GPTs for Those Who Know and Love OLS: The Statistics of Large Language Models

By Alyssa Bilinski, PhD*
Departments of Health Services, Policy, and Practice & Biostatistics, Brown University, Providence, RI
alyssa_bilinski@brown.edu

Jeremy Goldwasser
Department of Statistics, University of California Berkeley, Berkeley, CA

S. Ozair Ali
Department of Health Services, Policy, and Practice Brown University, Providence, RI

Summary

Large language models (LLMs) have become increasingly ubiquitous, but most researchers interact with them through chat interfaces rather than as statistical models. This paper provides an overview of how generative pre-trained transformers (GPTs) work for researchers with backgrounds in biostatistics, epidemiology, or health economics. We frame GPTs as an extension of familiar statistical methods: ordinary least squares, generalized linear models, and neural networks. We then describe the specific features that enable GPTs to generate text at scale across applications, including tokenization, embeddings, and the attention mechanism that allows models to weigh the relevance of different parts of an input sequence. Throughout, we emphasize that the mathematical operations underlying these models (e.g., matrix multiplication, gradient descent, softmax transformations) are conceptually accessible to researchers with quantitative training, even as optimal architectures and training procedures remain areas of active research. We conclude by discussing factors that have driven recent improvements in model performance, including preference learning, increased scale, expanded context windows, chain-of-thought reasoning, and supporting infrastructure.

* We are grateful for inspiration, background and feedback from Alex D'Amour, Natalia Emanuel, Jeffrey Imai-Eaton, Adam Jermyn, Alex Reinhart; students taking Brown University's PHP 2455a; and participants at Boston University's Biostatistics Seminar, the Harvard TH Chan School of Public Health's Center for Communicable Disease Dynamics seminar (particularly Marc Lipsitch and Kirstin Oliveira Roster), and Indiana University's AI and Public Affairs Workshop (particularly Kevin Bryan, Kosali Simon, and Coady Wing). This work was funded in part by the National Institute of General Medical Sciences (1R35GM155224, AB, JG). The content is solely the responsibility of the authors and should not be construed as views of the funders or others acknowledged. In addition to referenced sources, we used large language models and associated tools (ChatGPT 5.2, Claude Opus 4.5, and Refine.ink) to assist with literature search, review text, and identify errors. Since the start of this project, we have revised our working example from "Every week, the little girl gives treats to a furry, friendly cat" to "Every week, the little girl and boy give treats to a furry, friendly cat." To that end, we also give due acknowledgment to the natural generative intelligence in our lives, whose growth has paralleled that of the artificial variety and brought much more joy.

1. Introduction

Following the release of ChatGPT in 2022, researchers have increasingly interacted with large language models (LLMs) [1]. In medicine and health policy, LLMs both support research tasks like coding and classification and are themselves objects of study, with the objective of understanding performance on tasks like exam performance and diagnostic support [2–5]. The preeminent LLM architecture is the generative pre-trained transformer (GPT). Popular GPTs include models from OpenAI (e.g., GPT-4o, GPT-o1, GPT-5.2), Meta (Llama), Google (Gemini), and Anthropic (Claude) [6–9]. The name "GPT" reflects that these models *generate* text after being *pre-trained* on a large corpus (e.g., from the internet) in a self-supervised manner absent explicit labels, employing a neural network architecture called a *transformer* [10].

Most researchers interact with GPTs primarily through chat interfaces, and few work directly with their underlying statistical architecture. Furthermore, because GPTs are recent innovations originating in computer science and industry [10, 11], which have different jargon, notation, and dissemination practices than health and medicine, few researchers encountered them in formal training. We wrote this paper because, in our attempt to learn about LLMs and teach them to our students, it was difficult to find resources with familiar language that built on our mathematical foundations and intuitions. We hope this translation effort may be useful for others in understanding GPTs.

In this work, we provide an overview of how GPTs work for researchers with a background in biostatistics, epidemiology, health economics or other fields outside computer science for which ordinary least squares remains a mainstay. We start by explaining the objective of GPTs: completing text sequences. We then review OLS and generalized linear models (GLMs) as building blocks for neural networks. Last, we describe features that allowed GPTs, a type of neural network, to achieve text generation at scale. Throughout, we highlight that the mathematical operations underlying GPTs should be familiar to those with an understanding of OLS, even as GPT performance, interpretability, and optimal architecture remain active areas of investigation.

2. Text Generation Problems

Most users interact with GPTs through chat interfaces, providing a *prompt* and receiving output text, a *completion* [12–14]. Though completions appear as a single block, they in fact represent multiple iterative model runs [11].

At a high level, GPTs are "text generation" or "completion generation" models: given a sequence of words, they predict what follows [12, 15, 16] (Figure 1). Readers are likely familiar with the convention of notating a prediction $\hat{Y}$ for a true outcome Y based on a vector of inputs $\mathbf{X}$. Here, functions of the user prompt are used to create the $\mathbf{X}$ vector, and GPTs output a predicted probability distribution over possible next words (or sub-words, called tokens) [10]. As an illustration, consider predicting the blank in: "Every week, the little girl and boy give treats to a furry, friendly _____." The $\mathbf{X}$ vector should encode attributes of the preceding words (e.g., the adjectives). A high-quality model would assign probability to various friendly animals, perhaps 40% to dog, 30% to cat, 10% to guinea pig. The precise distribution would depend on the training data: a model trained on sitcoms would produce different predictions than one trained on cartoons (e.g., higher probability of "squirrel") or horror films (e.g., higher probability of "Cujo").

Using this distribution, we generate a prediction for the output sequence $\hat{Y}$ by generating predictions for sequential tokens in an autoregressive process [17]. At each step, the model predicts a distribution over the next token given the input $\mathbf{X}$ and all previously generated tokens. The selected token is appended to the sequence, and this process is repeated until a special end-of-sequence (EOS) token is generated, at which point it terminates [11]. For example, given the sequence described above, the model might first predict the token *guinea*, then *pig*, and finally predict EOS. The result $\hat{Y} = (\{guinea\}, \{pig\})$ constitutes the model's full response.

Although text completion models have long been subject of research, prior to transformers, they often performed poorly and were difficult to scale [11]. High-quality text generation requires representing word meaning, position, and context while remaining computationally tractable. The following sections explain how GPTs achieve this.

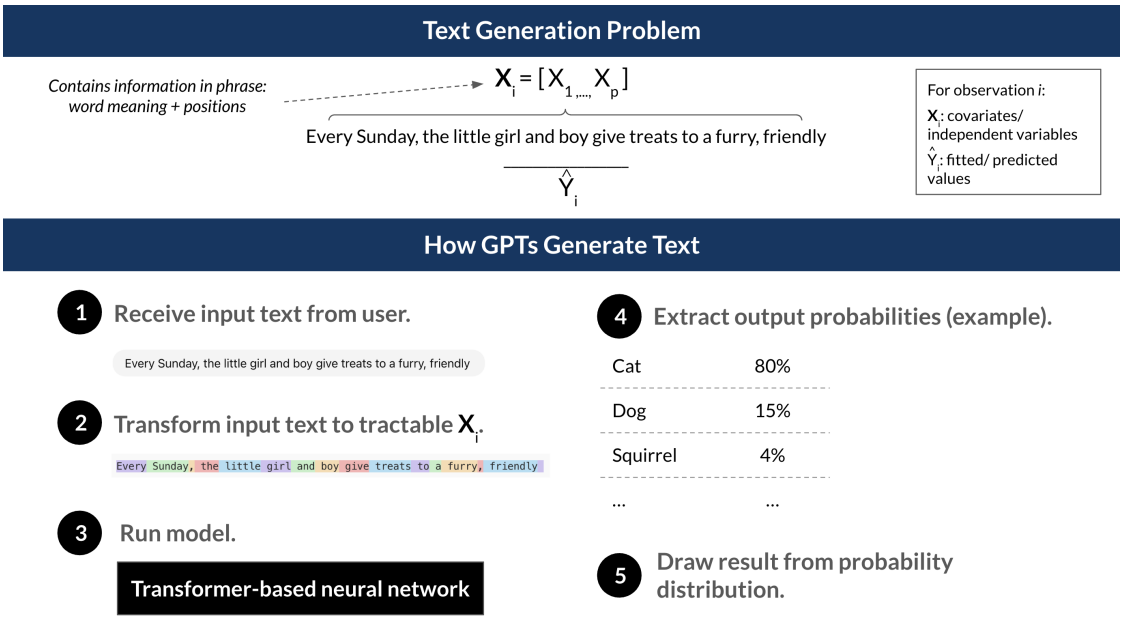

Fig. 1. The text generation problem and an overview of generative pre-trained transformer (GPT) structure.

3. Basic Prediction Models: OLS, GLMs, and Neural Networks

We begin with a brief review of OLS and GLMs and then use these to explain neural networks, the class of machine learning models that includes GPTs (Figure 2). In the next section, we will detail specific features of GPTs.

3.1. *Ordinary least squares (OLS)*

As the workhorse for many scientific analyses, OLS estimates parameters (commonly called $\boldsymbol{\beta}$s) that minimize mean-squared error of predictions of an outcome Y_i for observation i from a linear combination of inputs $\mathbf{X}_i = [1\ x_{1i}\ \ldots\ x_{pi}]$ (a row vector for observation i). We start here not because OLS was a realistic option for natural language processing, but to ground our discussion in the most intuitive and widely-used method in many fields. Given n observations ($i = 1, \ldots, n$) and p predictors, OLS finds

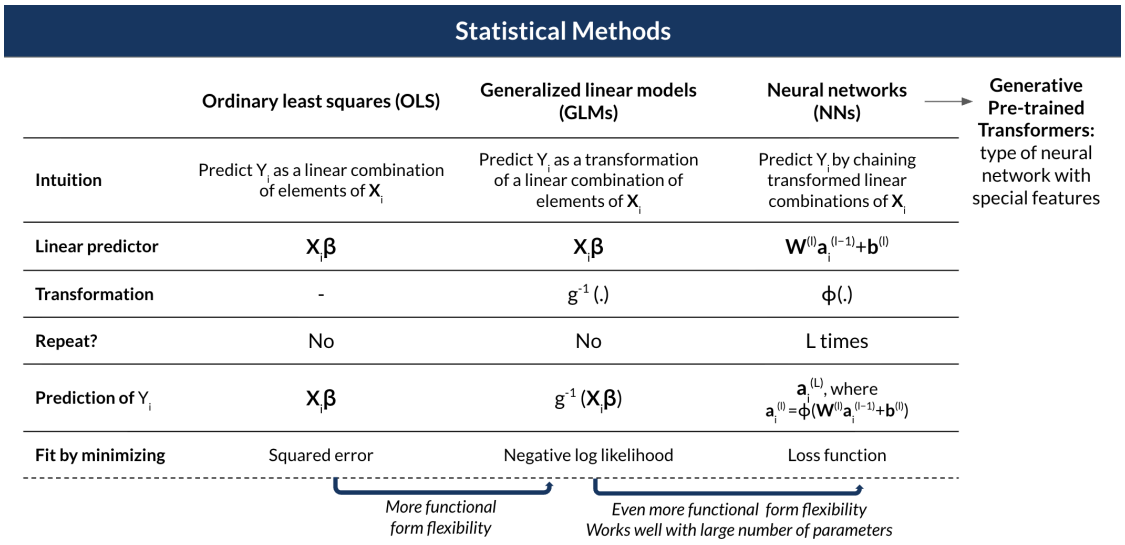

Fig. 2. An overview of statistical methods for prediction problems. Neural networks are shown abstractly; the final layer depends on the task: regression (identity link), binary classification (sigmoid), or multiclass classification (softmax).

parameters that minimize:

$$\arg \min_{\beta_0,\beta_1,\ldots\beta_p} \sum_{i=1}^{n} \left(Y_i - (\beta_0 + \beta_1 x_{1i} + ... + \beta_p x_{pi})\right)^2, \text{ or equivalently,}$$

$$\arg \min_{\boldsymbol{\beta}} \sum_{i=1}^{n} (Y_i - \mathbf{X}_i\boldsymbol{\beta})^2,$$

where $\boldsymbol{\beta} = [\beta_0\ \beta_1\ \ldots\ \beta_p]^T$. We obtain a vector of parameter estimates $\hat{\boldsymbol{\beta}}$ in closed form by taking the derivative of the objective function, setting it equal to 0, and solving for the optimal $\hat{\boldsymbol{\beta}}$. With these estimates, our best prediction for the mean of the outcome, conditional on covariates, is $\hat{Y}_i = \mathbf{X}_i\hat{\boldsymbol{\beta}}$. When there is only one covariate ($p = 1$), we can think of this as generating the best-fit line, in terms of minimizing mean-squared error.

Consider a stylized example of text generation with OLS: letting V be the number of possible next words, suppose $\mathbf{X_i}$ denotes some simple representation of the preceding text. For example, $\mathbf{X_i}$ could be a basis (one-hot) vector indicating the most recent word, a concatenation of vectors indicating the last several, or counts of word frequency (a "bag of words"). With this representation, we could then fit V separate OLS regressions, each predicting a binary outcome for whether the next word takes a particular value. This

would yield next-word predictions $\hat{Y}_i = \{\hat{Y}_{i1}, \ldots, \hat{Y}_{iV}\}$ for any $\mathbf{X}_i$. We might normalize these to represent probabilities, potentially inspired by linear probability models:

$$\hat{p}_{iw} = \frac{\hat{Y}_{iw}}{\sum_{j=1}^{V} \hat{Y}_{ij}}.$$

To generate new text, we could then draw from the resulting distribution. For example, we might deterministically select the word corresponding to the largest $\hat{p}_{iw}$. Alternatively, we could randomly draw words with the probability of choosing each word represented by $\hat{p}_{iw}$.

Though this helps us set up the problem, it is unsurprisingly not a viable approach. Language has complex structure that does not lend itself to simple linear representation. OLS estimates are unbounded, meaning probability predictions could be negative. Directly modeling probabilities themselves could conceivably yield better predictions. Last, OLS performs poorly with many predictors, with reduced efficiency (i.e., greater variance) as the number of parameters grows relative to n. When p exceeds n, parameter estimates are undefined (absent regularization). To model complex features and interactions of words, we will need a different approach.

3.2. *Generalized linear models (GLMs)*

Generalized linear models (GLMs) allow a partial relaxation of the functional form assumptions imposed by OLS [18]. This class of models includes OLS as well as logistic and Poisson regression. Rather than modeling the mean as a linear function of parameters, GLMs model a monotonic transformation of the mean as a linear function of parameters. We call this transformation the link function $g(\cdot)$. We solve:

$$\arg\min_{\boldsymbol{\beta}} \sum_{i} \big[-\log p\big(Y_i \mid \mu_i(\boldsymbol{\beta})\big)\big]$$

where $\mu_i(\boldsymbol{\beta}) = g^{-1}(\mathbf{X}_i\boldsymbol{\beta})$, and $p(Y_i \mid \mu_i(\boldsymbol{\beta}))$ is the likelihood for observation i. OLS arises with an identity link ($g(\mu) = \mu$) and Gaussian likelihood, as minimizing negative log-likelihood is equivalent to minimizing squared error.

Although most GLMs cannot be solved in closed form, parameter estimates can be obtained through gradient descent, which iteratively adjusts parameter estimates along the gradient of the log-likelihood (loss function) until it reaches stable values [18]. We then predict the mean of Y_i as $\hat{Y}_i = g^{-1}(\mathbf{X}_i\hat{\boldsymbol{\beta}})$.

To predict the next word from a vocabulary of V candidates on a probability scale, we can adopt the framework of multinomial logistic regression. For each candidate word w, we estimate a linear function $\mathbf{X}_i\boldsymbol{\beta}_w$, which is a raw logit-transformed score reflecting how likely that word is to follow. These can be converted to probabilities via the softmax function:

$$\hat{p}_{iw} = \text{softmax}\left(\mathbf{X}_i\hat{\boldsymbol{\beta}}\right)_w = \frac{\exp\left(\mathbf{X}_i\hat{\boldsymbol{\beta}}^w\right)}{\sum_{j=1}^{V}\exp\left(\mathbf{X}_i\hat{\boldsymbol{\beta}}^j\right)}$$

Exponentiation ensures all values are positive, and the denominator normalizes them to sum to one. Softmax also amplifies differences between inputs: larger values receive disproportionately more probability, concentrating mass on the most likely outcomes. We return to this property when discussing temperature in Section 4.5.

Overall, GLMs enable modeling a wider range of functional forms than OLS. However, they still rely on linear structure relating $\mathbf{X}_i$ to the transformed mean $g(E[Y_i|\mathbf{X}_i])$ and face similar limitations as OLS in terms of performance with high-dimensional inputs.

3.3. *Neural networks*

Neural networks can be thought of as chaining GLM-like transformations to more flexibly model outcomes and work well with large p relative to n [19, 20]. The building block is a *neuron*, which has the same basic structure as a GLM: a linear combination of inputs, transformed by a nonlinear function. Given a vector $\mathbf{a}$, a single neuron $f(\mathbf{a})$ computes $\phi(\mathbf{a}\mathbf{w} + b)$ [20]. The transformation ϕ is known as the "activation function,"

and is analogous to the inverse link function g^{-1} in a GLM. The "weights" $\mathbf{w}$ and "bias" b are analogous to the GLM parameters $\boldsymbol{\beta}$. (Here the i denoting a specific observation is dropped following computer science notation.) Note that although the general structure of a neuron is similar to that of a GLM, neural networks are not motivated by modeling the distribution of Y, and the requirements for a GLM need not be strictly met. For example, a neural-network neuron can use non-canonical link/activation functions (e.g., a Rectified Linear Unit (ReLU) function, Appendix Figure S1), and need not assume exponential family error distributions.

Neural networks organize neurons into "layers" (Figure 3). The first layer of a neural network is the "input layer" ($\ell = 1$) and consists of the sample value, $\mathbf{a} = \mathbf{x}$. That layer has p neurons, with each neuron corresponding to one of the elements in $\mathbf{x}$. The next layer of a neural network is created from several neurons, with the output of each mapping onto an input in the following ($\ell = 2$). For deeper networks, these neuron then become the inputs that define the third layer, and so on. Layers between the input and output layers are called "hidden" layers because users do not interact with these values. The final layer is called the "output" layer.

The value of a single node layer $\ell > 1$ can be written [20]:

$$a_j^{(\ell)} = \phi\left(\sum_k w_{jk}^{(\ell)} a_k^{(\ell-1)} + b_j^{(\ell)}\right).$$

where $a_j^{(\ell)}$ is the value of the j-th neuron in the ℓ-th layer, $a_k^{(\ell-1)}$ is the value of the k-th neuron in the $\ell - 1$-th layer, $w_{jk}^{(l)}$ is the weight connecting neuron k in layer $\ell - 1$ with neuron j in layer ℓ, and $b_j^{(\ell)}$ is the bias term for layer ℓ. Alternatively, in vector form, we may write:

$$\mathbf{a}^{(l)} = \phi\big(\mathbf{w}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}\big)$$

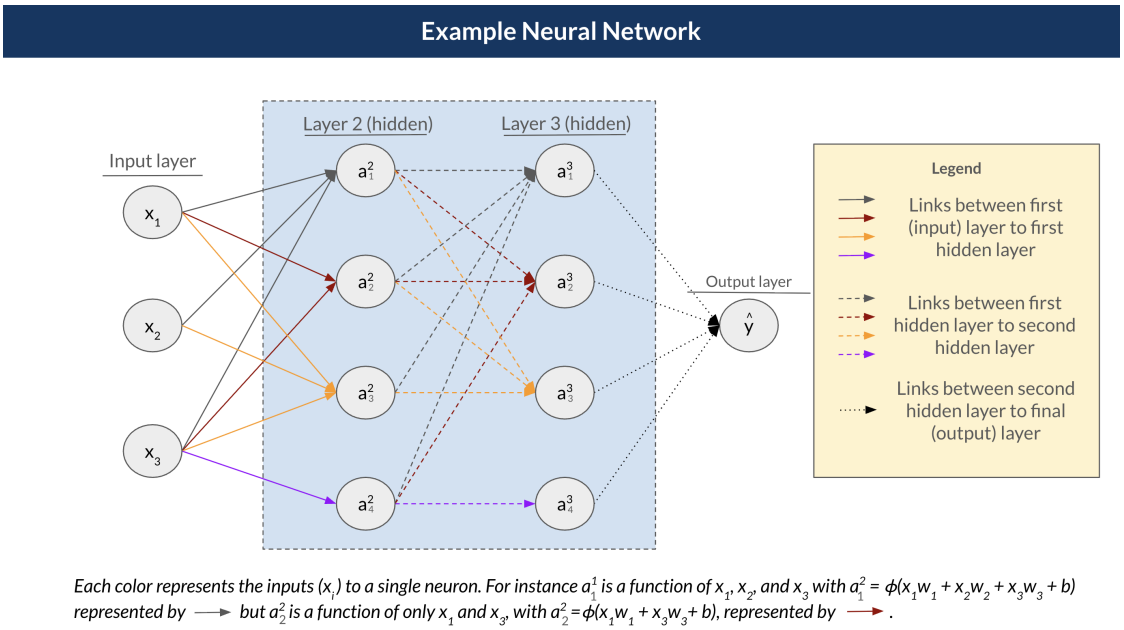

Fig. 3. An example neural network, demonstrating a single input layer, two hidden layers and an output layer. The arrows illustrate how variables are used from one layer to compute the next.

As with GLMs, neural networks are fit using gradient descent. Computing gradients through multiple layers requires the chain rule, working backward from output to input, a procedure called backpropagation [20].

In theory, two-layer neural networks with arbitrary hidden dimension can approximate nearly any functional form [20, 21]. However, to make learning from data tractable, networks are often provided additional structure and typically stack many hidden layers together (i.e., "deep learning") [20]. For example, to process images, researchers use convolutional neural networks (CNNs) that take weighted sums over groups of pixels of different sizes to detect edges and objects [22]. However, while CNNs worked well for image recognition, they struggled to encode longer-range dependencies between words.

Recurrent neural networks (RNNs) were designed to address this by updating a hidden state when processing each word in each layer, to be passed forward for future computations [23, 24]. Although RNNs were initially a promising approach to text completion and dominated the natural language space for several years, they struggled to capture relationships across longer context windows and were slow to train because they

could not be fully parallelized [11]. In the next section, we discuss the architecture ("the transformer") and other features of generative pre-trained tranformers that dramatically improved next-token prediction at scale.

4. Generative Pre-Trained Transformers

Generative pre-trained transformers (GPTs) are a class of neural network optimized for predicting the next word in a sequence (Figure 1). In this section, we describe the core components of GPTs in detail.

As a roadmap, GPTs first convert an input sequence of words into smaller units called "tokens." Tokens are then mapped to vectors called embeddings, which allow researchers to efficiently represent word meaning and position. These components were not unique to transformers or GPTs; tokenization and word embeddings were developed initially with RNNs. Next, embeddings are passed through a neural network with at least one "transformer" block. Transformer blocks include an "attention mechanism", designed to compute how much each preceding token in a sequence should influence predictions of what follows. The output from this neural network defines a probability distribution over all possible next tokens in the model's vocabulary, from which a final output is drawn (with randomness controlled by a temperature parameter). We close the section with discussion model training, as well as factors that determine their performance.

As GPTs have advanced, architectural details have become more proprietary. Therefore, we describe core structures from published early models here to provide a useful foundation, noting that more recent models may include additional advancements.

4.1. *Tokenization*

As the first step of processing input text, words are mapped onto "tokens", either entire words or chunks of words that carry some meaning. Prefixes and suffixes may be tokens; for example, the word "jumping" may be broken into "jump" and "ing" [25]. On average, tokens used by OpenAI as of November 2025 contained 4 characters and represented $\frac{3}{4}$th of a word [26].

The set of possible tokens is determined by algorithms that identify frequently occurring sequences of characters and group these into tokens until the total number of unique tokens reaches the target vocabulary size [27, 28]. Tokenization tends to compress the word vocabulary into a smaller set of subwords [29, 30]. For instance, GPT-2 used 50,000 tokens and GPT-4o about 200,000, compared to roughly a million words in English [29–31]. By breaking words into smaller, reusable pieces, tokenization allows related forms to share information (e.g., *some* in both *something* and *somewhere*).

To ensure that all characters can be mapped to tokens, modern tokenizers operate on bytes (of which there are 256) rather than characters, ensuring that any text, including misspellings, novel words, or emoji, can be represented without requiring an "unknown" token [32].

4.2. *Embeddings*

Tokens are then mapped to high-dimensional vectors called "embeddings", which serve as model inputs. Embeddings are designed so that mathematical operations on them like cosine similarity (a normalized dot product) and vector addition reflect the semantic meaning of tokens and, separately, their position within a sequence. By capturing these relationships in a continuous vector space, models can represent complex interactions using far fewer dimensions than would be required by simpler representations. There are two types of embeddings: semantic and positional embeddings.

Semantic embeddings

Semantic embeddings represent the meaning of tokens. In a naive setup, we could imagine that we would represent each token as a unique basis vector – e.g., a $= \begin{bmatrix}1\,0\,0\,\ldots\end{bmatrix}^T$, the $= \begin{bmatrix}0\,1\,0\,\ldots\end{bmatrix}^T$. In this setup, the dimensionality of the vector space would have to be at least as large as the vocabulary, and mathematical operations between vectors would not yield linguistically meaningful results. (For example, all dot products would be zero.) Capturing relationships between tokens, such as similarity or compositional meaning, would therefore require explicitly modeling two-way or higher-order interac-

tions between all vectors, requiring many parameters and scaling poorly with sequence length.

In 2013, researchers at Google released a method to address this in the package *word2vec* [33]. Their approach learned low-dimensional vector representations such that mathematical operations on them reflected semantic relationships. For example, tokens that were similar to each other (e.g., "emperor" and "king") or found in similar contexts (e.g., "Berlin" and "Germany") had a higher cosine-similarity score [17, 33]. Simple algebraic operations on the vectors could also yield intuitive results: e.g. vector("King") - vector "Man") + vector("Woman") resulted in a vector that was close, as calculated by cosine similarity, to vector("Queen") [34].

Most GPTs today learn their own embeddings as part of an end-to-end training process. OpenAI's GPT-3 model, the last model for which they published architectural details, used embeddings of length 12,288 [35].

Positional embeddings

Each token in the sequence is also mapped to a positional embedding of the same length as the semantic embedding, which encode where the token appears in the sequence relative to other tokens. In early models, positional embeddings were constructed using sinusoidal functional so that relative position was recoverable from cosine similarity [11]. For instance, in the sequence "Every week, the little girl and boy give treats to a furry, friendly", the dot product of the first position (corresponding to Every) and the second position (corresponding to week) was higher than the dot product of the first position and the fifth position (corresponding to girl).

As with semantic embeddings, more contemporary models (e.g., the OpenAI GPTs) learned positional embeddings, with each position's embedding a trained parameter [10, 29, 35]. Other models (e.g., LLAMA-2) incorporated position differently, integrating it later into the attention mechanism rather than at the input layer [36, 37]. The key intuition nevertheless remains that models seek to represent both meaning and position.

Model inputs

Transformers generally combine semantic and positional embeddings for each token. The original Vaswani et al. [11] paper proposed adding positional embeddings to the semantic embedding at the first layer of the network. In high-dimensional space, randomly chosen vectors tend to be nearly orthogonal, so adding two embeddings may largely preserved both signals while concatenation would double the dimensionality and slow training. The resulting $n \times d_{\text{model}}$ matrix (in which d_{model} represents the chosen length of the final vector embeddings) would be the model input. This has remained a standard choice, with both absolute and learned positional embeddings [29, 38]. Alternatively, some models add positional embeddings into various stages of the attention blocks [36, 37, 39, 40].

4.3. *The attention mechanism*

Once a sequence is mapped to embeddings, it is then passed into a neural network with a transformer architecture, displayed in Figure 4. The key breakthrough in this design was using an "attention mechanism" without recurrence (i.e., the hidden memory states as had been used in RNNs but hindered parallelization), hence the title of the seminal paper: "Attention is All You Need" [11]. In this section, we describe the standard attention mechanism design before moving to other aspects of the transformer.

Consider the text completion task used throughout this paper: "Every week, the little girl and boy give treats to a furry, friendly _____." To predict the missing token, the model must use context. Some tokens are clearly more relevant than others for guessing what comes next. For example, "furry" and "friendly" strongly suggest that the missing token is a noun and, specifically, a pet. Conversely, tokens like "Every" or "week" provide relatively little information to guide this prediction.

The attention mechanism (Figure 5) allows a model to determine which tokens in the input sequence are most relevant for predicting each output [11, 17]. Importantly, these relevance weights can be computed in parallel, enabling the approach to scale efficiently to long sequences. Broadly, it has a similar structure to search and retrieval algorithms

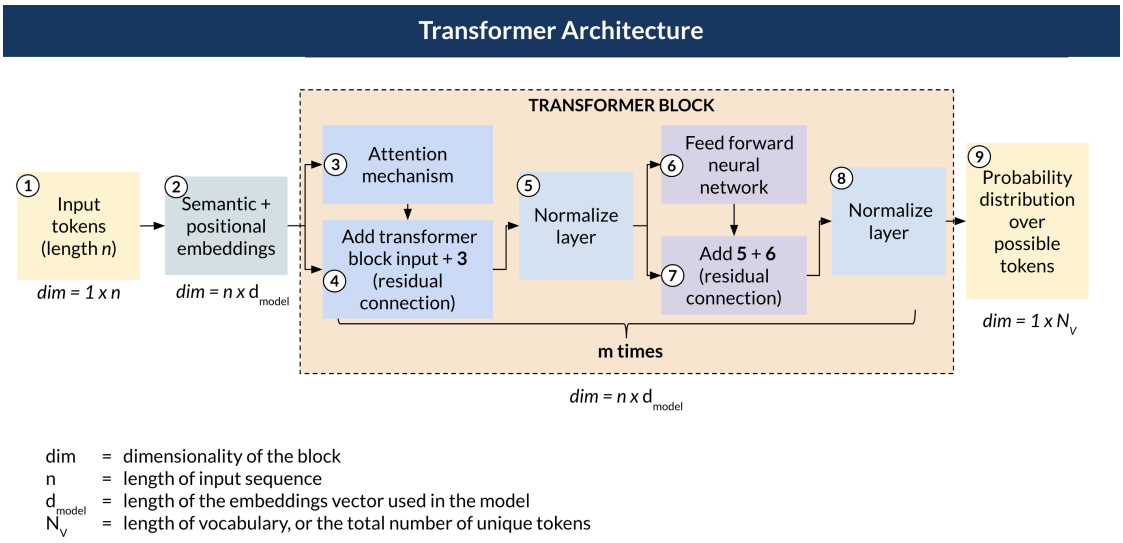

Fig. 4. A simplified transformer architecture that takes in a sequence of tokens as input and produces a probability distribution over all possible next tokens as output.

that use key-value pairs and queries [41]. To illustrate, consider a search algorithm in a video website. The value for each video could be the URL associated with a video, and the key is a summary of the video contents (e.g., a title, a series of tags, or both). A query may be what a user searches for in the search bar. For instance, a user may type in "cute cat video" as a query. A search and retrieval algorithm will typically calculate a similarity score between the query and all keys stored within the database, and sort the keys by score. Finally, the algorithm will return the value (in this case the video) associated with the key most similar to the query. When fitting an LLM, researchers estimate parameters that allow us to define a key matrix (characterizing what a token contains) and a query matrix (characterizing what information a token seeks, with the end goal of predicting the next token). The value matrix then characterizes what each token contributes to when matched.

Mathematically, the attention mechanism computes:

$$\text{Attention}\left(\mathbf{X}_n\right) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V}$$

where $\mathbf{Q}$ (the query matrix), $\mathbf{K}$ (the key matrix), and $\mathbf{V}$ (the value matrix) are linear transformations of $\mathbf{X}_n$ (defined below).

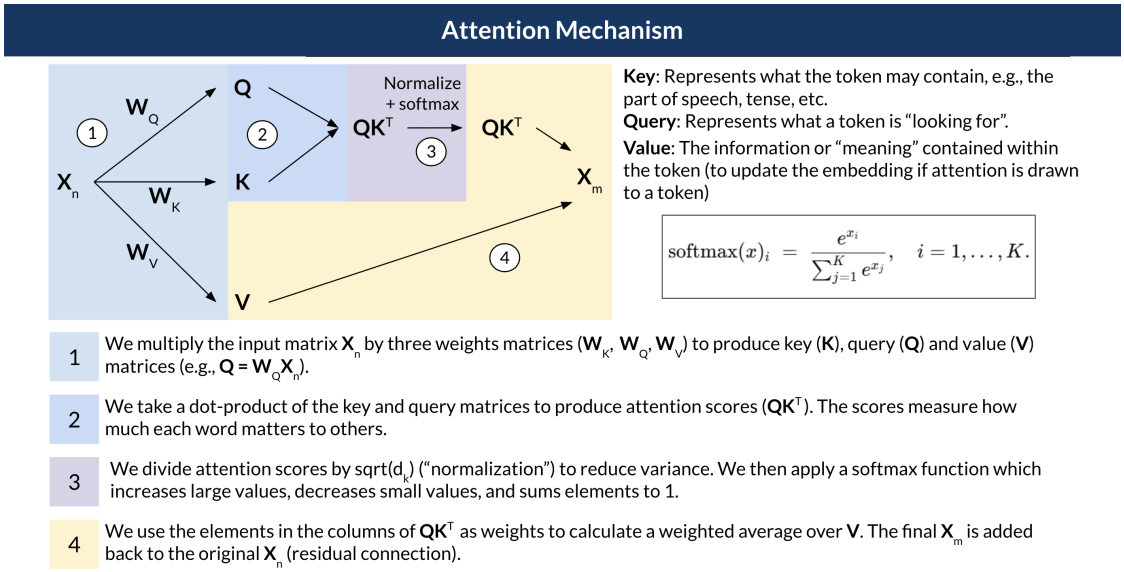

Fig. 5. The attention mechanism described in detail as a sequence of matrix multiplication and other operations.

Here is how we can understand these components [11, 17]. The key matrix ($\mathbf{K}$) encodes what information each token "contains" in its attributes that might be relevant to other tokens' queries. It is computed as $\mathbf{K} = \mathbf{X}_n\mathbf{W}_K$, where $\mathbf{W}_K$ is a learned parameter matrix of dimension $d_e \times d_k$, which we can think of as analogous to $\boldsymbol{\beta}$ in a traditional linear model. We can also say $\mathbf{K}$ contains the key vector for each element in the sequence. The query matrix ($\mathbf{Q}$) encodes what each token is "looking for" in surrounding context. For instance, the query for "friendly" might encode "looking for a noun to attach to." It is computed as $\mathbf{Q} = \mathbf{X}_n\mathbf{W}_Q$, where $\mathbf{W}_Q$ has the same dimensions as $\mathbf{W}_K$.

Attention scores are computed as the product $\mathbf{Q}\mathbf{K}^\top$, an $n \times n$ matrix which contains dot product between every query vector and every key vector. Each entry (i, j) measures how relevant token j is to token i, where higher values indicate greater relevance. This

matrix is normalized by dividing by $\sqrt{d_k}$, the square root of the key dimension. This normalization prevents the variance of the dot products from growing with d_k, which would cause the subsequent softmax to produce extreme values. The softmax function is then applied to each row, transforming the raw scores into a probability distribution. This exaggerates large values, suppresses small ones, and ensures each row sums to one. The resulting matrix $\mathbf{\Omega} = \text{softmax}\left(\mathbf{Q}\mathbf{K}^\top/\sqrt{d_k}\right)$ contains the *attention weights*.

Because GPTs are autoregressive, each position may only attend to itself and earlier positions. This is enforced by a causal mask: before the softmax, the entries of $\mathbf{Q}\mathbf{K}^\top$ corresponding to future positions are in fact set to $-\infty$, so that their attention weights become zero. The resulting weight matrix $\mathbf{\Omega}$ is therefore lower-triangular.

We last define one additional matrix: the value matrix ($\mathbf{V}$) contains the information that will be passed forward if a token is deemed relevant. It is computed as $\mathbf{V} = \mathbf{X}_n\mathbf{W}_V$, where $\mathbf{W}_V$ is of dimension $d_e \times d_e$.

Multiplying $\mathbf{\Omega}$ by $\mathbf{V}$ produces a weighted sum of the value vectors:

$$\mathbf{\Omega V} = \begin{bmatrix} w_{11}\mathbf{v}_{\text{Every}} \\ w_{21}\mathbf{v}_{\text{Every}} + w_{22}\mathbf{v}_{\text{week}} \\ w_{31}\mathbf{v}_{\text{Every}} + w_{32}\mathbf{v}_{\text{week,}} + w_{33}\mathbf{v}_{\text{the}} \\ \vdots \end{bmatrix} \tag{1}$$

Each row of the output is a weighted combination of the current and preceding tokens' value vectors, where the weights reflect how much attention each token pays to the preceding token. Tokens deemed more relevant (via the query-key interaction) contribute more to the output.

Finally, we add this result to update the input in a "residual connection." This helps prevent vanishing gradients (where gradients shrink toward zero as they pass through many layers during backpropagation, stalling learning) and allows the model to learn incremental refinements to token representations (i.e., updating initial embeddings or results of the prior layer) rather than entirely new representations at each layer.

The output of each residual connection is also passed through a layer normalization step [11, 17]. Layer normalization rescales each token's representation (i.e., the result of the embedding being added to the output of the attention mechanism and renormalized) to have zero mean and unit variance. Empirically, this improves numerical stability and accelerates convergence during training, particularly in more complex models.

Each block also includes a feedforward network that applies the same nonlinear transformation independently to each token's representation [11, 17]. Unlike the attention mechanism, which mixes information across tokens, the feedforward network transforms each representation in isolation, using the standard neural network architecture discussed above. This allows the model to further process each token after contextual information has been incorporated through attention. Finally, models often use another residual connection and layer normalization after the feedforward network.

In practice, the attention mechanism is also employed in "multi-headed" setup. Rather than computing attention weights once, the model computes them multiple times in parallel as part of the same attention mechanism step. For example, the original "Attention Is All You Need" paper used 8 heads per attention block [11]. Each head has its own query, key, and value matrices, producing a unique set of attention weights. This allows the model to capture different types of relationships between tokens simultaneously. In principle, one head might learn to attend to syntactic relationships, another to semantic similarity, another to positional patterns, and so on (although roles may not be as clean in practice to human eyes) [42]. The outputs of all heads are concatenated and then projected back to the model dimension d_{model} using another parameter matrix $\mathbf{W}_O$. Because each head has a smaller number of parameters (typically $d_k = d_{\text{model}}/h$, where h is the number of heads), the total computational cost of multi-head attention is similar to that of single-head attention.

In addition to using "multi-headed" attention, models typically stack many such attention blocks in sequence. GPT-3, for instance, used 96 blocks [35]. Each block refines

the token representations further, allowing the model to build increasingly abstract representations of the input.

4.5. *Output*

As its final output, the model produces an updated matrix of token representations, still of dimension $n \times d_e$. To generate a prediction, the model focuses on the embedding corresponding to the final token in the sequence, which we denote $\mathbf{y}_n$. This embedding summarizes the full context observed so far and can be interpreted as the model's best representation of "what should come next."

The model converts this representation into a probability distribution over the vocabulary. One approach to do this is simply to compute the dot product of $\mathbf{y}_n$ with the embedding of every possible token in the language [38]. Recall that embeddings have the property that semantically similar tokens have large dot products. This means that this operation assigns higher value to tokens whose meanings are most compatible with model's prediction of what comes next given context. For the example sentence "Every week, the little girl and boy give treats to a furry, friendly _____," tokens such as "cat" or "dog" will have higher dot products with $\mathbf{y}_n$ than less related tokens (such as "alligator"). These can then be converted to probabilities using a softmax function. (Alternatively, models may have another step of an "unembedding" matrix, which allows this process to be a more flexible transformation.)

To make a prediction, the model may select the token with the highest probability (greedy decoding) or sample from this distribution, which introduces randomness and allows for more diverse text generation. A *temperature* parameter controls how concentrated the distribution is: lower temperatures sharpen the distribution toward the highest-probability tokens, producing more deterministic outputs, while higher temperatures flatten it, introducing more randomness [43]. As noted above, this process is repeated, with each generated token appended to the input sequence, until the EOS token is generated.

4.6. *Model size*

So, how many parameters does an LLM have? Consider a single head of attention operating on a sequence of n tokens with embedding dimension m. The three weight matrices have dimensions [11]:

$$\dim(\mathbf{W}_Q) = \dim(\mathbf{W}_K) = d_e \times d_k \tag{2}$$

$$\dim(\mathbf{W}_V) = d_e \times d_v \tag{3}$$

where d_k and d_v are the dimensions of the query/key and value vectors respectively. A single head thus contributes $2d_e d_k + d_e d_v$ parameters.

In many LLMs like GPT-3, $d_k = d_v = d_e/m$, where m is the number of heads [29, 35]. In this case, the total parameters across all m heads for the $d_e \times d_e/m$-dimensional Q, K, and V matrices is $3d_e^2$. An additional matrix $\mathbf{W}_o$ of dimension $d_e \times d_e$ combines the conjoined head outputs. The attention mechanism therefore contains $4d_e^2$ parameters per block. The feedforward network added further parameters. In GPT-3, two layers with an inner dimension of $4d_e$ contributed $8d_e^2$ parameters [35]. Thus, each of the b blocks has $12d_e^2$ parameters, yielding a total count of roughly $12d_e^2 b$.

The resulting models are enormous, due to large embeddings and a deep stacks of blocks. GPT-3 used $m = 12{,}288$ and $b = 96$, totaling about $12m^2 b = 174$ billion parameters. Token and position embeddings contributed to the final tally of 175 billion parameters [35]. Llama 3.1, released by Meta in 2024, is over twice as large. With $16,384$-dimensional embeddings and 128 blocks, it totals 405 billion parameters [44]. DeepSeek's V3 model contains over 600 billion parameters; GPT-4 and GPT-5 are rumored to be significantly larger.

4.7. *Training LLMs*

Training LLMs is feasible for several reasons [11]. First, the core operations in a transformer, matrix multiplications and element-wise nonlinearities, can be parallelized. Second, training examples can also be processed in parallel batches, allowing gradients to be computed across many examples simultaneously. Third, modern GPUs (particularly

those from NVIDIA) are optimized for exactly these operations, with training distributed across thousands of GPUs simultaneously.

The primary computational bottleneck is the attention mechanism. Computing $\mathbf{QK}^\top$ produces an $n \times n$ matrix, meaning that computational and memory costs scale quadratically with sequence length. This is one reason why models specify a maximum *context window*, the total number of tokens the model can process at once. A model's context window determines how much text it can "see" when generating a response and, in a chat conversation, includes both user input and prior responses. When generating each token, the model considers everything within this window; information beyond it is effectively invisible, as users may encounter when a model appears to forget information from the start of a long conversation.[1]

Training typically consists of 3 training phases: pre-training, supervised fine-tuning, and preference learning. This pipeline was standardized by the InstructGPT paper, an important step preceding the release of ChatGPT [28, 46].

Pre-Training

The first phase of training LLMs is pre-training on vast amounts of data, typically massive corpus of text, comprising books, websites, code repositories, and other webpages. Because pre-training is self-supervised, it does not require labeled data, so any reasonable piece of text may be included in the training corpus; the model simply predicts each next word given the prior sequence. A major innovation of GPT-3 over its predecessor was scaling pre-training: It trained on 300 billion tokens, an order of magnitude bigger than GPT-2 [29, 35]. This scaling has only since continued, with contemporary LLMs pre-training on trillions of tokens [44, 47, 48]. Llama 4 claims to use a staggeringly large dataset of 40 trillion tokens [7].

Data curation is an important part of pre-training, and not all pre-training data is equally valuable. For example, academic papers may be higher-quality text than commercial websites (though some may reasonably disagree), and might be upsampled ac-

[1] As an alternative to GPTs, Mamba is a state-space model whose complexity is linear, rather than quadratic in the sequence length [45]. Although promising, it generally underperforms GPTs; a detailed description is beyond the scope of this work.

cordingly. In addition, not all text that can be scraped should be used to pretrain a model. Internet datasets are rife with offensive or dangerous content, which researchers attempt to filter out. Models trained on such data are known to learn such behaviors. GPT-3 was found to produce racist, sexist, and violent text based on patterns learned in its pre-training [35].

Supervised Learning

Supervised learning adapts a pre-trained model for a range of different tasks. This phase often uses labeled data, with a prompt and a human- or machine-labeled output. Because labeled data are scarce, the total dataset used for supervised fine-tuning is small compared to the pre-training corpus. Pre-training data accounted for 98% of the text used to train InstructGPT [46].

Pre-trained models are fine-tuned on several tasks at once, each with their own labeled datasets. Tasks might include question answering, document summarization, machine translation, open-ended generation, and rewriting. In the pre-LLM era, models would be trained directly on these datasets. However, pre-training improved performance substantially as it instilled a deep sense of linguistic fluency, background knowledge, and perhaps some degree of reasoning ability. Fine-tuning is thought to contribute to hallucinations, as models learn to produce assertive responses even when relevant information was absent from pre-training [49, 50].

Preference Learning

The third stage, preference learning, is another round of fine-tuning. This phase is responsible for much of the "personality" of modern chatbots, including their confident tone, sycophancy, and tendency toward helpfulness [46]. It was a major breakthrough in the transition from GPT-3 to ChatGPT.

The first preference learning procedure introduced for LLMs was reinforcement learning from human feedback, or RLHF. Originally demonstrated in robotics [51], RLHF was applied to language models at scale in InstructGPT [46]. In RLHF, human raters compare different model outputs and indicate which response is preferred. These preferences

are used to train a reward model that predicts human approval. The language model is then fine-tuned to maximize this reward.

The leading alternative to RLHF is Direct Preference Optimization [52]. DPO obviates the need for a reward model and RL training. It fine-tunes the LLM directly on the preference data, maximizing the likelihood that the winning responses are preferred. This approach is computationally cheaper and easier to execute, and thus has become common in practice.

More recently, researchers have explored reinforcement learning with verifiable rewards, for instance, training models on mathematics or coding problems where correctness can be checked automatically, reducing reliance on human judgment [53].

4.8. *Why Are Models Getting So Much Better?*

The prior sections have characterized the general structure of LLMs, but models have drastically improved over the past few years. Here, we outline the factors contributing to this, with the aim of helping readers to understand bottlenecks and anticipate how future models may evolve.

First, models have improved as they have become larger, both in parameter count and training data. GPT-3's 175 billion parameters represented a hundred-fold increase over GPT-2 (1.5 billion) and a thousand-fold increase over GPT-1 (117 million) [10, 29]. Subsequent models have continued this trend; the largest publicly documented models now exceed 400 billion parameters [44]. Researchers have also become better at optimizing the ratio of model parameters to training data [54]. For example, GPT-3 was discovered to have an insufficient number of training points relative to its parameter size.

Second, context windows have expanded dramatically. The original transformer had a context window of 512 tokens [11]; GPT-2 extended this to 1,024 tokens [29], GPT-3 to 2,048 tokens [35], Llama-3 to 128,000 tokens [44] and Gemini reporting up to 10 million in experiments [55]. This expansion increases the model's effective working memory and allows it to perform better in longer conversations. Because context window size remains a key factor affecting both performance and computational costs, strategies have been

developed both to speed computation (e.g., developing optimization methods that make attention faster on modern hardware [56] and improve performance with a given limit (e.g., summarizing or "compacting" of earlier conversation history to reduce token count or using retrieval systems that fetch relevant information on demand rather than keeping everything in context).

Third, the introduction of intermediate reasoning steps, often called chain-of-thought prompting or extended thinking, has substantially improved performance on complex tasks [57]. This involves generating intermediate reasoning steps before producing a final answer and allows models models to tackle multi-step problems that would otherwise exceed their capabilities. This technique has proven particularly effective for mathematical reasoning [53]. Interestingly, it performs well even when models are not actually using the reasoning they are stating within intermediate steps [58, 59].

In addition to direct model improvements, users have also benefited from improvements in supporting architecture, including document processing, integrated code execution environments that make it easy to check computations, and structured output formatting. Still, some challenges persist; for example, when reading PDFs, layout, tables, and multi-column formatting may be lost or misinterpreted during text extraction.

Overall, these improvements have translated into dramatic gains in real-world performance. METR, an AI safety organization, proposed measuring model capabilities by the length of tasks models can complete autonomously with 50% reliability, where task length is defined by how long a task takes human professionals [60]. By this metric, frontier model capabilities have rapidly improved on software engineering tasks, with task length doubling approximately every seven months from 2019 through early 2025. As of late 2025, Claude Opus 4.5 achieved a time horizon of nearly five hours [61]. Notably, the models and configurations available to the public at any given time likely understate the frontier: major AI laboratories typically have more capable models in development that have not yet been released, deploy longer context windows internally than are publicly available, and support customized agent setups that outperform out-of-the-box tools. Scaling laws suggest that model performance improves predictably with

increases in data, parameters, and compute [62], and labs continue to invest heavily in all three. While the future remains uncertain, researchers can anticipate continued substantial growth in model capabilities.

5. Conclusion

This paper demonstrates that increasingly popular and powerful GPTs can be understood as extensions of familiar statistical tools: like OLS and GLMs, they estimate parameters by minimizing a loss function; like other neural networks, they chain nonlinear transformations to model complex relationships. The key innovations enabling modern text generation (tokenization, learned embeddings, and the attention mechanism) address the specific challenges of representing language and capturing dependencies across sequences. Understanding these foundations has practical value: researchers who grasp that models predict probability distributions over tokens, rather than retrieving facts from a database, are better positioned to anticipate failure modes like hallucination; those who understand context windows can structure prompts more effectively; and those who recognize that fine-tuning shapes behavior through human feedback can better interpret why models respond as they do. Some model APIs provide the predicted probabilities for output tokens, offering researchers a familiar statistical object to quantify uncertainty, assess model confidence, or construct more principled decision rules, and some researchers may also explore using GPTs outside text prediction. As LLMs become more integrated into research and society, continued engagement between researchers and the underlying methodology will help ensure these tools are applied appropriately and their limitations understood.

A. Appendix

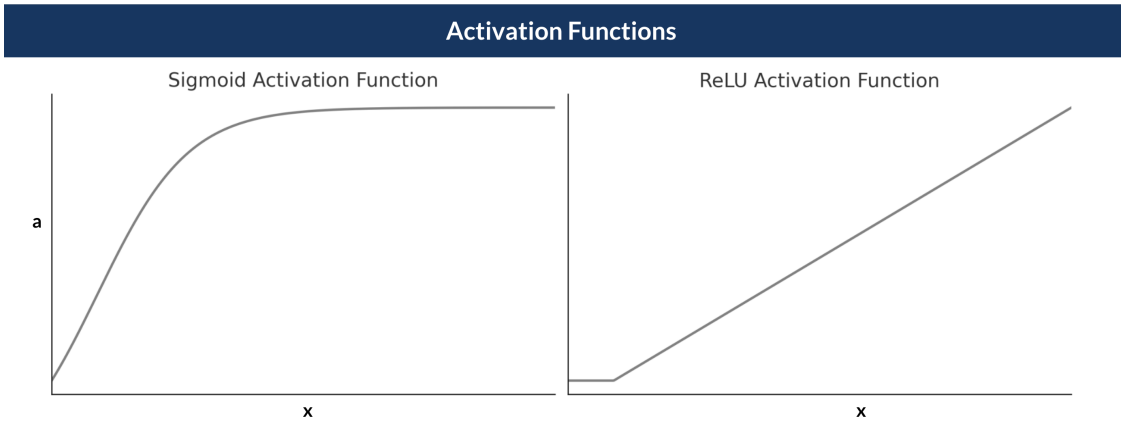

Fig. S1. Sigmoid and ReLU activation functions are both popular for neural networks. While a sigmoid activation function, which is differentiable (left), a ReLU function contains a "kink", below which the value is zero, and the function is linear above the kink.

References

[1] OpenAI. Introducing ChatGPT, November 2022. URL https://openai.com/index/chatgpt/.

[2] Tiffany H. Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, and Victor Tseng. Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. *PLOS Digital Health*, 2(2):e0000198, February 2023. ISSN 2767-3170. . URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9931230/.

[3] Ali Abbas, Mahad S Rehman, and Syed S Rehman. Comparing the Performance of Popular Large Language Models on the National Board of Medical Examiners Sample Questions. *Cureus*, March 2024. ISSN 2168-8184. . URL https://www.cureus.com/articles/203719-comparing-the-performance-of-popular-large-language-models-on-the-national-board-of-medical-examiners Publisher: Springer Science and Business Media LLC.

[4] Alexander V. Eriksen, Sören Möller, and Jesper Ryg. Use of GPT-4 to Diagnose Complex Clinical Cases. *NEJM AI*, 1(1):AIp2300031, January 2024. . URL https://ai.nejm.org/doi/full/10.1056/AIp2300031. Publisher: Massachusetts Medical Society.

[5] Uriel Katz, Eran Cohen, Eliya Shachar, Jonathan Somer, Adam Fink, Eli Morse, Beki Shreiber, and Ido Wolf. GPT versus Resident Physicians — A Benchmark Based on Official Board Scores. *NEJM AI*, 1(5):AIdbp2300192, April 2024. . URL https://ai.nejm.org/doi/full/10.1056/AIdbp2300192. Publisher: Massachusetts Medical Society.

[6] OpenAI. Models - OpenAI API, 2025. URL https://platform.openai.com.

[7] Meta. Llama by Meta, 2025. URL https://www.llama.com/models/llama-4/#models.

[8] Google. Gemini models | Gemini API, 2025. URL https://ai.google.dev/gemini-api/docs/models.

[9] Anthropic. Models overview, 2025. URL https://docs.anthropic.com/en/docs/about-claude/models/overview.

[10] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving Language Understanding by Generative Pre-Training. 2018.

[11] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is All You Need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[12] HuggingFace. What is Text Generation?, 2025. URL https://huggingface.co/tasks/text-generation.

[13] OpenAI. Text generation and prompting, 2025. URL https://platform.openai.com.

[14] Community. Terminology evolution? "completion" vs "response", March 2025. URL https://community.openai.com/t/terminology-evolution-completion-vs-response/1143324. Section: Community.

[15] Vrunda Gadesha. What is text generation?, March 2024. URL https://www.ibm.com/think/topics/text-generation.

[16] Junyi Li, Tianyi Tang, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Pre-Trained Language Models for Text Generation: A Survey. *ACM Comput. Surv.*, 56(9):230:1–230:39, April 2024. ISSN 0360-0300. . URL https://dl.acm.org/doi/10.1145/3649449.

[17] Grant Sanderson. Transformers, the tech behind LLMs | Deep Learning Chapter 5, 2024. URL https://www.3blue1brown.com/lessons/gpt.

[18] P. McCullagh. *Generalized Linear Models*. Routledge, Boca Raton, 2018. ISBN 978-0-412-31760-6.

[19] Andrew Ng and Tengyu Ma. CS229 Lecture Notes, June 2023. URL https://cs229.stanford.edu/main_notes.pdf.

[20] Michael A. Nielsen. Neural Networks and Deep Learning. 2015. URL http://neuralnetworksanddeeplearning.com. Publisher: Determination Press.

[21] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, December 1989. ISSN 1435-568X. . URL https://doi.org/10.1007/BF02551274.

[22] Afshine Amidi and Shervine Amidi. CS 230 - Convolutional Neural Networks Cheatsheet, 2024. URL https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-networks.

[23] Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*, pages 1017–1024, 2011. URL https://icml.cc/Conferences/2011/papers/524_icmlpaper.pdf.

[24] Robin M. Schmidt. Recurrent Neural Networks (RNNs): A gentle Introduction and Overview, November 2019. URL http://arxiv.org/abs/1912.05911. arXiv:1912.05911 [cs].

[25] OpenAI. Tokenizer, 2025. URL https://platform.openai.com.

[26] OpenAI. What are tokens and how to count them?, 2025. URL https://help.openai.com/en/articles/4936856-what-are-tokens-and-how-to-count-them.

[27] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the ACL (ACL 2016)*, pages 1715–1725, 2016. . URL https://aclanthology.org/P16-1162/.

[28] Stanford Online. Stanford CS229 I Machine Learning I Building Large Language Models (LLMs), August 2024. URL https://www.youtube.com/watch?v=9vM4p9NN0Ts.

[29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language Models are Unsupervised Multitask Learners. 2019.

[30] OpenAI. openai/tiktoken, January 2026. URL https://github.com/openai/tiktoken. original-date: 2022-12-01T23:22:11Z.

[31] Merriam-Webster. How many words are there in English?, 2025. URL https://www.merriam-webster.com/help/faq-how-many-english-words.

[32] Hugging Face. Byte-Pair Encoding tokenization - Hugging Face LLM Course, 2024. URL https://huggingface.co/learn/llm-course/en/chapter6/5.

[33] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space, September 2013. URL http://arxiv.org/abs/1301.3781. arXiv:1301.3781 [cs].

[34] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In *Proceedings of the 2013 Conference of the North American Chapter of the ACL: Human Language Technologies (NAACL-HLT 2013)*, pages 746–751, 2013. URL https://aclanthology.org/N13-1090/.

[35] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[36] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing*, 568:127063, February 2024. ISSN 0925-2312. . URL https://www.sciencedirect.com/science/article/pii/S0925231223011864.

[37] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models, July 2023. URL http://arxiv.org/abs/2307.09288. arXiv:2307.09288 [cs].

[38] Daniel Dugas. The GPT-3 Architecture, on a Napkin, 2023. URL https://dugas.ch/artificial_curiosity/GPT_architecture.html.

[39] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of NAACL-HLT*, 2018.

[40] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[41] Ebrahim Pichka. What is Query, Key, and Value (QKV) in the Transformer Architecture and Why Are They Used?, January 2025. URL https://medium.com/data-science/what-are-query-key-and-value-in-the-transformer-architecture-and-why-are-they-used-acbe73f731f2.

[42] Connor Kissane, Robert Krzyzanowski, Joseph Isaac Bloom, Arthur Conmy, and Neel Nanda. Interpreting Attention Layer Outputs with Sparse Autoencoders, June 2024. URL http://arxiv.org/abs/2406.17759. arXiv:2406.17759 [cs].

[43] Max Peeperkorn, Tom Kouwenhoven, Dan Brown, and Anna Jordanous. Is Temperature the Creativity Parameter of Large Language Models? In *Proceedings of the 15th International Conference*

on Computational Creativity (ICCC 2024), 2024. URL https://computationalcreativity.net/iccc24/papers/ICCC24_paper_60.pdf.

[44] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny

Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 Herd of Models, November 2024. URL `http://arxiv.org/abs/2407.21783`. arXiv:2407.21783 [cs].

[45] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL `https://arxiv.org/abs/2312.00752`.

[46] 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, and Ryan Lowe. Training language models to follow instructions with human feedback, March 2022. URL `http://arxiv.org/abs/2203.02155`. arXiv:2203.02155 [cs].

[47] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint*, abs/2204.02311, 2022. URL `https://arxiv.org/abs/2204.02311`.

[48] Deepseek. DeepSeek, 2025. URL `https://www.deepseek.com/`.

[49] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems*, 43(2):1–55, March 2025. ISSN 1046-8188, 1558-2868. . URL `http://arxiv.org/abs/2311.05232`. arXiv:2311.05232 [cs].

[50] Adam Tauman Kalai, Ofir Nachum, Santosh S. Vempala, and Edwin Zhang. Why Language Models Hallucinate, September 2025. URL `http://arxiv.org/abs/2509.04664`. arXiv:2509.04664 [cs].

[51] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL `https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf`.

[52] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024. URL `https://arxiv.org/abs/2305.18290`.

[53] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda

Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, September 2025. ISSN 1476-4687. . URL `https://www.nature.com/articles/s41586-025-09422-z`. Publisher: Nature Publishing Group.

[54] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training Compute-Optimal Large Language Models. In *Advances in Neural Information Processing Systems*, volume 35. Curran Associates, Inc., 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/c1e2faff6f588870935f114ebe04a3e5-Abstract-Conference.html`.

[55] Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, Soroosh Mariooryad, Yifan Ding, Xinyang Geng, Fred Alcober, Roy Frostig, Mark Omernick, Lexi Walker, Cosmin Paduraru, Christina Sorokin, Andrea Tacchetti, Colin Gaffney, Samira Daruki, Olcan Sercinoglu, Zach Gleicher, Juliette Love, Paul Voigtlaender, Rohan Jain, Gabriela Surita, Kareem Mohamed, Rory Blevins, Junwhan Ahn, Tao Zhu, Kornraphop Kawintiranon, Orhan Firat, Yiming Gu, Yujing Zhang, Matthew Rahtz, Manaal Faruqui, Natalie Clay, Justin Gilmer, J. D. Co-Reyes, Ivo Penchev, Rui Zhu, Nobuyuki Morioka, Kevin Hui, Krishna Haridasan, Victor Campos, Mahdis Mahdieh, Mandy Guo, Samer Hassan, Kevin Kilgour, Arpi Vezer, Heng-Tze Cheng, Raoul de Liedekerke, Siddharth Goyal, Paul Barham, D. J. Strouse, Seb Noury, Jonas Adler, Mukund Sundararajan, Sharad Vikram, Dmitry Lepikhin, Michela Paganini, Xavier Garcia, Fan Yang, Dasha Valter, Maja Trebacz, Kiran Vodrahalli, Chulayuth Asawaroengchai, Roman Ring, Norbert Kalb, Livio Baldini Soares, Siddhartha Brahma, David Steiner, Tianhe Yu, Fabian Mentzer, Antoine He, Lucas Gonzalez, Bibo Xu, Raphael Lopez Kaufman, Laurent El Shafey, Junhyuk Oh, Tom Hennigan, George van den Driessche, Seth Odoom, Mario Lucic, Becca Roelofs, Sid Lall, Amit Marathe, Betty Chan, Santiago Ontanon, Luheng He, Denis Teplyashin, Jonathan Lai, Phil Crone, Bogdan Damoc, Lewis Ho, Sebastian Riedel, Karel Lenc, Chih-Kuan Yeh, Aakanksha Chowdhery, Yang Xu, Mehran Kazemi, Ehsan Amid, Anastasia Petrushkina, Kevin Swersky, Ali Khodaei, Gowoon Chen, Chris Larkin, Mario Pinto, Geng Yan, Adria Puigdomenech Badia, Piyush Patil, Steven Hansen, Dave Orr, Sebastien M. R. Arnold, Jordan Grimstad, Andrew Dai, Sholto Douglas, Rishika Sinha, Vikas Yadav, Xi Chen, Elena Gribovskaya, Jacob Austin, Jeffrey Zhao, Kaushal Patel, Paul Komarek, Sophia Austin, Sebastian Borgeaud, Linda Friso, Abhimanyu Goyal, Ben Caine, Kris Cao, Da-Woon Chung, Matthew Lamm, Gabe Barth-Maron, Thais Kagohara, Kate Olszewska, Mia Chen, Kaushik Shivakumar, Rishabh Agarwal, Harshal Godhia, Ravi Rajwar, Javier Snaider, Xerxes Dotiwalla, Yuan Liu, Aditya Barua, Victor Ungureanu, Yuan Zhang, Bat-Orgil Batsaikhan, Mateo Wirth, James Qin, Ivo Danihelka, Tulsee Doshi, Martin Chadwick, Jilin Chen, Sanil Jain, Quoc Le, Arjun Kar, Madhu Gurumurthy, Cheng Li, Ruoxin Sang, Fangyu Liu, Lampros Lamprou, Rich Munoz, Nathan Lintz, Harsh Mehta, Heidi Howard, Malcolm Reynolds, Lora Aroyo, Quan Wang, Lorenzo Blanco, Albin Cassirer, Jordan Griffith, Dipanjan Das, Stephan Lee, Jakub Sygnowski, Zach Fisher, James Besley, Richard Powell, Zafarali Ahmed, Dominik Paulus, David Reitter, Zalan Borsos, Rishabh Joshi, Aedan Pope, Steven

Hand, Vittorio Selo, Vihan Jain, Nikhil Sethi, Megha Goel, Takaki Makino, Rhys May, Zhen Yang, Johan Schalkwyk, Christina Butterfield, Anja Hauth, Alex Goldin, Will Hawkins, Evan Senter, Sergey Brin, Oliver Woodman, Marvin Ritter, Eric Noland, Minh Giang, Vijay Bolina, Lisa Lee, Tim Blyth, Ian Mackinnon, Machel Reid, Obaid Sarvana, David Silver, Alexander Chen, Lily Wang, Loren Maggiore, Oscar Chang, Nithya Attaluri, Gregory Thornton, Chung-Cheng Chiu, Oskar Bunyan, Nir Levine, Timothy Chung, Evgenii Eltyshev, Xiance Si, Timothy Lillicrap, Demetra Brady, Vaibhav Aggarwal, Boxi Wu, Yuanzhong Xu, Ross McIlroy, Kartikeya Badola, Paramjit Sandhu, Erica Moreira, Wojciech Stokowiec, Ross Hemsley, Dong Li, Alex Tudor, Pranav Shyam, Elahe Rahimtoroghi, Salem Haykal, Pablo Sprechmann, Xiang Zhou, Diana Mincu, Yujia Li, Ravi Addanki, Kalpesh Krishna, Xiao Wu, Alexandre Frechette, Matan Eyal, Allan Dafoe, Dave Lacey, Jay Whang, Thi Avrahami, Ye Zhang, Emanuel Taropa, Hanzhao Lin, Daniel Toyama, Eliza Rutherford, Motoki Sano, HyunJeong Choe, Alex Tomala, Chalence Safranek-Shrader, Nora Kassner, Mantas Pajarskas, Matt Harvey, Sean Sechrist, Meire Fortunato, Christina Lyu, Gamaleldin Elsayed, Chenkai Kuang, James Lottes, Eric Chu, Chao Jia, Chih-Wei Chen, Peter Humphreys, Kate Baumli, Connie Tao, Rajkumar Samuel, Cicero Nogueira dos Santos, Anders Andreassen, Nemanja Rakićević, Dominik Grewe, Aviral Kumar, Stephanie Winkler, Jonathan Caton, Andrew Brock, Sid Dalmia, Hannah Sheahan, Iain Barr, Yingjie Miao, Paul Natsev, Jacob Devlin, Feryal Behbahani, Flavien Prost, Yanhua Sun, Artiom Myaskovsky, Thanumalayan Sankaranarayana Pillai, Dan Hurt, Angeliki Lazaridou, Xi Xiong, Ce Zheng, Fabio Pardo, Xiaowei Li, Dan Horgan, Joe Stanton, Moran Ambar, Fei Xia, Alejandro Lince, Mingqiu Wang, Basil Mustafa, Albert Webson, Hyo Lee, Rohan Anil, Martin Wicke, Timothy Dozat, Abhishek Sinha, Enrique Piqueras, Elahe Dabir, Shyam Upadhyay, Anudhyan Boral, Lisa Anne Hendricks, Corey Fry, Josip Djolonga, Yi Su, Jake Walker, Jane Labanowski, Ronny Huang, Vedant Misra, Jeremy Chen, R. J. Skerry-Ryan, Avi Singh, Shruti Rijhwani, Dian Yu, Alex Castro-Ros, Beer Changpinyo, Romina Datta, Sumit Bagri, Arnar Mar Hrafnkelsson, Marcello Maggioni, Daniel Zheng, Yury Sulsky, Shaobo Hou, Tom Le Paine, Antoine Yang, Jason Riesa, Dominika Rogozinska, Dror Marcus, Dalia El Badawy, Qiao Zhang, Luyu Wang, Helen Miller, Jeremy Greer, Lars Lowe Sjos, Azade Nova, Heiga Zen, Rahma Chaabouni, Mihaela Rosca, Jiepu Jiang, Charlie Chen, Ruibo Liu, Tara Sainath, Maxim Krikun, Alex Polozov, Jean-Baptiste Lespiau, Josh Newlan, Zeyncep Cankara, Soo Kwak, Yunhan Xu, Phil Chen, Andy Coenen, Clemens Meyer, Katerina Tsihlas, Ada Ma, Juraj Gottweis, Jinwei Xing, Chenjie Gu, Jin Miao, Christian Frank, Zeynep Cankara, Sanjay Ganapathy, Ishita Dasgupta, Steph Hughes-Fitt, Heng Chen, David Reid, Keran Rong, Hongmin Fan, Joost van Amersfoort, Vincent Zhuang, Aaron Cohen, Shixiang Shane Gu, Anhad Mohananey, Anastasija Ilic, Taylor Tobin, John Wieting, Anna Bortsova, Phoebe Thacker, Emma Wang, Emily Caveness, Justin Chiu, Eren Sezener, Alex Kaskasoli, Steven Baker, Katie Millican, Mohamed Elhawaty, Kostas Aisopos, Carl Lebsack, Nathan Byrd, Hanjun Dai, Wenhao Jia, Matthew Wiethoff, Elnaz Davoodi, Albert Weston, Lakshman Yagati, Arun Ahuja, Isabel Gao, Golan Pundak, Susan Zhang, Michael Azzam, Khe Chai Sim, Sergi Caelles, James Keeling, Abhanshu Sharma, Andy Swing, YaGuang Li, Chenxi Liu, Carrie Grimes Bostock, Yamini Bansal, Zachary Nado, Ankesh Anand, Josh Lipschultz, Abhijit Karmarkar, Lev Proleev, Abe Ittycheriah, Soheil Hassas Yeganeh, George Polovets, Aleksandra Faust, Jiao Sun, Alban Rrustemi, Pen Li, Rakesh Shivanna, Jeremiah Liu, Chris Welty, Federico Lebron, Anirudh Baddepudi, Sebastian Krause, Emilio Parisotto, Radu Soricut, Zheng Xu, Dawn Bloxwich, Melvin Johnson, Behnam Neyshabur, Justin Mao-Jones, Renshen Wang, Vinay Ramasesh, Zaheer Abbas, Arthur Guez, Constant Segal, Duc Dung Nguyen, James Svensson, Le Hou, Sarah York, Kieran Milan, Sophie Bridgers, Wiktor Gworek, Marco Tagliasacchi, James Lee-Thorp, Michael Chang, Alexey Guseynov, Ale Jakse Hartman, Michael Kwong, Ruizhe Zhao, Sheleem Kashem, Elizabeth Cole, Antoine Miech, Richard Tanburn, Mary Phuong, Filip Pavetic, Sebastien Cevey, Ramona Comanescu, Richard Ives, Sherry Yang, Cosmo Du, Bo Li, Zizhao Zhang, Mariko Iinuma, Clara Huiyi Hu, Aurko Roy, Shaan Bijwadia, Zhenkai Zhu, Danilo Martins, Rachel Saputro, Anita Gergely, Steven Zheng, Dawei Jia, Ioannis Antonoglou, Adam Sadovsky, Shane Gu, Yingying Bi, Alek Andreev, Sina Samangooei, Mina Khan, Tomas Kocisky, Angelos Filos, Chintu Kumar, Colton Bishop, Adams Yu, Sarah Hodkinson, Sid Mittal, Premal Shah, Alexandre Moufarek, Yong Cheng, Adam Bloniarz, Jaehoon Lee, Pedram Pejman, Paul Michel, Stephen Spencer, Vladimir Feinberg, Xuehan Xiong, Nikolay Savinov, Charlotte Smith, Siamak Shakeri, Dustin Tran, Mary Chesus, Bernd Bohnet, George Tucker, Tamara von Glehn, Carrie Muir, Yiran Mao, Hideto Kazawa, Ambrose Slone, Kedar Soparkar, Disha Shrivastava, James Cobon-Kerr, Michael Sharman, Jay Pavagadhi, Carlos Araya, Karolis Misiunas, Nimesh Ghelani, Michael Laskin, David Barker, Qiujia Li, Anton Briukhov, Neil Houlsby, Mia Glaese, Balaji Lakshminarayanan, Nathan Schucher, Yunhao Tang, Eli Collins, Hyeontaek Lim, Fangxiaoyu Feng, Adria Recasens, Guangda Lai, Alberto Magni, Nicola De Cao, Aditya Siddhant, Zoe Ashwood, Jordi Orbay, Mostafa Dehghani, Jenny Brennan, Yifan He, Kelvin Xu, Yang Gao, Carl Saroufim, James Molloy, Xinyi Wu, Seb Arnold, Solomon Chang, Julian Schrittwieser, Elena Buchatskaya, Soroush Radpour, Martin Polacek, Skye Giordano, Ankur Bapna, Simon Tokumine, Vincent Hellendoorn,

Thibault Sottiaux, Sarah Cogan, Aliaksei Severyn, Mohammad Saleh, Shantanu Thakoor, Laurent Shefey, Siyuan Qiao, Meenu Gaba, Shuo-yiin Chang, Craig Swanson, Biao Zhang, Benjamin Lee, Paul Kishan Rubenstein, Gan Song, Tom Kwiatkowski, Anna Koop, Ajay Kannan, David Kao, Parker Schuh, Axel Stjerngren, Golnaz Ghiasi, Gena Gibson, Luke Vilnis, Ye Yuan, Felipe Tiengo Ferreira, Aishwarya Kamath, Ted Klimenko, Ken Franko, Kefan Xiao, Indro Bhattacharya, Miteyan Patel, Rui Wang, Alex Morris, Robin Strudel, Vivek Sharma, Peter Choy, Sayed Hadi Hashemi, Jessica Landon, Mara Finkelstein, Priya Jhakra, Justin Frye, Megan Barnes, Matthew Mauger, Dennis Daun, Khuslen Baatarsukh, Matthew Tung, Wael Farhan, Henryk Michalewski, Fabio Viola, Felix de Chaumont Quitry, Charline Le Lan, Tom Hudson, Qingze Wang, Felix Fischer, Ivy Zheng, Elspeth White, Anca Dragan, Jean-baptiste Alayrac, Eric Ni, Alexander Pritzel, Adam Iwanicki, Michael Isard, Anna Bulanova, Lukas Zilka, Ethan Dyer, Devendra Sachan, Srivatsan Srinivasan, Hannah Muckenhirn, Honglong Cai, Amol Mandhane, Mukarram Tariq, Jack W. Rae, Gary Wang, Kareem Ayoub, Nicholas FitzGerald, Yao Zhao, Woohyun Han, Chris Alberti, Dan Garrette, Kashyap Krishnakumar, Mai Gimenez, Anselm Levskaya, Daniel Sohn, Josip Matak, Inaki Iturrate, Michael B. Chang, Jackie Xiang, Yuan Cao, Nishant Ranka, Geoff Brown, Adrian Hutter, Vahab Mirrokni, Nanxin Chen, Kaisheng Yao, Zoltan Egyed, Francois Galilee, Tyler Liechty, Praveen Kallakuri, Evan Palmer, Sanjay Ghemawat, Jasmine Liu, David Tao, Chloe Thornton, Tim Green, Mimi Jasarevic, Sharon Lin, Victor Cotruta, Yi-Xuan Tan, Noah Fiedel, Hongkun Yu, Ed Chi, Alexander Neitz, Jens Heitkaemper, Anu Sinha, Denny Zhou, Yi Sun, Charbel Kaed, Brice Hulse, Swaroop Mishra, Maria Georgaki, Sneha Kudugunta, Clement Farabet, Izhak Shafran, Daniel Vlasic, Anton Tsitsulin, Rajagopal Ananthanarayanan, Alen Carin, Guolong Su, Pei Sun, Shashank V, Gabriel Carvajal, Josef Broder, Iulia Comsa, Alena Repina, William Wong, Warren Weilun Chen, Peter Hawkins, Egor Filonov, Lucia Loher, Christoph Hirnschall, Weiyi Wang, Jingchen Ye, Andrea Burns, Hardie Cate, Diana Gage Wright, Federico Piccinini, Lei Zhang, Chu-Cheng Lin, Ionel Gog, Yana Kulizhskaya, Ashwin Sreevatsa, Shuang Song, Luis C. Cobo, Anand Iyer, Chetan Tekur, Guillermo Garrido, Zhuyun Xiao, Rupert Kemp, Huaixiu Steven Zheng, Hui Li, Ananth Agarwal, Christel Ngani, Kati Goshvadi, Rebeca Santamaria-Fernandez, Wojciech Fica, Xinyun Chen, Chris Gorgolewski, Sean Sun, Roopal Garg, Xinyu Ye, S. M. Ali Eslami, Nan Hua, Jon Simon, Pratik Joshi, Yelin Kim, Ian Tenney, Sahitya Potluri, Lam Nguyen Thiet, Quan Yuan, Florian Luisier, Alexandra Chronopoulou, Salvatore Scellato, Praveen Srinivasan, Minmin Chen, Vinod Koverkathu, Valentin Dalibard, Yaming Xu, Brennan Saeta, Keith Anderson, Thibault Sellam, Nick Fernando, Fantine Huot, Junehyuk Jung, Mani Varadarajan, Michael Quinn, Amit Raul, Maigo Le, Ruslan Habalov, Jon Clark, Komal Jalan, Kalesha Bullard, Achintya Singhal, Thang Luong, Boyu Wang, Sujeevan Rajayogam, Julian Eisenschlos, Johnson Jia, Daniel Finchelstein, Alex Yakubovich, Daniel Balle, Michael Fink, Sameer Agarwal, Jing Li, Dj Dvijotham, Shalini Pal, Kai Kang, Jaclyn Konzelmann, Jennifer Beattie, Olivier Dousse, Diane Wu, Remi Crocker, Chen Elkind, Siddhartha Reddy Jonnalagadda, Jong Lee, Dan Holtmann-Rice, Krystal Kallarackal, Rosanne Liu, Denis Vnukov, Neera Vats, Luca Invernizzi, Mohsen Jafari, Huanjie Zhou, Lilly Taylor, Jennifer Prendki, Marcus Wu, Tom Eccles, Tianqi Liu, Kavya Kopparapu, Francoise Beaufays, Christof Angermueller, Andreea Marzoca, Shourya Sarcar, Hilal Dib, Jeff Stanway, Frank Perbet, Nejc Trdin, Rachel Sterneck, Andrey Khorlin, Dinghua Li, Xihui Wu, Sonam Goenka, David Madras, Sasha Goldshtein, Willi Gierke, Tong Zhou, Yaxin Liu, Yannie Liang, Anais White, Yunjie Li, Shreya Singh, Sanaz Bahargam, Mark Epstein, Sujoy Basu, Li Lao, Adnan Ozturel, Carl Crous, Alex Zhai, Han Lu, Zora Tung, Neeraj Gaur, Alanna Walton, Lucas Dixon, Ming Zhang, Amir Globerson, Grant Uy, Andrew Bolt, Olivia Wiles, Milad Nasr, Ilia Shumailov, Marco Selvi, Francesco Piccinno, Ricardo Aguilar, Sara McCarthy, Misha Khalman, Mrinal Shukla, Vlado Galic, John Carpenter, Kevin Villela, Haibin Zhang, Harry Richardson, James Martens, Matko Bosnjak, Shreyas Rammohan Belle, Jeff Seibert, Mahmoud Alnahlawi, Brian McWilliams, Sankalp Singh, Annie Louis, Wen Ding, Dan Popovici, Lenin Simicich, Laura Knight, Pulkit Mehta, Nishesh Gupta, Chongyang Shi, Saaber Fatehi, Jovana Mitrovic, Alex Grills, Joseph Pagadora, Tsendsuren Munkhdalai, Dessie Petrova, Danielle Eisenbud, Zhishuai Zhang, Damion Yates, Bhavishya Mittal, Nilesh Tripuraneni, Yannis Assael, Thomas Brovelli, Prateek Jain, Mihajlo Velimirovic, Canfer Akbulut, Jiaqi Mu, Wolfgang Macherey, Ravin Kumar, Jun Xu, Haroon Qureshi, Gheorghe Comanici, Jeremy Wiesner, Zhitao Gong, Anton Ruddock, Matthias Bauer, Nick Felt, Anirudh GP, Anurag Arnab, Dustin Zelle, Jonas Rothfuss, Bill Rosgen, Ashish Shenoy, Bryan Seybold, Xinjian Li, Jayaram Mudigonda, Goker Erdogan, Jiawei Xia, Jiri Simsa, Andrea Michi, Yi Yao, Christopher Yew, Steven Kan, Isaac Caswell, Carey Radebaugh, Andre Elisseeff, Pedro Valenzuela, Kay McKinney, Kim Paterson, Albert Cui, Eri Latorre-Chimoto, Solomon Kim, William Zeng, Ken Durden, Priya Ponnapalli, Tiberiu Sosea, Christopher A. Choquette-Choo, James Manyika, Brona Robenek, Harsha Vashisht, Sebastien Pereira, Hoi Lam, Marko Velic, Denese Owusu-Afriyie, Katherine Lee, Tolga Bolukbasi, Alicia Parrish, Shawn Lu, Jane Park, Balaji Venkatraman, Alice Talbert, Lambert Rosique, Yuchung Cheng, Andrei Sozanschi, Adam Paszke, Praveen Kumar, Jessica Austin, Lu Li, Khalid Salama, Bartek Perz, Wooyeol Kim, Nandita Dukkipati, Anthony Baryshnikov, Chris-

tos Kaplanis, XiangHai Sheng, Yuri Chervonyi, Caglar Unlu, Diego de Las Casas, Harry Askham, Kathryn Tunyasuvunakool, Felix Gimeno, Siim Poder, Chester Kwak, Matt Miecnikowski, Vahab Mirrokni, Alek Dimitriev, Aaron Parisi, Dangyi Liu, Tomy Tsai, Toby Shevlane, Christina Kouridi, Drew Garmon, Adrian Goedeckemeyer, Adam R. Brown, Anitha Vijayakumar, Ali Elqursh, Sadegh Jazayeri, Jin Huang, Sara Mc Carthy, Jay Hoover, Lucy Kim, Sandeep Kumar, Wei Chen, Courtney Biles, Garrett Bingham, Evan Rosen, Lisa Wang, Qijun Tan, David Engel, Francesco Pongetti, Dario de Cesare, Dongseong Hwang, Lily Yu, Jennifer Pullman, Srini Narayanan, Kyle Levin, Siddharth Gopal, Megan Li, Asaf Aharoni, Trieu Trinh, Jessica Lo, Norman Casagrande, Roopali Vij, Loic Matthey, Bramandia Ramadhana, Austin Matthews, C. J. Carey, Matthew Johnson, Kremena Goranova, Rohin Shah, Shereen Ashraf, Kingshuk Dasgupta, Rasmus Larsen, Yicheng Wang, Manish Reddy Vuyyuru, Chong Jiang, Joana Ijazi, Kazuki Osawa, Celine Smith, Ramya Sree Boppana, Taylan Bilal, Yuma Koizumi, Ying Xu, Yasemin Altun, Nir Shabat, Ben Bariach, Alex Korchemniy, Kiam Choo, Olaf Ronneberger, Chimezie Iwuanyanwu, Shubin Zhao, David Soergel, Cho-Jui Hsieh, Irene Cai, Shariq Iqbal, Martin Sundermeyer, Zhe Chen, Elie Bursztein, Chaitanya Malaviya, Fadi Biadsy, Prakash Shroff, Inderjit Dhillon, Tejasi Latkar, Chris Dyer, Hannah Forbes, Massimo Nicosia, Vitaly Nikolaev, Somer Greene, Marin Georgiev, Pidong Wang, Nina Martin, Hanie Sedghi, John Zhang, Praseem Banzal, Doug Fritz, Vikram Rao, Xuezhi Wang, Jiageng Zhang, Viorica Patraucean, Dayou Du, Igor Mordatch, Ivan Jurin, Lewis Liu, Ayush Dubey, Abhi Mohan, Janek Nowakowski, Vlad-Doru Ion, Nan Wei, Reiko Tojo, Maria Abi Raad, Drew A. Hudson, Vaishakh Keshava, Shubham Agrawal, Kevin Ramirez, Zhichun Wu, Hoang Nguyen, Ji Liu, Madhavi Sewak, Bryce Petrini, DongHyun Choi, Ivan Philips, Ziyue Wang, Ioana Bica, Ankush Garg, Jarek Wilkiewicz, Priyanka Agrawal, Xiaowei Li, Danhao Guo, Emily Xue, Naseer Shaik, Andrew Leach, Sadh MNM Khan, Julia Wiesinger, Sammy Jerome, Abhishek Chakladar, Alek Wenjiao Wang, Tina Ornduff, Folake Abu, Alireza Ghaffarkhah, Marcus Wainwright, Mario Cortes, Frederick Liu, Joshua Maynez, Andreas Terzis, Pouya Samangouei, Riham Mansour, Tomasz Kepa, François-Xavier Aubet, Anton Algymr, Dan Banica, Agoston Weisz, Andras Orban, Alexandre Senges, Ewa Andrejczuk, Mark Geller, Niccolo Dal Santo, Valentin Anklin, Majd Al Merey, Martin Baeuml, Trevor Strohman, Junwen Bai, Slav Petrov, Yonghui Wu, Demis Hassabis, Koray Kavukcuoglu, Jeff Dean, and Oriol Vinyals. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context, December 2024. URL `http://arxiv.org/abs/2403.05530`. arXiv:2403.05530 [cs].

[56] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, December 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract-Conference.html`.

[57] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems*, 35:24824–24837, December 2022. URL `https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html`.

[58] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, Kamilė Lukosiute, Karina Nguyen, Newton Cheng, Nicholas Joseph, Nicholas Schiefer, Oliver Rausch, Robin Larson, Sam McCandlish, Sandipan Kundu, Saurav Kadavath, Shannon Yang, Thomas Henighan, Timothy Maxwell, Timothy Telleen-Lawton, Tristan Hume, Zac Hatfield-Dodds, Jared Kaplan, Jan Brauner, Samuel R. Bowman, and Ethan Perez. Measuring Faithfulness in Chain-of-Thought Reasoning, July 2023. URL `http://arxiv.org/abs/2307.13702`. arXiv:2307.13702 [cs].

[59] Yanda Chen, Joe Benton, Ansh Radhakrishnan, Jonathan Uesato, Carson Denison, John Schulman, Arushi Somani, Peter Hase, Misha Wagner, Fabien Roger, Vlad Mikulik, Samuel R. Bowman, Jan Leike, Jared Kaplan, and Ethan Perez. Reasoning Models Don't Always Say What They Think, May 2025. URL `http://arxiv.org/abs/2505.05410`. arXiv:2505.05410 [cs].

[60] Thomas Kwa, Ben West, Joel Becker, Amy Deng, Katharyn Garcia, Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx, Ryan Bloom, Thomas Broadley, Haoxing Du, Brian Goodrich, Nikola Jurkovic, Luke Harold Miles, Seraphina Nix, Tao Lin, Neev Parikh, David Rein, Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler, Elizabeth Barnes, and Lawrence Chan. Measuring AI Ability to Complete Long Tasks, March 2025. URL `http://arxiv.org/abs/2503.14499`. arXiv:2503.14499 [cs].

[61] METR. Measuring AI Ability to Complete Long Tasks. *METR Blog*, March 2025. URL `https://metr.org/blog/2025-03-19-measuring-ai-ability-to-complete-long-tasks/`.

[62] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling Laws for Neural Language Models, January 2020. URL `http://arxiv.org/abs/2001.08361`. arXiv:2001.08361 [cs].