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
jeremy_goldwasser@berkeley.edu

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 innovations in preference learning, increased scale, expanded context windows, chain-of-thought reasoning, and supporting infrastructure.

Some key words: large language models; generative pre-trained transformers; ordinary least squares; generalized linear models; neural networks; tutorial; overview; review

* 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 here. Since the start of this project, we have had to revise 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) (OpenAI, 2022). 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 (Kung et al., 2023; Abbas et al., 2024; Eriksen et al., 2024; Katz et al., 2024). 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) (OpenAI, 2025a; Meta, 2025; Google, 2025; Anthropic, 2025). 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* (Radford et al., 2018).

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 (Vaswani et al., 2017; Radford et al., 2018), 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 text-based chat interfaces, providing a *prompt* and receiving output text, a *completion* (HuggingFace, 2025; OpenAI, 2025b; Community, 2025). Though completions appear as a single block, they in fact represent multiple iterative model runs (Vaswani et al., 2017).

At a high level, GPTs are "text generation" or "completion generation" models: given a sequence of words, they predict what follows (HuggingFace, 2025; Gadesha, 2024; Li et al., 2024) (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 $\boldsymbol{X}$. Here, functions of the user prompt are used to create the $\boldsymbol{X}$ vector, and GPTs output a predicted probability distribution over possible next words (or sub-words, called tokens) (Radford et al., 2018). As an illustration, consider predicting the blank in: "Every week, the little girl give treats to a furry, friendly _____." The $\boldsymbol{X}$ 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 (Sanderson, 2024). 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 (Vaswani et al., 2017). For example, given the sequence $\mathbf{X}$ described above, the model may first predict the token *guinea*, then condition on $\mathbf{X}$ augmented with *guinea* to predict *pig*, and finally predict EOS. The resulting decoded sequence $\hat{Y} = (\{guinea\}, \{pig\})$ 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 (Vaswani et al., 2017). High-quality text generation requires representing word meaning, position, and context while remaining computationally tractable. The following sections describe how modern GPTs achieve this.

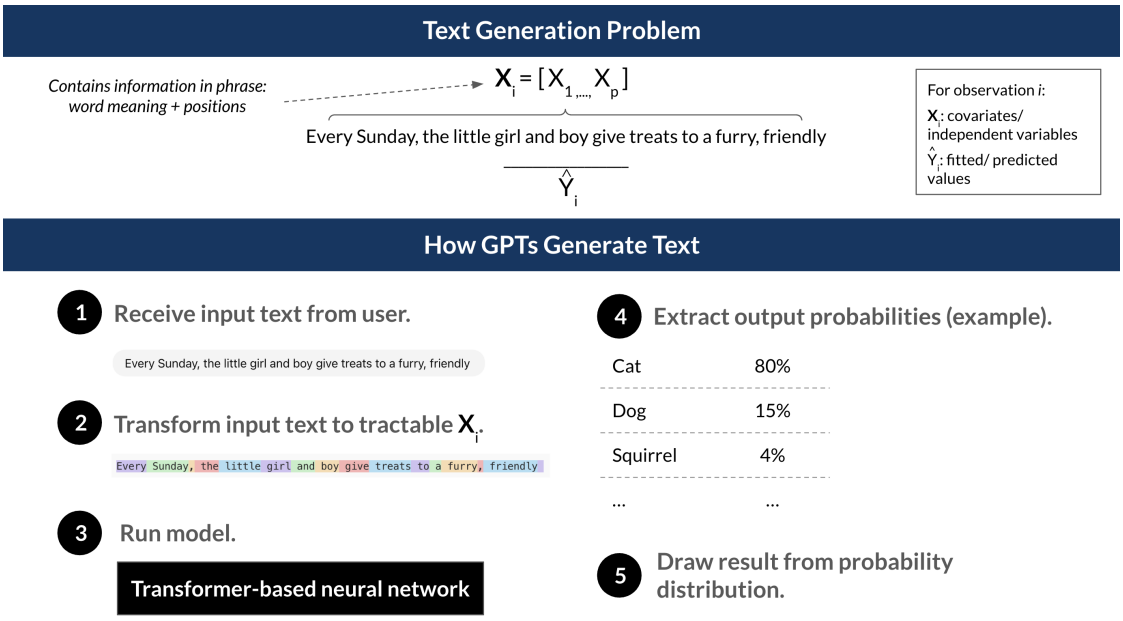

Fig. 1. The text generation problem and an overview of generative pre-trained transformer (GPT) structure.

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 ever a realistic option for natural language processing, but as a toy example 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, we find parameters that minimize:

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

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

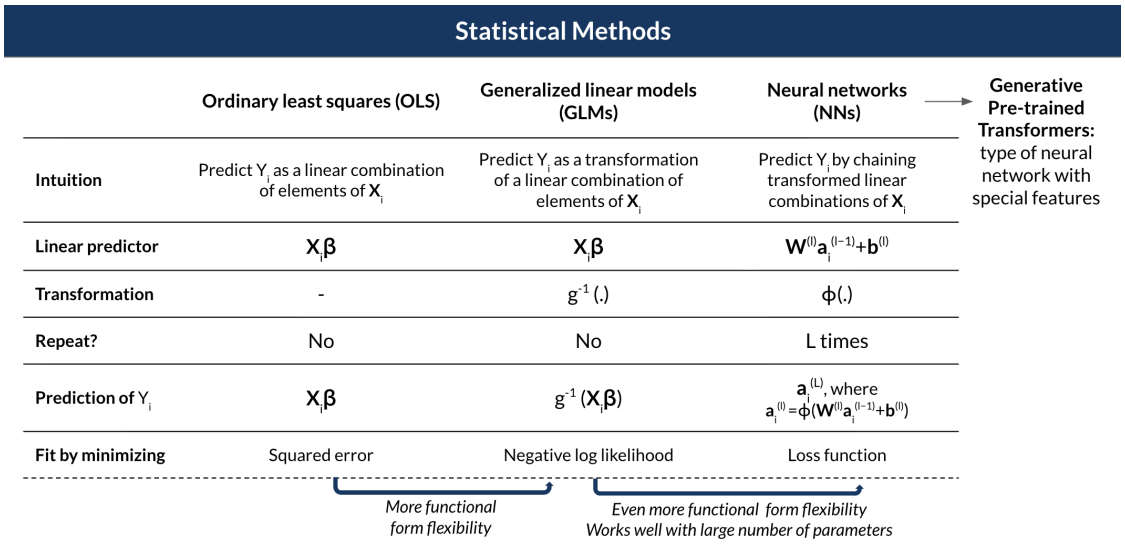

Fig. 2. An overview of statistical methods for prediction problems. Neural networks are shown abstractly; the final layer depends on the task: regression (identity link), binary classification (sigmoid), or multiclass classification (softmax).

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.

To better understand these limitations and motivate GPTs, we consider a stylized example of how OLS might be used for text generation. Letting V be the number of possible next words, suppose $\boldsymbol{X_i}$ denotes some simple representation of the preceding text. $\boldsymbol{X_i}$ 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 $\mathbf{X}_i$. These cannot be interpreted as probabilities, so we would need to normalize them, 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 $\hat{p}_{iw}$.

Unsurprisingly, this is not a viable approach. While OLS has many desirable properties, it is restricted to linear functions of inputs. When the true relationship between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is not linear, OLS will yield weak predictions, and language has complex structure that does not lend itself to linear representation. OLS estimates are also unbounded, meaning probability predictions could be negative. Directly modeling probabilities themselves would 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. 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 (McCullagh, 2018). This class of models, which includes OLS as well as logistic and Poisson regression, allow a partial relaxation of the functional form assumptions imposed by OLS (McCullagh, 2018). 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:

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

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 is a special case: with an identity link ($g(\mu) = \mu$) and Gaussian likelihood, 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 loss (or log-likelihood) until it reaches stable values (McCullagh, 2018). We then predict the mean of Y_i as $\hat{Y}_i = g^{-1}(\mathbf{X}_i\hat{\boldsymbol{\beta}})$.

For text completion with V possible next words, multinomial logistic regression provides a framework for predicting categorical outcomes. Specifying that $E[Y_{iw} \mid \boldsymbol{X}_i] = p_{iw}$, it uses a linear function $\boldsymbol{X}_i\boldsymbol{\beta}_{\boldsymbol{w}}$ to produce a score (logit) for each next word w, which is converted to a probability 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 $\boldsymbol{X}_i$ to the transformed mean $g(E[Y_i|\boldsymbol{X}_i])$, which does not hold in the text prediction setting and face similar limitations 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 (Ng & Ma, 2023; Nielsen, 2015). 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 $\boldsymbol{a}$, a single neuron $f(\boldsymbol{a})$ computes $\phi(\mathbf{aw} + b)$ (Nielsen, 2015). The transformation ϕ is known as the "activation function," and is analogous to the inverse link function g^{-1} in a GLM. The "weights" $\boldsymbol{w}$ and "bias" b are analogous to the GLM parameters $\boldsymbol{\beta}$. 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, $\mathbf{a} = \mathbf{x}$ (where i referring to a specific observation is dropped following computer science notation). 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 next layer ($\ell = 2$). 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.

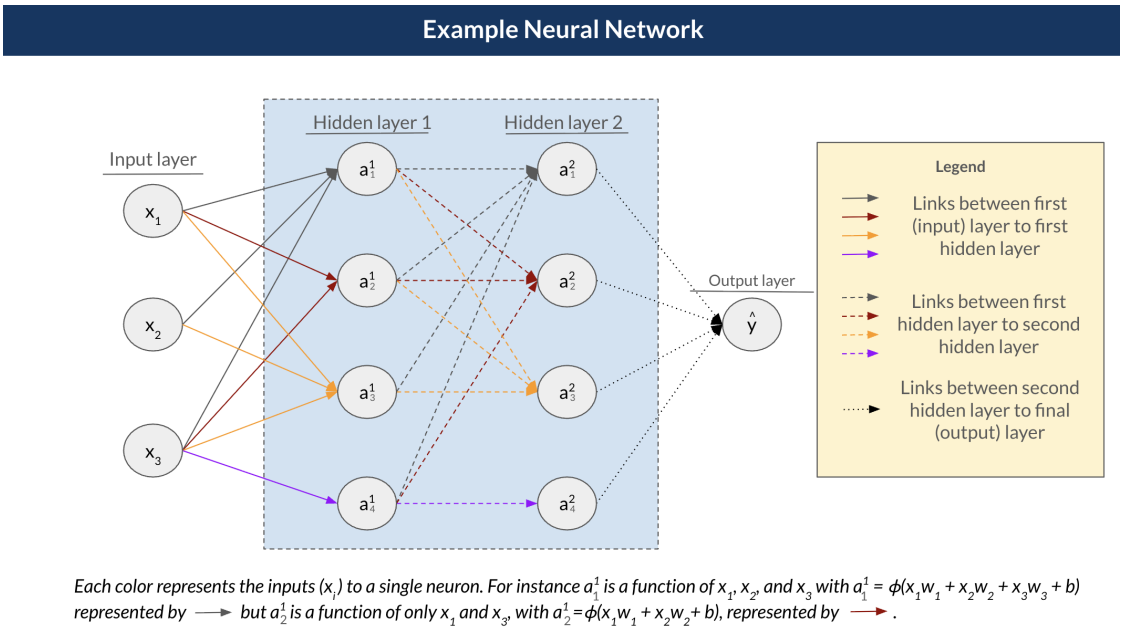

Fig. 3. An example neural network, demonstrating a single input layer, two hidden layers and an output layer. The arrows illustrate how variables are used from one layer to compute the next.

The value of a single node layer $\ell > 1$ can be written (Nielsen, 2015):

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

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, $b_j^{(\ell)}$ is the bias term for layer ℓ. Alternatively, in vector form, we may write:

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

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 (Nielsen, 2015).

In theory, two-layer neural networks with arbitrary hidden dimension can approximate nearly any functional form (Nielsen, 2015; Cybenko, 1989). However, to make learning from data tractable, networks are often provided additional structure and typically stack many hidden layers together (i.e., "deep learning") (Nielsen, 2015). 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 (Amidi & Amidi, 2024). 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 (Sutskever et al., 2011; Schmidt, 2019). 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 (Vaswani et al., 2017). 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" (OpenAI, 2025c). On average, tokens used by OpenAI as of November 2025 contained 4 characters and represented $\frac{3}{4}$th of a word (OpenAI, 2025d).

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 (Sennrich et al., 2016; Stanford Online, 2024). Tokenization tends to compress the word vocabulary into a smaller set of subwords (Radford et al., 2019; OpenAI, 2026). For instance, GPT-2 used 50,000 tokens and GPT-4o about 200,000, compared to roughly a million words in English (Merriam-Webster, 2025; Radford et al., 2019; OpenAI, 2026). 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 (Hugging Face, 2024).

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 = $\begin{bmatrix}1\ 0\ 0\ \dots\end{bmatrix}^T$, the = $\begin{bmatrix}0\ 1\ 0\ \dots\end{bmatrix}^T$. 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 interactions 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* (Mikolov et al., 2013a). 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 (Mikolov et al., 2013a; Sanderson, 2024). 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”) (Mikolov et al., 2013b).

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 (Brown et al., 2020).

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 (Vaswani et al., 2017) (Figure S2). 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 (Radford et al., 2018, 2019; Brown et al., 2020). Other models (e.g., LLAMA-2) incorporated position differently, integrating it later into the attention mechanism rather than at the input layer (Su et al., 2024; Touvron et al., 2023). 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. While the obvious approach would be to concatenate the two vectors, doing so would double the dimensionality and slow training. Instead, both signals can largely be preserved by adding the embeddings together, as random vectors in high-dimensional space are nearly orthogonal. Per this logic, the original Vaswani et al. (2017) paper proposed adding positional embeddings to the semantic embedding at the first layer of the network. 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 (Radford

et al., 2019; Dugas, 2023). Alternatively, some models add positional embeddings into various stages of the attention blocks (Shaw et al., 2018; Raffel et al., 2020; Su et al., 2021; Touvron et al., 2023).

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" (Vaswani et al., 2017). In this section, we describe the standard attention mechanism design before moving to other aspects of the transformer.

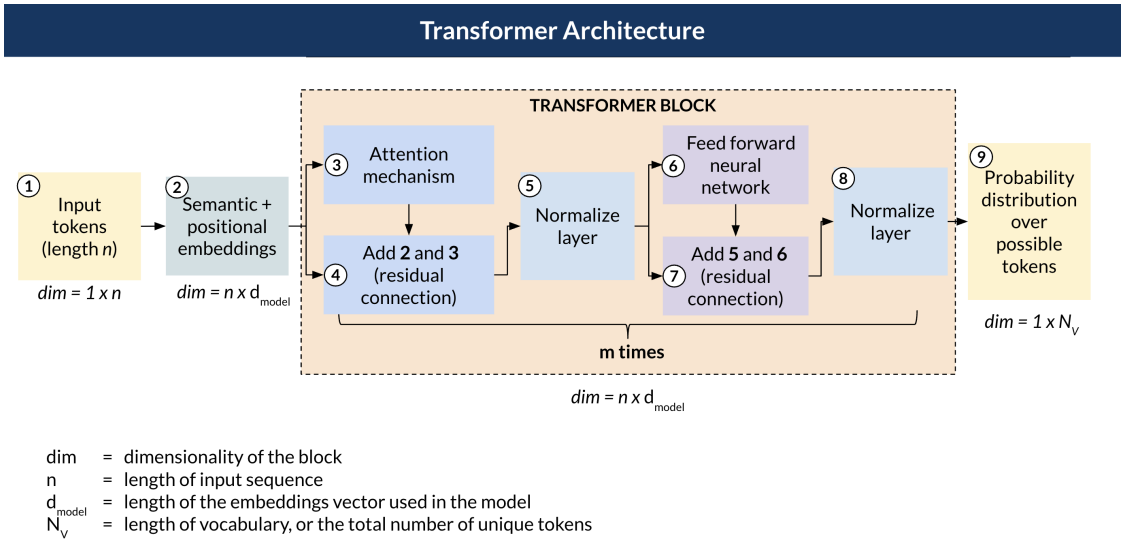

Fig. 4. A simplified transformer architecture that takes in a sequence of tokens as input and produces a probability distribution over all possible next tokens as output.

Consider the text completion task used throughout this paper: "Every Sunday, 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 "Sunday" 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 (Vaswani et al., 2017; Sanderson, 2024). 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 that use key-value pairs and queries (Pichka, 2025). 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:

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

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). We will discuss each step of this process in the following sections.

Here is how we can understand these components (Vaswani et al., 2017; Sanderson, 2024).

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 $d_k \times d_{model}$, 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.

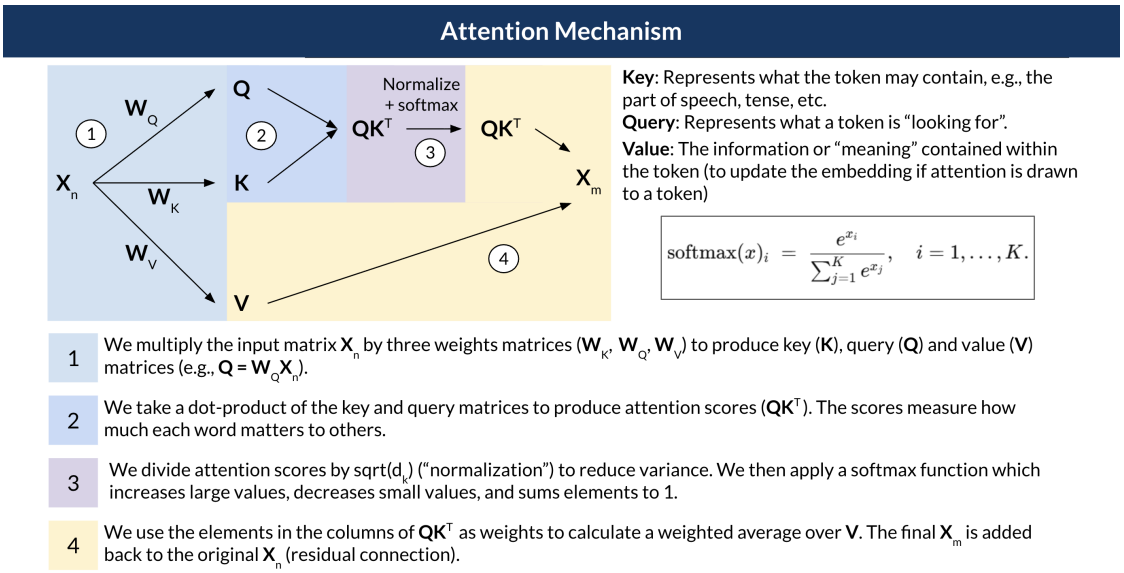

Fig. 5. The attention mechanism described in detail as a sequence of matrix multiplication and other operations.

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 $\mathbf{QK}^\top$, 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—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 $\left(\phi(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}\right)$. This exaggerates large values, suppresses small ones, and ensures each row sums to one. The resulting matrix $\mathbf{\Omega} = \text{softmax}(\mathbf{QK}^\top/\sqrt{d_k})$ contains the *attention weights*.

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 $m \times m$.

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

$$\mathbf{\Omega V} = \begin{bmatrix} w_{11}\mathbf{v}_{\text{Every}} + w_{12}\mathbf{v}_{\text{Sunday}} + w_{13}\mathbf{v}_{\text{the}} + \cdots \\ w_{21}\mathbf{v}_{\text{Every}} + w_{22}\mathbf{v}_{\text{Sunday}} + w_{23}\mathbf{v}_{\text{the}} + \cdots \\ \vdots \end{bmatrix} \tag{1}$$

Each row of the output is a weighted combination of all 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.

4.4. *Transformer architecture*

The output of each residual connection is also passed through a layer normalization step (Vaswani et al., 2017; Sanderson, 2024). 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 (Vaswani et al., 2017; Sanderson, 2024). 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 (Vaswani et al., 2017). Each head has its own query, key, and value matrices, producing a unique set of attention weights $\text{softmax}(\mathbf{QK}^\top/\sqrt{d_k})$. 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) (Kissane et al., 2024). The outputs of all heads are concatenated and then projected back to the model dimension d_{model} using a learned weight matrix $\mathbf{W}_O$. Because each head operates on a lower-dimensional subspace (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 (Brown et al., 2020). 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 $n \times m$. 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 (Dugas, 2023). 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 Sunday, 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 (Peeperkorn et al., 2024).

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 (Vaswani et al., 2017):

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

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

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

In many LLMs like GPT-3, $d_k = d_v = m/h$, where h is the number of heads (Radford et al., 2019; Brown et al., 2020). In this case, the total parameters across all h heads for the $m \times m/h$-dimensional Q, K, and V matrices is $3m^2$. An additional matrix $\mathbf{W}_o$ of dimension $m \times m$ combines the conjoined head outputs. The attention mechanism therefore contains $4m^2$ parameters per block. The feedforward network added further parameters. In GPT-3, two layers with an inner dimension of $4m$ contributed $8m^2$ parameters (Brown et al., 2020). Thus, each of the b blocks has $12m^2$ parameters, yielding a total count of roughly $12m^2b$.

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 $12m^2b = 174$ billion parameters. Token and position embeddings contributed to the final tally of 175 billion parameters (Brown et al., 2020). 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 (Grattafiori et al., 2024). 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 (Vaswani et al., 2017). 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 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 (Ouyang et al., 2022; Stanford Online, 2024).

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 (Radford et al., 2019; Brown

[1] As an alternative to GPTs, Mamba is a state-space model whose complexity is linear, rather than quadratic in the sequence length (Gu & Dao, 2023). Although promising, it generally underperforms GPTs; a detailed description is beyond the scope of this work.

et al., 2020). This scaling has only since continued, with contemporary LLMs pre-training on trillions of tokens (Chowdhery et al., 2022; Grattafiori et al., 2024; Deepseek, 2025). Llama 4 claims to use a staggeringly large dataset of 40 trillion tokens (Meta, 2025).

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 accordingly. 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 (Brown et al., 2020).

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 (Ouyang et al., 2022).

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. Hallucinations are thought to arise from fine-tuning, as models learn to produce assertive responses even when relevant information was absent from pre-training (Huang et al., 2025; Kalai et al., 2025).

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 (Ouyang et al., 2022). 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 (Christiano et al., 2017), RLHF was applied to language models at scale in InstructGPT (Ouyang et al., 2022). 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 (Rafailov et al., 2024). 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 (Guo et al., 2025).

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) (Radford et al., 2018, 2019). Subsequent models have continued this trend; the largest publicly documented models now exceed 400 billion parameters (Grattafiori et al., 2024). Researchers have also become better at optimizing the ratio of model parameters to training data (Hoffmann et al., 2022). 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 (Vaswani et al., 2017); GPT-2 extended this to 1,024 tokens (Radford et al., 2019), GPT-3 to 2,048 tokens (Brown et al., 2020), Llama-3 to 128,000 tokens (Grattafiori et al.,

2024) and Gemini reporting up to 10 million in experiments (Team et al., 2024). 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 (Dao et al., 2022) 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 (Wei et al., 2022). 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 (Guo et al., 2025). Interestingly, it performs well even when models are not actually using the reasoning they are stating within intermediate steps (Lanham et al., 2023; Chen et al., 2025).

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 (Kwa et al., 2025). 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 (METR, 2025). 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 (Kaplan et al., 2020), 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. For other applications, 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 GPTs may also be applied 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

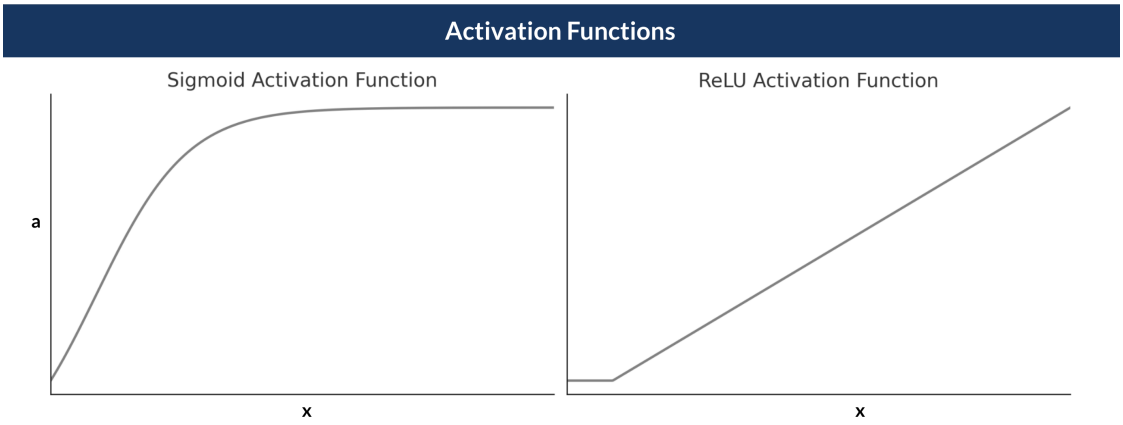

Fig. S1. Sigmoid and ReLU activation functions are both popular for neural networks. While a sigmoid activation function, which is differentiable (left), a ReLU function contains a ”kink”, below which the value is zero, and the function is linear above the kink.

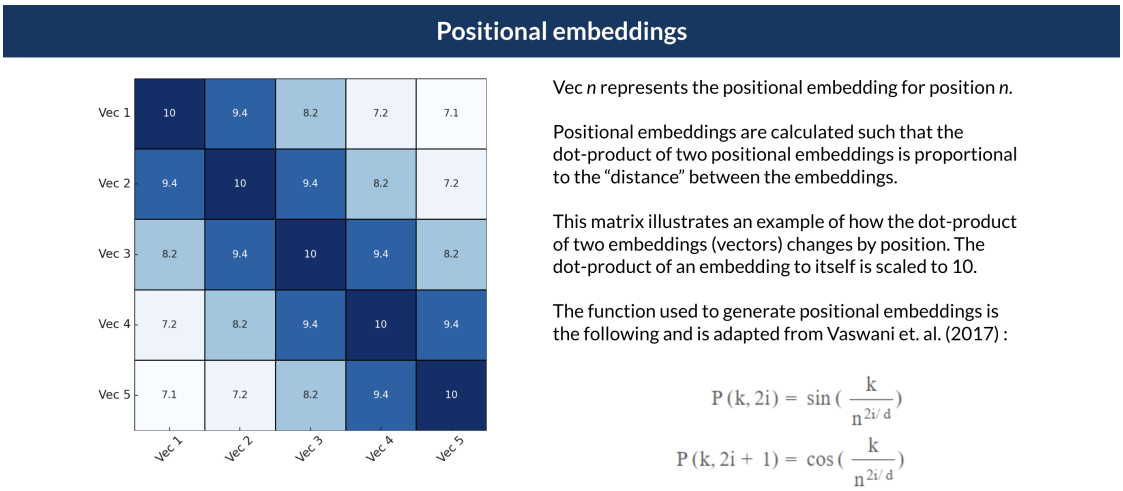

Fig. S2. Positional embeddings, calculated such that the dot product between two embeddings is proportional to the distance between the two positions.

References

Abbas, A., Rehman, M. S. & Rehman, S. S. (2024). Comparing the Performance of Popular Large Language Models on the National Board of Medical Examiners Sample Questions. *Cureus* Publisher: Springer Science and Business Media LLC.

Amidi, A. & Amidi, S. (2024). CS 230 - Convolutional Neural Networks Cheatsheet.

Anthropic (2025). Models overview.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I. & Amodei, D. (2020). Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc.

Chen, Y., Benton, J., Radhakrishnan, A., Uesato, J., Denison, C., Schulman, J., Somani, A., Hase, P., Wagner, M., Roger, F., Mikulik, V., Bowman, S. R., Leike, J., Kaplan, J. & Perez, E. (2025). Reasoning Models Don't Always Say What They Think. ArXiv:2505.05410 [cs].

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S. et al. (2022). Palm: Scaling language modeling with pathways. *arXiv preprint* **abs/2204.02311**.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S. & Amodei, D. (2017). Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan & R. Garnett, eds., vol. 30. Curran Associates, Inc.

Community (2025). Terminology evolution? "completion" vs "response". Section: Community.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **2**, 303–314.

Dao, T., Fu, D., Ermon, S., Rudra, A. & Ré, C. (2022). FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *Advances in Neural Information Processing Systems* **35**, 16344–16359.

Deepseek (2025). DeepSeek.

Dugas, D. (2023). The GPT-3 Architecture, on a Napkin.

Eriksen, A. V., Möller, S. & Ryg, J. (2024). Use of GPT-4 to Diagnose Complex Clinical Cases. *NEJM AI* **1**, AIp2300031. Publisher: Massachusetts Medical Society.

Gadesha, V. (2024). What is text generation?

Google (2025). Gemini models | Gemini API.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., Linde, J. v. d., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton, J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., Maaten, L. v. d., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., Oliveira, L. d., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speckbacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Goldschlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poulton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola,

B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Montgomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Caggioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satterfield, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mihailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z. & Ma, Z. (2024). The Llama 3 Herd of Models. ArXiv:2407.21783 [cs].

Gu, A. & Dao, T. (2023). Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752* .

Guo, D., Yang, D., Zhang, H., Song, J., Wang, P., Zhu, Q., Xu, R., Zhang, R., Ma, S., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Ruan, C., Dai, D., Chen, D., Ji, D. D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Xu, H., Ding, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Chen, J., Yuan, J., Tu, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., You, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Zhou, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z. & Zhang, Z. (2025). DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature* **645**, 633–638. Publisher: Nature Publishing Group.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., de Las Casas, D., Hendricks, L. A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., van den Driessche, G., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J. W., Vinyals, O. & Sifre, L. (2022). Training Compute-Optimal Large Language Models. In *Advances in Neural Information Processing Systems*, vol. 35. Curran Associates, Inc.

Huang, L., Yu, W., Ma, W., Zhong, W., Feng, Z., Wang, H., Chen, Q., Peng, W., Feng, X., Qin, B. & Liu, T. (2025). A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. *ACM Transactions on Information Systems* **43**, 1–55. ArXiv:2311.05232 [cs].

Hugging Face (2024). Byte-Pair Encoding tokenization - Hugging Face LLM Course.

HuggingFace (2025). What is Text Generation?

Kalai, A. T., Nachum, O., Vempala, S. S. & Zhang, E. (2025). Why Language Models Hallucinate. ArXiv:2509.04664 [cs].

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J. & Amodei, D. (2020). Scaling Laws for Neural Language Models. ArXiv:2001.08361 [cs].

Katz, U., Cohen, E., Shachar, E., Somer, J., Fink, A., Morse, E., Shreiber, B. & Wolf, I. (2024). GPT versus Resident Physicians — A Benchmark Based on Official Board Scores. *NEJM AI* **1**, AIdbp2300192. Publisher: Massachusetts Medical Society.

Kissane, C., Krzyzanowski, R., Bloom, J. I., Conmy, A. & Nanda, N. (2024). Interpreting Attention Layer Outputs with Sparse Autoencoders. ArXiv:2406.17759 [cs].

Kung, T. H., Cheatham, M., Medenilla, A., Sillos, C., De Leon, L., Elepaño, C., Madriaga, M., Aggabao, R., Diaz-Candido, G., Maningo, J. & Tseng, V. (2023). Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. *PLOS Digital Health* **2**, e0000198.

Kwa, T., West, B., Becker, J., Deng, A., Garcia, K., Hasin, M., Jawhar, S., Kinniment, M., Rush, N., Arx, S. V., Bloom, R., Broadley, T., Du, H., Goodrich, B., Jurkovic, N., Miles, L. H., Nix, S., Lin, T., Parikh, N., Rein, D., Sato, L. J. K., Wijk, H., Ziegler, D. M., Barnes, E. & Chan, L. (2025). Measuring AI Ability to Complete Long Tasks. ArXiv:2503.14499 [cs].

Lanham, T., Chen, A., Radhakrishnan, A., Steiner, B., Denison, C., Hernandez, D., Li, D., Durmus, E., Hubinger, E., Kernion, J., Lukoεsiūtė, K., Nguyen, K., Cheng, N., Joseph, N., Schiefer, N., Rausch, O., Larson, R., McCandlish, S., Kundu, S., Kadavath, S., Ya $of-ThoughtReasoning.ArXiv:2307.13702[cs]$.

Li, J., Tang, T., Zhao, W. X., Nie, J.-Y. & Wen, J.-R. (2024). Pre-Trained Language Models for Text Generation: A Survey. *ACM Comput. Surv.* **56**, 230:1–230:39.

McCullagh, P. (2018). *Generalized Linear Models*. Boca Raton: Routledge.

Merriam-Webster (2025). How many words are there in English?

Meta (2025). Llama by Meta.

METR (2025). Measuring AI Ability to Complete Long Tasks. *METR Blog* .

Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. ArXiv:1301.3781 [cs].

Mikolov, T., Yih, W.-t. & Zweig, G. (2013b). 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)*.

Ng, A. & Ma, T. (2023). CS229 Lecture Notes.

Nielsen, M. A. (2015). Neural Networks and Deep Learning Publisher: Determination Press.

OpenAI (2022). Introducing ChatGPT.

OpenAI (2025a). Models - OpenAI API.

OpenAI (2025b). Text generation and prompting.

OpenAI (2025c). Tokenizer.

OpenAI (2025d). What are tokens and how to count them?

OpenAI (2026). openai/tiktoken. Original-date: 2022-12-01T23:22:11Z.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J. & Lowe, R. (2022). Training language models to follow instructions with human feedback. ArXiv:2203.02155 [cs].

Peeperkorn, M., Kouwenhoven, T. & Brown, Dan and"; Jordanous, A. (2024). Is Temperature the Creativity Parameter of Large Language Models? In *Proceedings of the 15th International Conference on Computational Creativity (ICCC 2024)*.

Pichka, E. (2025). What is Query, Key, and Value (QKV) in the Transformer Architecture and Why Are They Used?

Radford, A., Narasimhan, K., Salimans, T. & Sutskever, I. (2018). Improving Language Understanding by Generative Pre-Training .

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D. & Sutskever, I. (2019). Language Models are Unsupervised Multitask Learners .

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D. & Finn, C. (2024). Direct preference optimization: Your language model is secretly a reward model.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W. & Liu, P. J. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research* **21**, 1–67.

Sanderson, G. (2024). Transformers, the tech behind LLMs | Deep Learning Chapter 5.

Schmidt, R. M. (2019). Recurrent Neural Networks (RNNs): A gentle Introduction and Overview. ArXiv:1912.05911 [cs].

Sennrich, R., Haddow, B. & Birch, A. (2016). Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the ACL (ACL 2016)*.

Shaw, P., Uszkoreit, J. & Vaswani, A. (2018). Self-attention with relative position representations. In *Proceedings of NAACL-HLT*.

Stanford Online (2024). Stanford CS229 I Machine Learning I Building Large Language Models (LLMs).

Su, J., Ahmed, M., Lu, Y., Pan, S., Bo, W. & Liu, Y. (2024). RoFormer: Enhanced transformer with Rotary Position Embedding. *Neurocomputing* **568**, 127063.

Su, J., Lu, Y., Pan, S., Wen, B. & Liu, Y. (2021). Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864* .

Sutskever, I., Martens, J. & Hinton, G. E. (2011). Generating Text with Recurrent Neural Networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML 2011)*.

Team, G., Georgiev, P., Lei, V. I., Burnell, R., Bai, L., Gulati, A., Tanzer, G., Vincent, D., Pan, Z., Wang, S., Mariooryad, S., Ding, Y., Geng, X., Alcober, F., Frostig, R., Omernick, M., Walker, L., Paduraru, C., Sorokin, C., Tacchetti, A., Gaffney, C., Daruki, S., Sercinoglu, O., Gleicher, Z., Love, J., Voigtlaender, P., Jain, R., Surita, G., Mohamed, K., Blevins, R., Ahn, J., Zhu, T., Kawintiranon, K., Firat, O., Gu, Y., Zhang, Y., Rahtz, M., Faruqui, M., Clay, N., Gilmer, J., Co-Reyes, J. D., Penchev, I., Zhu, R., Morioka, N., Hui, K., Haridasan, K., Campos, V., Mahdieh, M., Guo, M., Hassan, S., Kilgour, K., Vezer, A., Cheng, H.-T., Liedekerke, R. d., Goyal, S., Barham, P., Strouse, D. J., Noury, S., Adler, J., Sundararajan, M., Vikram, S., Lepikhin, D., Paganini, M., Garcia, X., Yang, F., Valter, D., Trebacz, M., Vodrahalli, K., Asawaroengchai, C., Ring, R., Kalb, N., Soares, L. B., Brahma, S., Steiner, D., Yu, T., Mentzer, F., He, A., Gonzalez, L., Xu, B., Kaufman, R. L., Shafey, L. E., Oh, J., Hennigan, T., Driessche, G. v. d., Odoom, S., Lucic, M., Roelofs, B., Lall, S., Marathe, A., Chan, B., Ontanon, S., He, L., Teplyashin, D., Lai, J., Crone, P., Damoc, B., Ho, L., Riedel, S., Lenc, K., Yeh, C.-K., Chowdhery, A., Xu, Y., Kazemi, M., Amid, E., Petrushkina, A., Swersky, K., Khodaei, A., Chen, G., Larkin, C., Pinto, M., Yan, G., Badia, A. P., Patil, P., Hansen, S., Orr, D., Arnold, S. M. R., Grimstad, J., Dai, A., Douglas, S., Sinha, R., Yadav, V., Chen, X., Gribovskaya, E., Austin, J., Zhao, J., Patel, K., Komarek, P., Austin, S., Borgeaud, S., Friso, L., Goyal, A., Caine, B., Cao, K., Chung, D.-W., Lamm, M., Barth-Maron, G., Kagohara, T., Olszewska, K., Chen, M., Shivakumar, K., Agarwal, R., Godhia, H., Rajwar, R., Snaider, J., Dotiwalla, X., Liu, Y., Barua, A., Ungureanu, V., Zhang, Y., Batsaikhan, B.-O., Wirth, M., Qin, J., Danihelka, I., Doshi, T., Chadwick, M., Chen, J., Jain, S., Le, Q., Kar, A., Gurumurthy, M., Li, C., Sang, R., Liu, F., Lamprou, L., Munoz, R., Lintz, N., Mehta, H., Howard, H., Reynolds, M., Aroyo, L., Wang, Q., Blanco, L., Cassirer, A., Griffith, J., Das, D., Lee, S., Sygnowski, J., Fisher, Z., Besley, J., Powell, R., Ahmed, Z., Paulus, D., Reitter, D., Borsos, Z., Joshi, R., Pope, A., Hand, S., Selo, V., Jain, V., Sethi, N., Goel, M., Makino, T., May, R., Yang, Z., Schalkwyk, J., Butterfield, C., Hauth, A., Goldin, A., Hawkins, W., Senter, E., Brin, S., Woodman, O., Ritter, M., Noland, E., Giang, M., Bolina, V., Lee, L., Blyth, T., Mackinnon, I., Reid, M., Sarvana, O., Silver, D., Chen, A., Wang, L., Maggiore, L., Chang, O., Attaluri, N., Thornton, G., Chiu, C.-C., Bunyan, O., Levine, N., Chung, T., Eltyshev, E., Si, X., Lillicrap, T., Brady, D., Aggarwal, V., Wu, B., Xu, Y., McIlroy, R., Badola, K., Sandhu, P., Moreira, E., Stokowiec, W., Hemsley, R., Li, D., Tudor, A., Shyam, P., Rahimtoroghi, E., Haykal, S., Sprechmann, P., Zhou, X., Mincu, D., Li, Y., Addanki, R., Krishna, K., Wu, X., Frechette, A., Eyal, M., Dafoe, A., Lacey, D., Whang, J., Avrahami, T., Zhang, Y., Taropa, E., Lin, H., Toyama, D., Rutherford, E., Sano, M., Choe, H., Tomala, A., Safranek-Shrader, C., Kassner, N., Pajarskas, M., Harvey, M., Sechrist, S., Fortunato, M., Lyu, C., Elsayed, G., Kuang, C., Lottes, J., Chu, E., Jia, C., Chen, C.-W., Humphreys, P., Baumli, K., Tao, C., Samuel, R., Santos, C. N. d., Andreassen, A., Rakićević, N., Grewe, D., Kumar, A., Winkler, S., Caton, J., Brock, A., Dalmia, S., Sheahan, H., Barr, I., Miao, Y., Natsev, P., Devlin, J., Behbahani, F., Prost, F., Sun, Y., Myaskovsky, A., Pillai, T. S., Hurt, D., Lazaridou, A., Xiong, X., Zheng, C., Pardo, F., Li, X., Horgan, D., Stanton, J., Ambar, M., Xia, F., Lince, A., Wang, M., Mustafa, B., Webson, A., Lee, H., Anil, R., Wicke, M., Dozat, T., Sinha, A., Piqueras, E., Dabir, E., Upadhyay, S., Boral, A., Hendricks, L. A., Fry, C., Djolonga, J., Su, Y., Walker, J., Labanowski, J., Huang, R., Misra, V., Chen, J., Skerry-Ryan, R. J., Singh, A., Rijhwani, S., Yu, D., Castro-Ros, A., Changpinyo, B., Datta, R., Bagri, S., Hrafnkelsson, A. M., Maggioni, M., Zheng, D., Sulsky, Y., Hou, S., Paine, T. L., Yang, A., Riesa, J., Rogozinska, D., Marcus, D., Badawy, D. E., Zhang, Q., Wang, L., Miller, H., Greer, J., Sjos, L. L., Nova, A., Zen, H., Chaabouni, R., Rosca, M., Jiang, J., Chen, C., Liu, R., Sainath, T., Krikun, M., Polozov, A., Lespiau, J.-B., Newlan, J., Cankara, Z., Kwak, S., Xu, Y., Chen, P., Coenen, A., Meyer, C., Tsihlas, K., Ma, A., Gottweis, J., Xing, J., Gu, C., Miao, J., Frank, C., Cankara, Z., Ganapathy, S., Dasgupta, I., Hughes-Fitt, S., Chen, H., Reid, D., Rong, K., Fan, H., Amersfoort, J. v., Zhuang, V., Cohen, A., Gu, S. S., Mohananey, A., Ilic, A., Tobin, T., Wieting, J., Bortsova, A., Thacker, P., Wang, E., Caveness, E., Chiu, J., Sezener, E., Kaskasoli, A., Baker, S., Millican, K., Elhawaty, M., Aisopos, K., Lebsack, C., Byrd, N., Dai, H., Jia, W., Wiethoff, M., Davoodi, E., Weston, A., Yagati, L., Ahuja, A., Gao, I., Pundak, G., Zhang, S., Azzam, M., Sim, K. C., Caelles, S., Keeling, J., Sharma, A., Swing, A., Li, Y., Liu, C., Bostock, C. G., Bansal, Y., Nado, Z., Anand, A., Lipschultz, J., Karmarkar, A., Proleev, L., Ittycheriah, A., Yeganeh, S. H., Polovets, G., Faust, A., Sun, J., Rrustemi, A., Li, P., Shivanna, R., Liu, J., Welty, C., Lebron, F., Baddepudi, A., Krause, S., Parisotto, E., Soricut, R., Xu, Z., Bloxwich, D., Johnson, M., Neyshabur, B., Mao-Jones, J., Wang, R., Ramasesh, V., Abbas, Z., Guez, A., Segal, C., Nguyen, D. D., Svensson, J., Hou, L., York, S., Milan, K., Bridgers, S., Gworek, W., Tagliasacchi, M., Lee-Thorp, J., Chang, M., Guseynov, A., Hartman, A. J., Kwong, M.,

Zhao, R., Kashem, S., Cole, E., Miech, A., Tanburn, R., Phuong, M., Pavetic, F., Cevey, S., Comanescu, R., Ives, R., Yang, S., Du, C., Li, B., Zhang, Z., Iinuma, M., Hu, C. H., Roy, A., Bijwadia, S., Zhu, Z., Martins, D., Saputro, R., Gergely, A., Zheng, S., Jia, D., Antonoglou, I., Sadovsky, A., Gu, S., Bi, Y., Andreev, A., Samangooei, S., Khan, M., Kocisky, T., Filos, A., Kumar, C., Bishop, C., Yu, A., Hodkinson, S., Mittal, S., Shah, P., Moufarek, A., Cheng, Y., Bloniarz, A., Lee, J., Pejman, P., Michel, P., Spencer, S., Feinberg, V., Xiong, X., Savinov, N., Smith, C., Shakeri, S., Tran, D., Chesus, M., Bohnet, B., Tucker, G., Glehn, T. v., Muir, C., Mao, Y., Kazawa, H., Slone, A., Soparkar, K., Shrivastava, D., Cobon-Kerr, J., Sharman, M., Pavagadhi, J., Araya, C., Misiunas, K., Ghelani, N., Laskin, M., Barker, D., Li, Q., Briukhov, A., Houlsby, N., Glaese, M., Lakshminarayanan, B., Schucher, N., Tang, Y., Collins, E., Lim, H., Feng, F., Recasens, A., Lai, G., Magni, A., Cao, N. D., Siddhant, A., Ashwood, Z., Orbay, J., Dehghani, M., Brennan, J., He, Y., Xu, K., Gao, Y., Saroufim, C., Molloy, J., Wu, X., Arnold, S., Chang, S., Schrittwieser, J., Buchatskaya, E., Radpour, S., Polacek, M., Giordano, S., Bapna, A., Tokumine, S., Hellendoorn, V., Sottiaux, T., Cogan, S., Severyn, A., Saleh, M., Thakoor, S., Shefey, L., Qiao, S., Gaba, M., Chang, S.-y., Swanson, C., Zhang, B., Lee, B., Rubenstein, P. K., Song, G., Kwiatkowski, T., Koop, A., Kannan, A., Kao, D., Schuh, P., Stjerngren, A., Ghiasi, G., Gibson, G., Vilnis, L., Yuan, Y., Ferreira, F. T., Kamath, A., Klimenko, T., Franko, K., Xiao, K., Bhattacharya, I., Patel, M., Wang, R., Morris, A., Strudel, R., Sharma, V., Choy, P., Hashemi, S. H., Landon, J., Finkelstein, M., Jhakra, P., Frye, J., Barnes, M., Mauger, M., Daun, D., Baatarsukh, K., Tung, M., Farhan, W., Michalewski, H., Viola, F., Quitry, F. d. C., Lan, C. L., Hudson, T., Wang, Q., Fischer, F., Zheng, I., White, E., Dragan, A., Alayrac, J.-b., Ni, E., Pritzel, A., Iwanicki, A., Isard, M., Bulanova, A., Zilka, L., Dyer, E., Sachan, D., Srinivasan, S., Muckenhirn, H., Cai, H., Mandhane, A., Tariq, M., Rae, J. W., Wang, G., Ayoub, K., FitzGerald, N., Zhao, Y., Han, W., Alberti, C., Garrette, D., Krishnakumar, K., Gimenez, M., Levskaya, A., Sohn, D., Matak, J., Iturrate, I., Chang, M. B., Xiang, J., Cao, Y., Ranka, N., Brown, G., Hutter, A., Mirrokni, V., Chen, N., Yao, K., Egyed, Z., Galilee, F., Liechty, T., Kallakuri, P., Palmer, E., Ghemawat, S., Liu, J., Tao, D., Thornton, C., Green, T., Jasarevic, M., Lin, S., Cotruta, V., Tan, Y.-X., Fiedel, N., Yu, H., Chi, E., Neitz, A., Heitkaemper, J., Sinha, A., Zhou, D., Sun, Y., Kaed, C., Hulse, B., Mishra, S., Georgaki, M., Kudugunta, S., Farabet, C., Shafran, I., Vlasic, D., Tsitsulin, A., Ananthanarayanan, R., Carin, A., Su, G., Sun, P., V, S., Carvajal, G., Broder, J., Comsa, I., Repina, A., Wong, W., Chen, W. W., Hawkins, P., Filonov, E., Loher, L., Hirnschall, C., Wang, W., Ye, J., Burns, A., Cate, H., Wright, D. G., Piccinini, F., Zhang, L., Lin, C.-C., Gog, I., Kulizhskaya, Y., Sreevatsa, A., Song, S., Cobo, L. C., Iyer, A., Tekur, C., Garrido, G., Xiao, Z., Kemp, R., Zheng, H. S., Li, H., Agarwal, A., Ngani, C., Goshvadi, K., Santamaria-Fernandez, R., Fica, W., Chen, X., Gorgolewski, C., Sun, S., Garg, R., Ye, X., Eslami, S. M. A., Hua, N., Simon, J., Joshi, P., Kim, Y., Tenney, I., Potluri, S., Thiet, L. N., Yuan, Q., Luisier, F., Chronopoulou, A., Scellato, S., Srinivasan, P., Chen, M., Koverkathu, V., Dalibard, V., Xu, Y., Saeta, B., Anderson, K., Sellam, T., Fernando, N., Huot, F., Jung, J., Varadarajan, M., Quinn, M., Raul, A., Le, M., Habalov, R., Clark, J., Jalan, K., Bullard, K., Singhal, A., Luong, T., Wang, B., Rajayogam, S., Eisenschlos, J., Jia, J., Finchelstein, D., Yakubovich, A., Balle, D., Fink, M., Agarwal, S., Li, J., Dvijotham, D., Pal, S., Kang, K., Konzelmann, J., Beattie, J., Dousse, O., Wu, D., Crocker, R., Elkind, C., Jonnalagadda, S. R., Lee, J., Holtmann-Rice, D., Kallarackal, K., Liu, R., Vnukov, D., Vats, N., Invernizzi, L., Jafari, M., Zhou, H., Taylor, L., Prendki, J., Wu, M., Eccles, T., Liu, T., Kopparapu, K., Beaufays, F., Angermueller, C., Marzoca, A., Sarcar, S., Dib, H., Stanway, J., Perbet, F., Trdin, N., Sterneck, R., Khorlin, A., Li, D., Wu, X., Goenka, S., Madras, D., Goldshtein, S., Gierke, W., Zhou, T., Liu, Y., Liang, Y., White, A., Li, Y., Singh, S., Bahargam, S., Epstein, M., Basu, S., Lao, L., Ozturel, A., Crous, C., Zhai, A., Lu, H., Tung, Z., Gaur, N., Walton, A., Dixon, L., Zhang, M., Globerson, A., Uy, G., Bolt, A., Wiles, O., Nasr, M., Shumailov, I., Selvi, M., Piccinno, F., Aguilar, R., McCarthy, S., Khalman, M., Shukla, M., Galic, V., Carpenter, J., Villela, K., Zhang, H., Richardson, H., Martens, J., Bosnjak, M., Belle, S. R., Seibert, J., Alnahlawi, M., McWilliams, B., Singh, S., Louis, A., Ding, W., Popovici, D., Simicich, L., Knight, L., Mehta, P., Gupta, N., Shi, C., Fatehi, S., Mitrovic, J., Grills, A., Pagadora, J., Munkhdalai, T., Petrova, D., Eisenbud, D., Zhang, Z., Yates, D., Mittal, B., Tripuraneni, N., Assael, Y., Brovelli, T., Jain, P., Velimirovic, M., Akbulut, C., Mu, J., Macherey, W., Kumar, R., Xu, J., Qureshi, H., Comanici, G., Wiesner, J., Gong, Z., Ruddock, A., Bauer, M., Felt, N., GP, A., Arnab, A., Zelle, D., Rothfuss, J., Rosgen, B., Shenoy, A., Seybold, B., Li, X., Mudigonda, J., Erdogan, G., Xia, J., Simsa, J., Michi, A., Yao, Y., Yew, C., Kan, S., Caswell, I., Radebaugh, C., Elisseeff, A., Valenzuela, P., McKinney, K., Paterson, K., Cui, A., Latorre-Chimoto, E., Kim, S., Zeng, W., Durden, K., Ponnapalli, P., Sosea, T., Choquette-Choo, C. A., Manyika, J., Robenek, B., Vashisht, H., Pereira, S., Lam, H., Velic, M., Owusu-Afriyie, D., Lee, K., Bolukbasi, T., Parrish, A., Lu, S., Park, J., Venkatraman, B., Talbert, A., Rosique, L., Cheng, Y., Sozanschi, A., Paszke, A., Kumar, P., Austin, J., Li, L., Salama, K., Perz, B., Kim, W., Dukkipati, N., Baryshnikov, A., Kaplanis, C., Sheng, X., Chervonyi, Y., Unlu, C., Casas, D. d. L., Askham, H., Tunyasuvunakool, K., Gimeno, F., Poder, S., Kwak, C., Miecnikowski, M., Mirrokni, V., Dimitriev, A., Parisi, A., Liu, D., Tsai, T., Shevlane, T., Kouridi, C., Garmon, D., Goedeckemeyer, A., Brown, A. R., Vijayakumar, A., Elqursh, A., Jazayeri, S., Huang, J., Carthy, S. M., Hoover, J., Kim, L., Kumar, S., Chen, W., Biles, C., Bingham, G., Rosen, E., Wang, L., Tan, Q., Engel, D., Pongetti, F., Cesare, D. d., Hwang, D., Yu, L., Pullman, J., Narayanan, S., Levin, K., Gopal, S., Li, M., Aharoni, A., Trinh, T., Lo, J., Casagrande,

N., Vij, R., Matthey, L., Ramadhana, B., Matthews, A., Carey, C. J., Johnson, M., Goranova, K., Shah, R., Ashraf, S., Dasgupta, K., Larsen, R., Wang, Y., Vuyyuru, M. R., Jiang, C., Ijazi, J., Osawa, K., Smith, C., Boppana, R. S., Bilal, T., Koizumi, Y., Xu, Y., Altun, Y., Shabat, N., Bariach, B., Korchemniy, A., Choo, K., Ronneberger, O., Iwuanyanwu, C., Zhao, S., Soergel, D., Hsieh, C.-J., Cai, I., Iqbal, S., Sundermeyer, M., Chen, Z., Bursztein, E., Malaviya, C., Biadsy, F., Shroff, P., Dhillon, I., Latkar, T., Dyer, C., Forbes, H., Nicosia, M., Nikolaev, V., Greene, S., Georgiev, M., Wang, P., Martin, N., Sedghi, H., Zhang, J., Banzal, P., Fritz, D., Rao, V., Wang, X., Zhang, J., Patraucean, V., Du, D., Mordatch, I., Jurin, I., Liu, L., Dubey, A., Mohan, A., Nowakowski, J., Ion, V.-D., Wei, N., Tojo, R., Raad, M. A., Hudson, D. A., Keshava, V., Agrawal, S., Ramirez, K., Wu, Z., Nguyen, H., Liu, J., Sewak, M., Petrini, B., Choi, D., Philips, I., Wang, Z., Bica, I., Garg, A., Wilkiewicz, J., Agrawal, P., Li, X., Guo, D., Xue, E., Shaik, N., Leach, A., Khan, S. M., Wiesinger, J., Jerome, S., Chakladar, A., Wang, A. W., Ornduff, T., Abu, F., Ghaffarkhah, A., Wainwright, M., Cortes, M., Liu, F., Maynez, J., Terzis, A., Samangouei, P., Mansour, R., Kepa, T., Aubet, F.-X., Algymr, A., Banica, D., Weisz, A., Orban, A., Senges, A., Andrejczuk, E., Geller, M., Santo, N. D., Anklin, V., Merey, M. A., Baeuml, M., Strohman, T., Bai, J., Petrov, S., Wu, Y., Hassabis, D., Kavukcuoglu, K., Dean, J. & Vinyals, O. (2024). Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. ArXiv:2403.05530 [cs].

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S. & Scialom, T. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. ArXiv:2307.09288 [cs].

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, u. & Polosukhin, I. (2017). Attention is All You Need. In *Advances in Neural Information Processing Systems*, vol. 30. Curran Associates, Inc.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q. V. & Zhou, D. (2022). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Advances in Neural Information Processing Systems* **35**, 24824–24837.