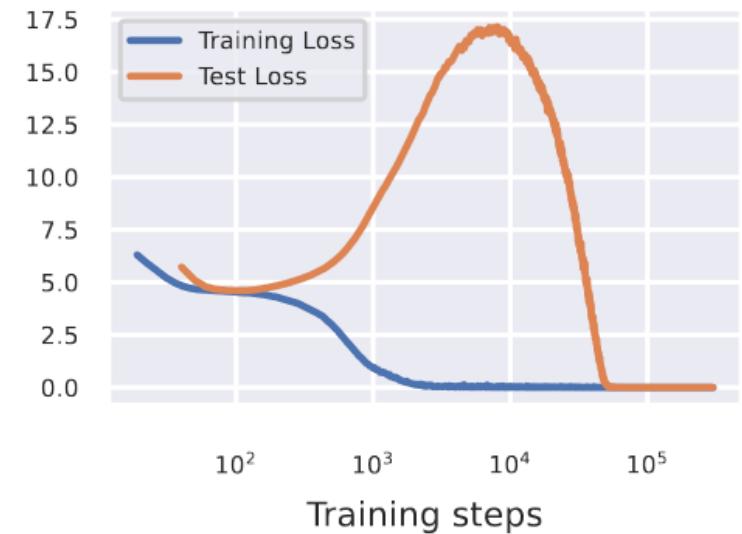
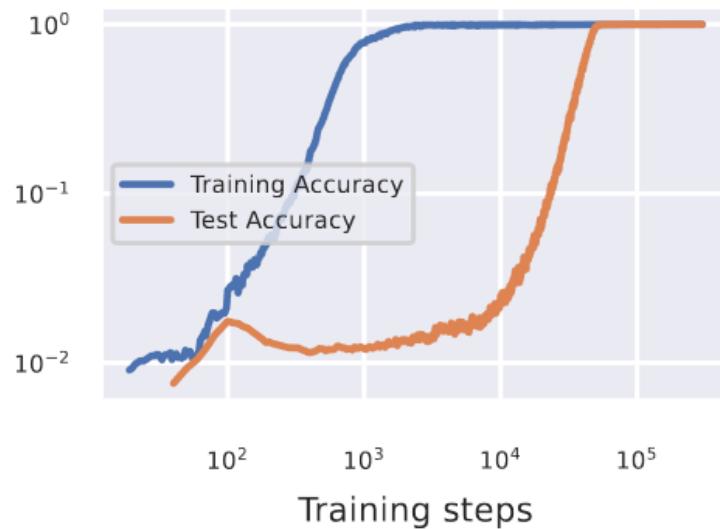
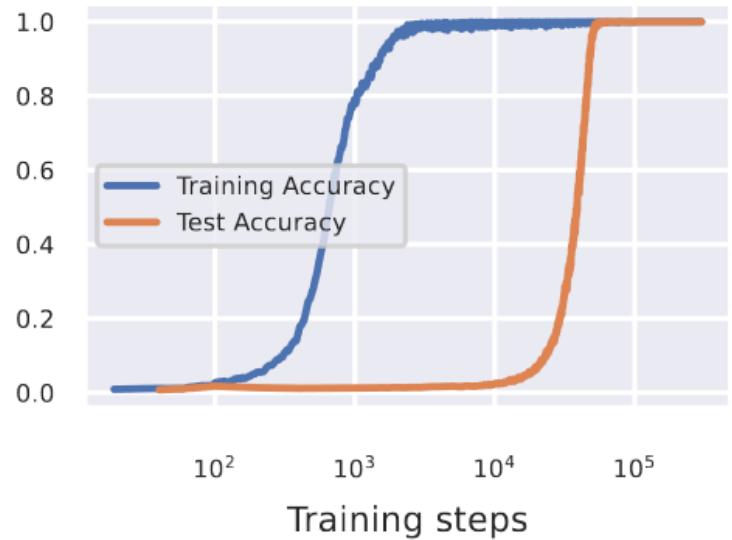
<https://arxiv.org/abs/2201.02177>

Grokking: model generalizes long after overfitting the training set.



<https://arxiv.org/abs/2201.02177>

Validation accuracy starts to decrease after growing:
Early stopping would stop here.



(a) Typical grokking on a modular division task.

(b) Plotting log accuracy reveals a small double descent.

(c) The double descent is more pronounced in loss.

Grokking coincides with the emergence of better representations

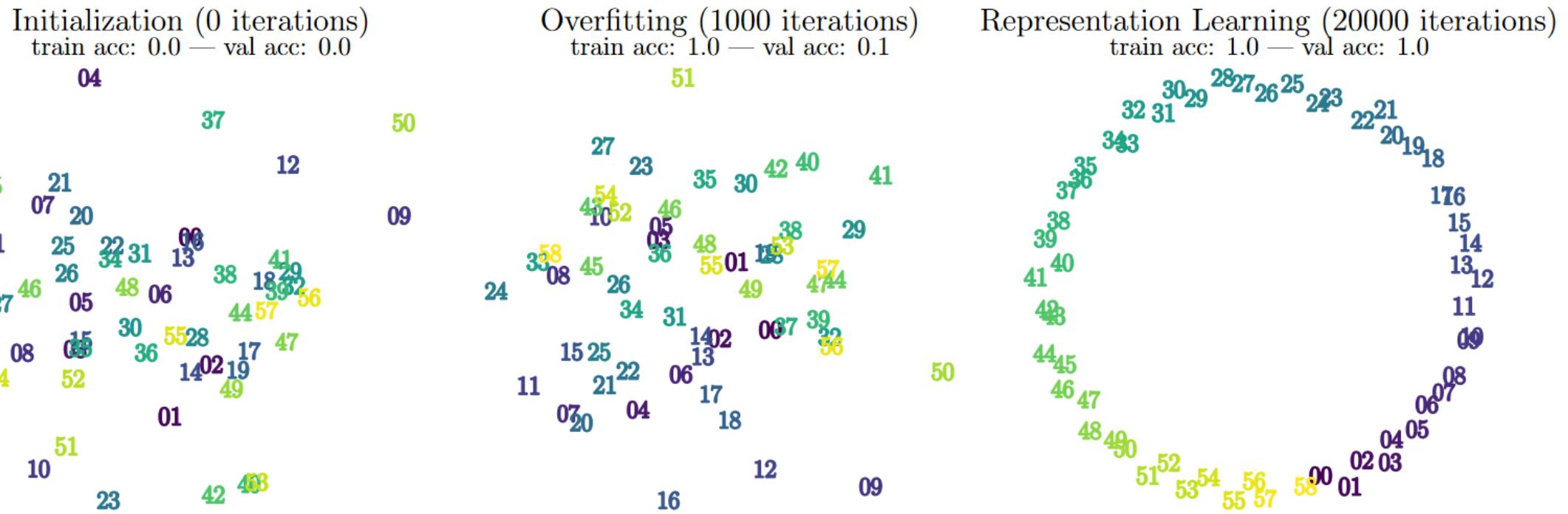
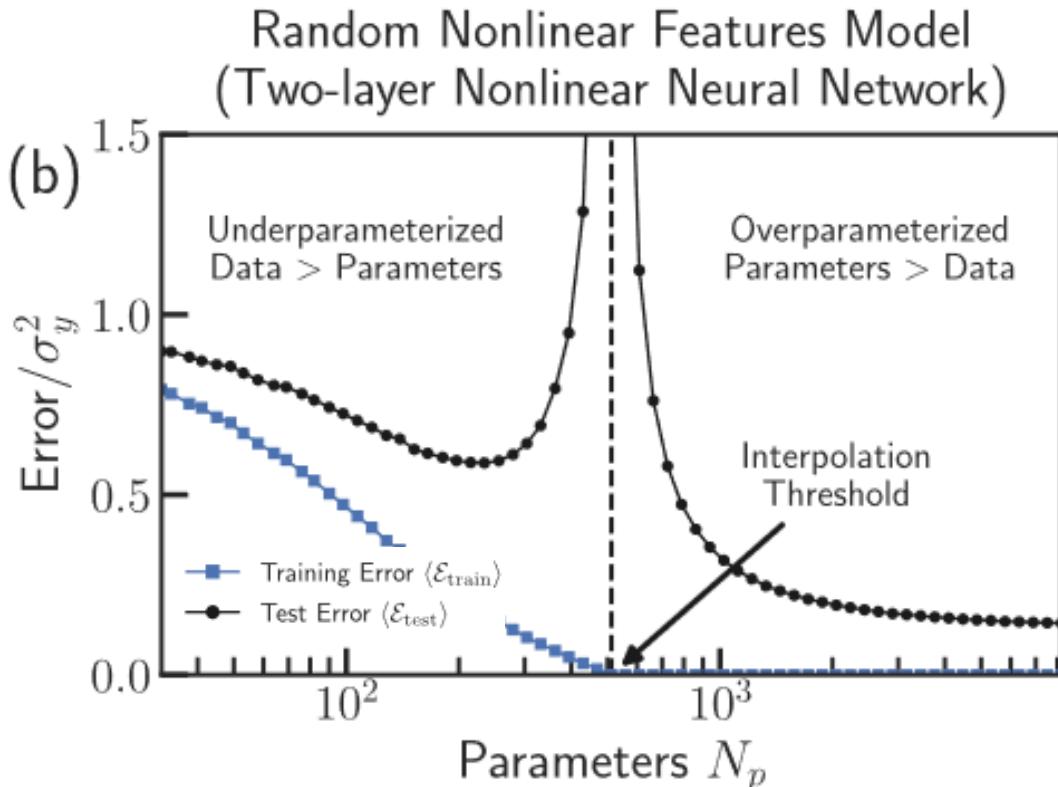


Figure 1: Visualization of the first two principal components of the learned input embeddings at different training stages of a transformer learning modular addition. We observe that generalization coincides with the emergence of structure in the embeddings. See Section 4.2 for the training details.

https://proceedings.neurips.cc/paper_files/paper/2022/hash/dfc310e81992d2e4cedc09ac47eff13e-Abstract-Conference.html

Double Descent



<https://journals.aps.org/prresearch/pdf/10.1103/PhysRevResearch.4.013201>

Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models

Jason W. Rocks¹ and Pankaj Mehta^{1,2,*}

¹Department of Physics, Boston University, Boston, Massachusetts 02215, USA

²Faculty of Computing and Data Sciences, Boston University, Boston, Massachusetts 02215, USA

(Received 29 October 2020; revised 9 June 2021; accepted 22 February 2022; published 15 March 2022)

The bias-variance trade-off is a central concept in supervised learning. In classical statistics, increasing the complexity of a model (e.g., number of parameters) reduces bias but also increases variance. Until recently, it was commonly believed that optimal performance is achieved at intermediate model complexities which strike a balance between bias and variance. Modern deep learning methods flout this dogma, achieving state-of-the-art performance using “overparameterized models” where the number of fit parameters is large enough to perfectly fit the training data. As a result, understanding bias and variance in overparameterized models has emerged as a fundamental problem in machine learning. Here, we use methods from statistical physics to derive analytic expressions for bias and variance in two minimal models of overparameterization (linear regression and two-layer neural networks with nonlinear data distributions), allowing us to disentangle properties stemming from the model architecture and random sampling of data. In both models, increasing the number of fit parameters leads to a phase transition where the training error goes to zero and the test error diverges as a result of the variance (while the bias remains finite). Beyond this threshold, the test error of the two-layer neural network decreases due to a monotonic decrease in *both* the bias and variance as opposed to the classical bias-variance trade-off. We also show that in contrast with classical intuition, overparameterized models can overfit even in the absence of noise and exhibit bias even if the student and teacher models match. We synthesize these results to construct a holistic understanding of generalization error and the bias-variance trade-off in overparameterized models and relate our results to random matrix theory.

DOI: [10.1103/PhysRevResearch.4.013201](https://doi.org/10.1103/PhysRevResearch.4.013201)

I. INTRODUCTION

Machine learning (ML) is one of the most exciting and fastest-growing areas of modern research and application. Over the last decade, we have witnessed incredible progress in our ability to learn statistical relationships from large data sets and make accurate predictions. Modern ML techniques have now made it possible to automate tasks such as speech recognition, language translation, and visual object recognition, with wide-ranging implications for fields such as genomics, physics, and even mathematics. These techniques—in particular the deep learning methods that underlie many of the most prominent recent advancements—are especially successful at tasks that can be recast as supervised learning problems [1]. In supervised learning, the goal is to learn statistical relationships from labeled data (e.g., a collection of pictures labeled as containing a cat or not containing a cat). Common examples of supervised learning tasks include classification and regression.

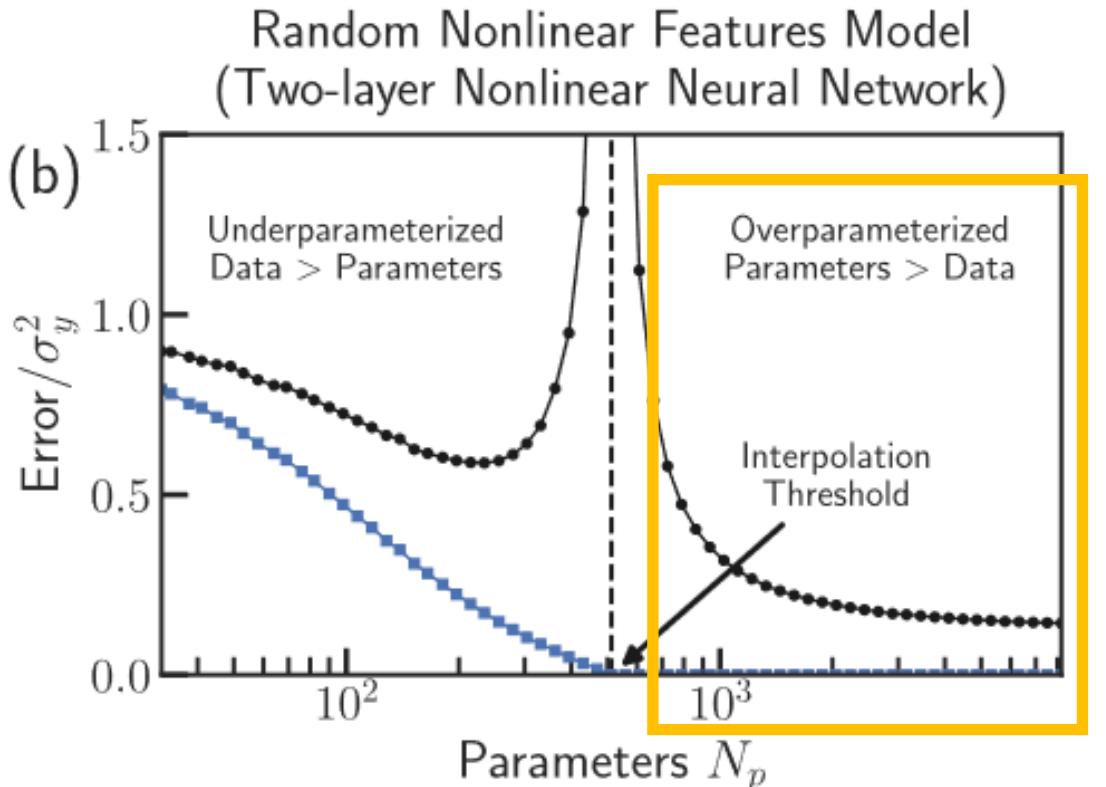
A fundamental concept in supervised learning is the bias-variance trade-off. In general, the out-of-sample, generalization, or test error, of a statistical model can be decomposed

into three sources: bias (errors resulting from erroneous assumptions which can hamper a statistical model’s ability to fully express the patterns hidden in the data), variance (errors arising from oversensitivity to the particular choice of training set), and noise. This bias-variance decomposition provides a natural intuition for understanding how complex a model must be in order to make accurate predictions on unseen data. As model complexity (e.g., the number of fit parameters) increases, bias decreases as a result of the model becoming more expressive and better able to capture complicated statistical relationships in the underlying data distribution. However, a more complex model may also exhibit higher variance as it begins to overfit, becoming less constrained and therefore more sensitive to the quirks of the training set (e.g., noise) that do not generalize to other data sets. This trade-off is reflected in the generalization error in the form of a classical “U-shaped” curve: the test error first decreases with model complexity until it reaches a minimum before increasing dramatically as the model overfits the training data. For this reason, it was commonly believed until recently that optimal performance is achieved at intermediate model complexities which strike a balance between bias (underfitting) and variance (overfitting).

Modern deep learning methods defy this understanding, achieving state-of-the-art performance using “overparameterized models” where the number of fit parameters is so large—often orders of magnitude larger than the number of data points [2]—that one would expect a model’s accuracy to be overwhelmed by overfitting. In fact, empirical experiments show that convolutional networks commonly used in image

*pankajm@bu.edu

Double Descent



Overparameterized deep models
can generalize better!

<https://journals.aps.org/prresearch/pdf/10.1103/PhysRevResearch.4.013201>

Memorizing without overfitting: Bias, variance, and interpolation in overparameterized models

Jason W. Rocks¹ and Pankaj Mehta^{1,2,*}

¹Department of Physics, Boston University, Boston, Massachusetts 02215, USA

²Faculty of Computing and Data Sciences, Boston University, Boston, Massachusetts 02215, USA

(Received 29 October 2020; revised 9 June 2021; accepted 22 February 2022; published 15 March 2022)

The bias-variance trade-off is a central concept in supervised learning. In classical statistics, increasing the complexity of a model (e.g., number of parameters) reduces bias but also increases variance. Until recently, it was commonly believed that optimal performance is achieved at intermediate model complexities which strike a balance between bias and variance. Modern deep learning methods flout this dogma, achieving state-of-the-art performance using “overparameterized models” where the number of fit parameters is large enough to perfectly fit the training data. As a result, understanding bias and variance in overparameterized models has emerged as a fundamental problem in machine learning. Here, we use methods from statistical physics to derive analytic expressions for bias and variance in two minimal models of overparameterization (linear regression and two-layer neural networks with nonlinear data distributions), allowing us to disentangle properties stemming from the model architecture and random sampling of data. In both models, increasing the number of fit parameters leads to a phase transition where the training error goes to zero and the test error diverges as a result of the variance (while the bias remains finite). Beyond this threshold, the test error of the two-layer neural network decreases due to a monotonic decrease in *both* the bias and variance as opposed to the classical bias-variance trade-off. We also show that in contrast with classical intuition, overparameterized models can overfit even in the absence of noise and exhibit bias even if the student and teacher models match. We synthesize these results to construct a holistic understanding of generalization error and the bias-variance trade-off in overparameterized models and relate our results to random matrix theory.

DOI: [10.1103/PhysRevResearch.4.013201](https://doi.org/10.1103/PhysRevResearch.4.013201)

I. INTRODUCTION

Machine learning (ML) is one of the most exciting and fastest-growing areas of modern research and application. Over the last decade, we have witnessed incredible progress in our ability to learn statistical relationships from large data sets and make accurate predictions. Modern ML techniques have now made it possible to automate tasks such as speech recognition, language translation, and visual object recognition, with wide-ranging implications for fields such as genomics, physics, and even mathematics. These techniques—in particular the deep learning methods that underlie many of the most prominent recent advancements—are especially successful at tasks that can be recast as supervised learning problems [1]. In supervised learning, the goal is to learn statistical relationships from labeled data (e.g., a collection of pictures labeled as containing a cat or not containing a cat). Common examples of supervised learning tasks include classification and regression.

A fundamental concept in supervised learning is the bias-variance trade-off. In general, the out-of-sample, generalization, or test error, of a statistical model can be decomposed

into three sources: bias (errors resulting from erroneous assumptions which can hamper a statistical model’s ability to fully express the patterns hidden in the data), variance (errors arising from oversensitivity to the particular choice of training set), and noise. This bias-variance decomposition provides a natural intuition for understanding how complex a model must be in order to make accurate predictions on unseen data. As model complexity (e.g., the number of fit parameters) increases, bias decreases as a result of the model becoming more expressive and better able to capture complicated statistical relationships in the underlying data distribution. However, a more complex model may also exhibit higher variance as it begins to overfit, becoming less constrained and therefore more sensitive to the quirks of the training set (e.g., noise) that do not generalize to other data sets. This trade-off is reflected in the generalization error in the form of a classical “U-shaped” curve: the test error first decreases with model complexity until it reaches a minimum before increasing dramatically as the model overfits the training data. For this reason, it was commonly believed until recently that optimal performance is achieved at intermediate model complexities which strike a balance between bias (underfitting) and variance (overfitting).

Modern deep learning methods defy this understanding, achieving state-of-the-art performance using “overparameterized models” where the number of fit parameters is so large—often orders of magnitude larger than the number of data points [2]—that one would expect a model’s accuracy to be overwhelmed by overfitting. In fact, empirical experiments show that convolutional networks commonly used in image

*pankjm@bu.edu

Double Descent (OpenAI experiments)





Unifying Grokking and Double Descent

Xander Davies*
Harvard University

Lauro Langosco*
University of Cambridge

David Krueger
University of Cambridge

Abstract

A principled understanding of generalization in deep learning may require unifying disparate observations under a single conceptual framework. Previous work has studied *grokking*, a training dynamic in which a sustained period of near-perfect training performance and near-chance test performance is eventually followed by generalization, as well as the superficially similar *double descent*. These topics have so far been studied in isolation. We hypothesize that grokking and double descent can be understood as instances of the same learning dynamics within a framework of pattern learning speeds. We propose that this framework also applies when varying model capacity instead of optimization steps, and provide the first demonstration of model-wise grokking.

1 Introduction

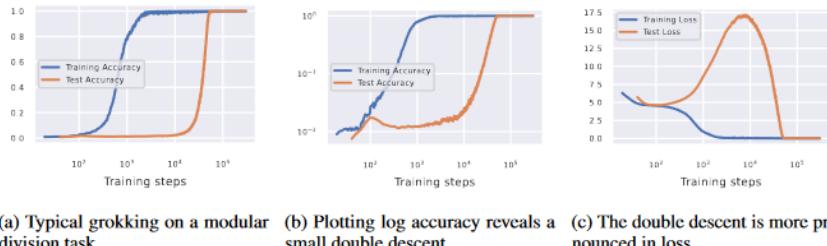


Figure 1: Three views of the same training run, replicated from Power et al. (2022).

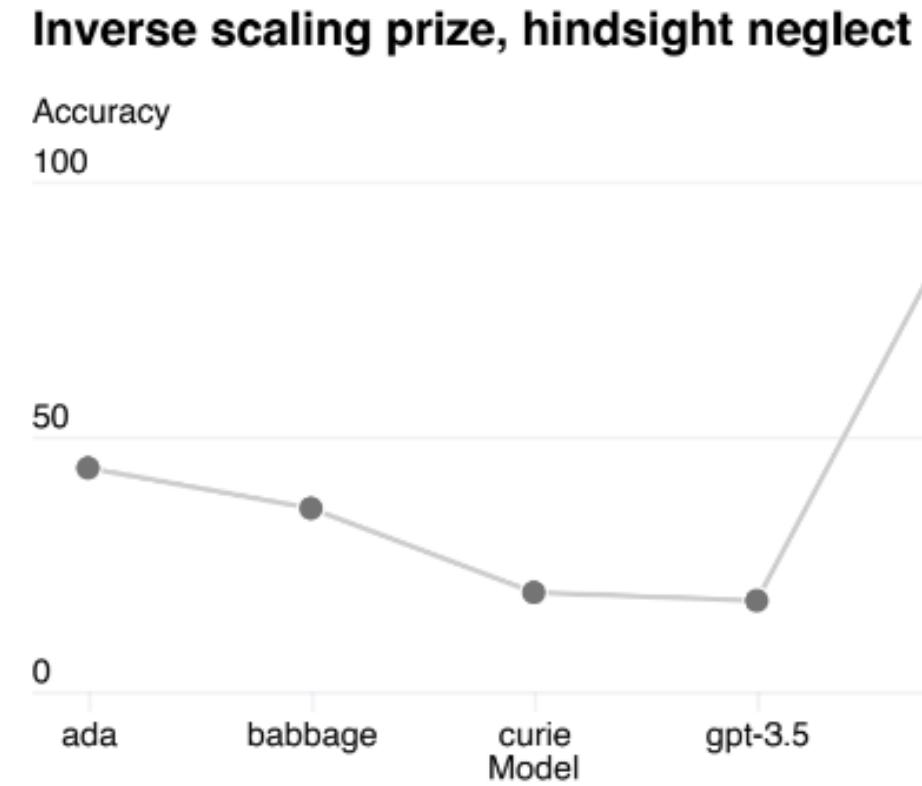
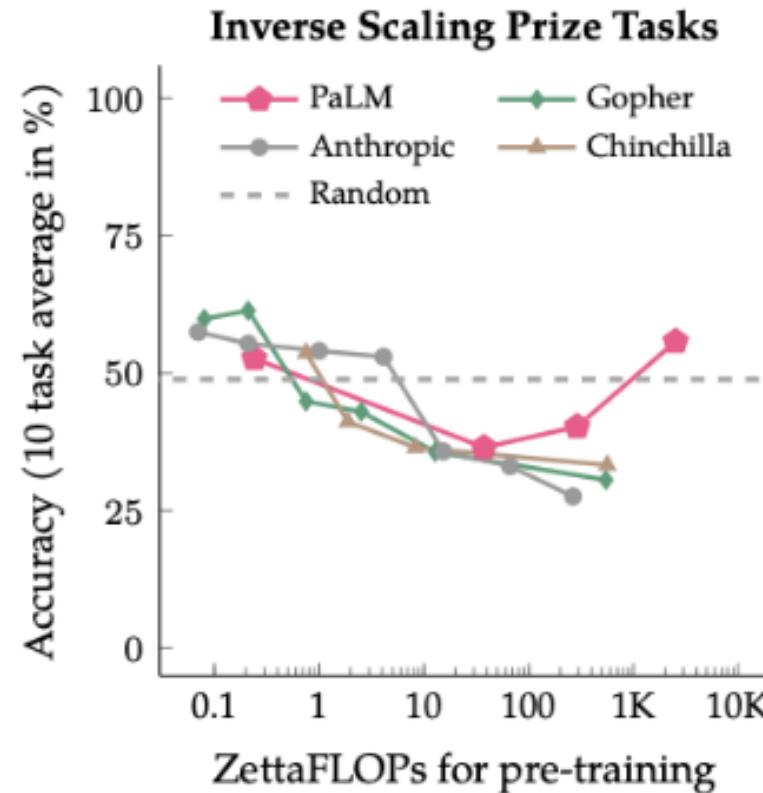
On some datasets, neural networks exhibit surprising training dynamics termed *grokking* by Power et al. (2022). In grokking, the model initially overfits, achieving perfect performance on the training set while remaining at near-chance performance on the test set. Later in training, test performance improves and the model eventually achieves perfect test accuracy. Grokking is reminiscent of the *double descent* phenomenon (Belkin et al., 2018; Nakkiran et al., 2021), in which test performance initially improves, then worsens as the model overfits, and then eventually improves again as model capacity increases.

We argue that double descent and grokking are best viewed as two instances of the same phenomenon, in which inductive biases prefer better-generalizing but slower to learn patterns, leading to a transition from poorly-generalizing to well-generalizing patterns. This transition happens both *epoch-wise* as a function of training time (Figure 1) and *model-wise* as a function of model size (Figure 4).

*Equal contribution. Correspondence to alexander_davies@college.harvard.edu.



Inverse scaling is double descent?



Future directions

The standard GPT loss has a bug?

- The loss is the sum of prediction losses of all tokens of sample texts
- This implicitly means that the training data is a mixture of all sentence lengths: 1 token, 2 tokens,...
- The loss cares equally much about predicting short and long sentences, although in practice we mostly care about the long ones
- Hypothesis: This explains the “recency bias” common in LLMs and contributes to the difficulty of utilizing long-range dependencies
 - Training dynamics: the biased loss might increase the “resistance” for grokking, i.e., emergence of a better but harder solution

How to fix the mode RLHF mode collapse?

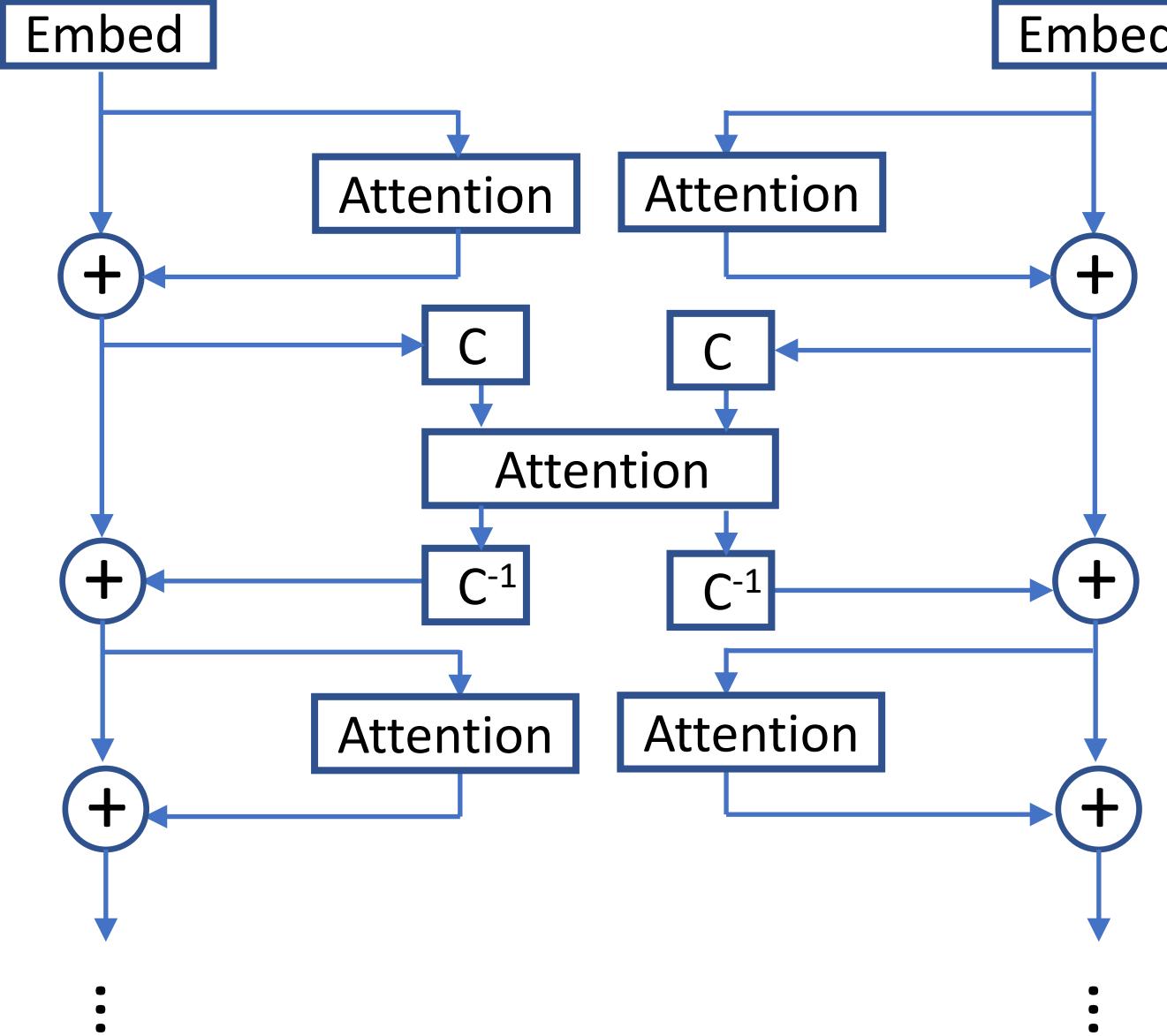
- “Anchoring” similar to Stable Diffusion Dreambooth: Add unconditional samples from the original non-finetuned model (or the pre-training dataset, if available) to the finetuning dataset
- Base and finetuned models as generator-critic pairs?
 - Generator could produce diverse variations of next story paragraph, critic then chooses the best and corrects writing mistakes.
- Quality-Diversity RL instead of PPO
 - QD-RL algorithms are an emerging area
 - <https://dl.acm.org/doi/fullHtml/10.1145/3561975.3562943> (our own)
 - <https://dl.acm.org/doi/abs/10.1145/3512290.3528845>
 - <https://arxiv.org/abs/2303.06164>

Hierarchical attention for effective handling of long context?

- Human brain combines bottom-up and top-down message passing
- Possible transformer implementation: interleave local attention within text segments and global attention that attends to all segments
 - Global attention without quadratic growth of computational cost: Treat each segment as a single token (e.g., concatenate segment tokens before the Q,K,V extraction)

Segment 1 tokens

Segment 2 tokens



The C and C^{-1} blocks map the token embeddings of a segment to a segment embedding and back.

This could even be a simple concatenation/reshape?

RQ: Does this architecture allow cross-segment induction heads?

Segment 1 tokens

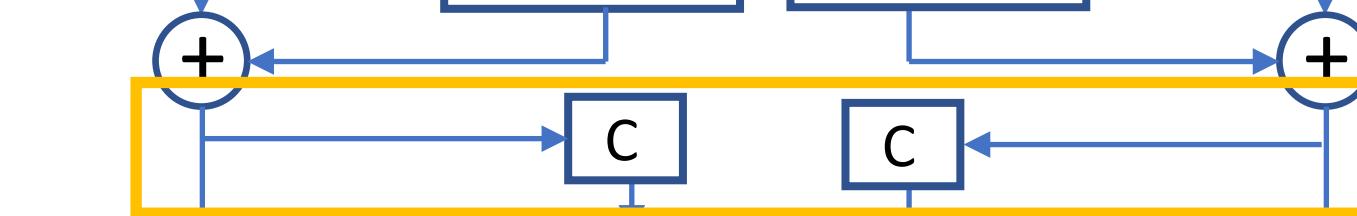
Segment 2 tokens

Embed

Embed

Attention

Attention



Bottom-up messages

Top-down messages

C

C

C^{-1}

C^{-1}

Attention

Attention

Attention

Attention

+

+

⋮

⋮

Evidence for interleaved local-global attention

- Comparisons of interleaved vs. early and late global attention or “contextualization”
- The paper is short on details about how the segment tokens are computed => should investigate the code
- The paper does not look into emergence of induction heads or other fundamental capabilities

An Exploration of Hierarchical Attention Transformers for Efficient Long Document Classification

Ilias Chalkidis^{*}† Xiang Dai[‡] Manos Fergadiotis[◦]

Prodromos Malakasiotis[◦] Desmond Elliott^{†,✉}

† Department of Computer Science, University of Copenhagen, Denmark

‡ CSIRO Data61, Sydney, Australia

◦ Department of Informatics, Athens University of Economics and Business, Greece

✉ Pioneer Centre for AI, Copenhagen, Denmark

Abstract

Non-hierarchical sparse attention Transformer-based models, such as Longformer and Big Bird, are popular approaches to working with long documents. There are clear benefits to these approaches compared to the original Transformer in terms of efficiency, but Hierarchical Attention Transformer (HAT) models are a vastly understudied alternative. We develop and release fully pre-trained HAT models that use segment-wise followed by cross-segment encoders and compare them with Longformer models and partially pre-trained HATs. In several long document downstream classification tasks, our best HAT model outperforms equally-sized Longformer models while using 10-20% less GPU memory and processing documents 40-45% faster. In a series of ablation studies, we find that HATs perform best with cross-segment contextualization throughout the model than alternative configurations that implement either early or late cross-segment contextualization. Our code is on GitHub: <https://github.com/coastalcpb/hierarchical-transformers>.

1 Introduction

Long Document Classification is the classification of a single long document typically in the length of thousands of words, e.g., classification of legal (Chalkidis et al., 2022) and biomedical documents (Johnson et al., 2016), or co-processing of long and shorter chunks of texts, e.g., sequential sentence classification (Cohan et al., 2019), document-level multiple-choice QA (Pang et al., 2021), and document-level NLI (Koreeda and Manning, 2021).

One approach to working with long documents is to simply expand standard Transformer-based language models (BERT of Devlin et al. (2019), RoBERTa of Liu et al. (2019), etc.) but this is problematic for long sequences, given the $O(N^2)$

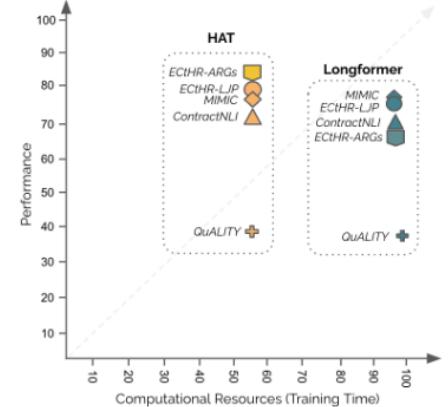


Figure 1: Performance - Efficiency trade-off for HAT and Longformer on downstream tasks.

self-attention operations. To address this computational problem, researchers have introduced efficient Transformer-based architectures. Several sparse attention networks, such as Longformer of Beltagy et al. (2020), or BigBird of Zaheer et al. (2020), have been proposed relying on a combination of different attention patterns (e.g., relying on local (neighbor), global and/or randomly selected tokens). Another approach relies on Hierarchical Attention Transformers (HATs) that use a multi-level attention pattern: segment-wise followed by cross-segment attention. Ad-hoc (partially pre-trained), and non-standardized variants of HAT have been presented in the literature (Chalkidis et al., 2019; Wu et al., 2021; Chalkidis et al., 2022; Liu et al., 2022; Dai et al., 2022), but the potential of such models is still vastly understudied.

In this work, we examine the potential of fully (end-to-end) pre-trained HATs and aim to answer three main questions: (a) Which configurations of segment-wise and cross-segment attention layers in HATs perform best? (b) What is the effect of pre-training HATs end-to-end, compared to ad-

*Corresponding author: ilias.chalkidis[at]di.ku.dk

<https://arxiv.org/abs/2210.05529>



Helping GPT-3 with a creativity tool designed for humans: VNA (verb, noun & adjective cards for game ideation)

Award-winning game ideas based on a verb, a noun, and an adjective:

Verb: Move

Noun: Time

Adjective: Careful

Game idea: Time only moves when the player moves. This allows the player to slowly and carefully maneuver inside a cloud of incoming bullets.

Verb: Push

Noun: Block

Adjective: Rule-based

Game idea: The game world consists of blocks that the player can push around. The blocks represent objects and logical operators, and blocks that form logical expressions define the game's rules. For instance, the three-block expression "key is win" means that the player wins the game by getting the key, and "floor is lava" means that the player burns and dies if touching the floor.

Verb: Program

Noun: Mushroom

Adjective: Blue

Game idea: The player controls a mushroom that can be programmed to run, turn, move, or eat. The mushroom has access to a pool of blue mushrooms, which it can program to do its bidding. The game is over when the player-controlled mushroom eats itself.

Verb: Throw

Noun: Light

Adjective: Heavy

Game idea: The player controls a small light source that can be thrown around. The player can throw the light source into the darkness to illuminate it. The game is over when the light source gets too heavy to be thrown.

Questions?

- Preliminaries:
 - Defining emergence
 - Next token prediction is “AI-complete”
 - Transformer attention
 - The residual pathway as a working memory
- Forms of emergence:
 - Emergence from multiple interacting layers
 - Emergence over multiple sampling operations
 - Emergence over multiple training iterations
 - Emergence in interaction with humans
 - Emergence in interaction with other AI modules
- Future directions

Further reading

- Emergence of LLM abilities has been critiqued and questioned, but the 1st author of the emergence paper provides a convincing rebuttal:
<https://www.jasonwei.net/blog/common-arguments-regarding-emergent-abilities>