— 10 —
Generative models

The ImageNet era ends as attention shifts to powerful generative models trained on the internet. The new era also marks a turning point for machine learning benchmarks.

Source: The Emerging Science of Machine Learning Benchmarks. M. Hardt, 2025. URL: https://mlbenchmarks.org. Compiled on 2025-11-03.

At the peak of the ImageNet era, something remarkable happened outside the view of computer vision. Overshadowed by headline-making breakthroughs in image classification, some researchers worked hard to figure out what to do about language. Whereas deep convolutional neural networks had essentially solved static image classification problems, it was much less clear how to best apply deep learning to *natural language processing* (NLP).

At the time, state-of-the-art language modeling involved some form of recurrent neural network (RNN). RNNs consume one word at a time, updating an internal state representation at each step. There's no easy way to parallelize this computation, since each step depends on the result of the previous step. In contrast to the feed-forward networks of the ImageNet era, RNNs painfully underutilized the parallelism of modern GPUs (Graphics Processing Units), the engine of the deep learning revolution.

The long sequential computation also spelled trouble for gradient-based optimization. Gradients typically either grow or shrink exponentially in the number of sequential operations, a problem known as *vanishing* or *exploding gradients*. This made it generally hard for the model to remember what had happened at the beginning of a sentence by the time it reached the end. In fact, for machine translation it helped to also feed in the source sentence in reverse.[1] Fancier architectures, like the Long Short-Term Memory (LSTM) model, somewhat addressed the problem. But many issues remained. Recurrent models were hard to scale and finicky to work with. They required a huge amount of training time and a bag of tricks.

As a result, simpler models trained on more data often matched or outperformed RNNs in practice. An example is feed-forward neural networks trained on n-grams, counts of word pairs, triplets, and so forth. Less expressive than recurrent models, they made up for their disadvantage by training faster on more data.

Against this backdrop, researchers aimed to make feed-forward models more expressive without sacrificing training speed—to get the best of both worlds. The breakthrough came in 2017 with the *transformer* architecture, introduced in the paper *Attention is All You Need*.[2] It reconciled fast training with complex language modeling and marked the start of a revolution in sequence modeling.

The key idea behind the transformer architecture, counterintuitively, starts with what seems like a performance penalty. You have every word in the sequence interact with every other word. This gives you a quadratic number of interactions in the sequence length. You then assemble a new sequence

where each word is a weighted sum of these interactions. This operation is called *dot-product attention* and it's the core component of a transformer architecture. A transformer model repeats the attention block some number of times in a feed-forward fashion.

Despite the quadratic blow-up, there are two reasons why transformers scale well. Since each word is represented by a vector, computing all pairwise interactions corresponds to matrix multiplication. Matrix multiplication is what GPUs excel at. Modern GPUs perform hundreds of thousands of floating point operations in parallel during a single matrix multiplication. Second, the depth of the network is much smaller than the sequence length. This greatly reduces the amount of inherently sequential computation compared with an RNN.

Although the basic equation behind the transformer architecture is simple, a lot more goes into actually making it work. One of the inventors, Jakob Uszkoreit, explained in a 2024 interview how the "alchemy" was at least as important as the "conceptual stuff."[3] A working transformer implementation needed all sorts of tricks of the trade, from specific layer normalization and other black magic in the architecture to label smoothing and a custom learning rate schedule during training. In 2025, researcher Ellie Pavlick recalled:

> There had already been a feeling of the neural nets taking over, and so people were very skeptical and pushing back. Everyone's main takeaway was, "This is all just hacks."[4]

Yet it was precisely this patchwork of conceptual advances, hacks, heuristics, and hardware advances that turned the transformer from a clever idea into the foundation of a new era in language modeling. Another triumph of the *anything goes*.

10.1 Language models

Pick an English sentence you'd find on the internet:

> The quick brown fox jumps over the lazy dog.

To feed the sentence into a language model, you first have to encode it in some specific way. The choices you make here affect both training and evaluation.

For example, you can think of the sentence as a sequence of words and

represent each word as a vector indexing the position of the word in the dictionary. This way of doing it gives you a relatively short sequence, but each position can take on many values: The Oxford English Dictionary has about 600,000 words. Alternatively, you could think of the sentence as a sequence of characters. This results in a much longer sequence, but each position has fewer options: the ASCII text encoding uses only 7 bits per character.

The most common practice today lies somewhere in between. You split the sequence into chunks called *tokens*. The number of distinct tokens, called *vocabulary size*, is typically in the tens of thousands. If each token represents a byte pair of text, we have 65536 tokens. Allocating one token for each possible byte pair, though, isn't the most economical use of our tokens. The character sequence "have" is far more common in the English language than the sequence "qj". It doesn't make much sense to have to encode the common word "have" with two tokens, but reserve a special token for "qj". We'd prefer to have a single token for common strings like "have" and multiple tokens for rare strings like "qj". By choosing tokens cleverly, we can reduce the sequence length considerably. This is what *tokenization* does. It's the first step of the language modeling pipeline.

A *language model* is a probabilistic model that assigns a probability

$$p(w_1 w_2 \cdots w_d) \in [0, 1]$$

to a sequence of n tokens $w_1 w_2 \cdots w_d$. Mathematically, a language model therefore represents a distribution over sequences of varying lengths.

Using the chain rule from probability theory, we can express the probability of a sequence as a product of *next-token probabilities*:

$$p(w_1 w_2 \cdots w_d) = \prod_{i=1}^{n} p(w_i \mid w_1 w_2 \cdots w_{i-1})$$

The next-token probability $p(w_i \mid w_1 w_2 \cdots w_{i-1})$ is the *likelihood* of token w_i given *context* $w_1 w_2 \cdots w_{i-1}$.

A good language model should make these next-token probabilities large for plausible sequences and small for implausible sequences. The likelihood of *fox* given context "The quick brown" should be much larger than the likelihood of *meatball* given the same context. Keep in mind that words like *fox* and *meatball* may correspond to multiple tokens, depending on the tokenizer.

We can *decode* sequences of multiple tokens from a language model token by token: Start from some piece of context (a *prompt*), extract the most likely token given the context, add it to the context, and repeat. Instead of picking the most likely token at each step, we could sample one randomly from the next-token distribution. A common practice lies somewhere in between: Raise each next-token probability to some power $1/\tau$, where the scalar $\tau > 0$ is the *temperature* parameter, and sample from the modified distribution. Choosing a temperature less than 1 makes the distribution more peaked, favoring tokens with larger probabilities. Higher temperatures make the decoding more random, approaching the uniform distribution as the temperature rises. What works best depends on the application.

In an intuitive sense, a good language model should be able to solve a range of interesting problems. For example, we can try to get answers to knowledge questions. The name *Charles Dickens* should somehow come up when the model generates an answer to the prompt: "The author of *Bleak House* is." A language model may also perform translation tasks, if we prompt for things like "The French word for monkey is." By varying the prompt, we can try to solve all sorts of problems, from writing poems to solving math puzzles. Such *prompt engineering* is a major part of applying language models, and a headache for evaluation. There could always be some other prompt giving a better solution.

How useful a language model actually is hinges on what distribution over text it models. It's often not straightforward to describe what distribution a language model should represent. Echoing our conversation in Chapter 2, language is a diverse and dynamic cultural artifact that defies a static representation. In practice, a language model ends up representing whatever *corpus*, that is, collection of sequences, the model builder used for training.

Training a language model

The chain rule reveals how to turn language modeling into a prediction problem: Predict the next token given context. Think of the next token $y = w_i$ as the target label in a prediction problem given the features $x = w_1 w_2 \cdots w_{i-1}$ consisting of the preceding tokens. Each instance of the prediction problem corresponds to a token prefix x and a next-token target y. This turns unlabeled data into a sequence of supervised learning examples. Now unleash the *anything goes* of machine learning on language modeling.

A parametric language model p_θ specified by model parameters θ often has a maximum *context length* k. In this case, we only consider next-token

probabilities with a *context window* of size k, defined as

$$p_\theta(w_i \mid w_{i-k} w_{i-k+1} \cdots, w_{i-1}).$$

To avoid notational clutter, when $k \geq i$, we take this expression to mean conditioning on the first $i-1$ tokens in the sequence. The hope is that for large enough context length, conditioning on the first k tokens approximates conditioning on all preceding tokens. Define the probability that a model with limited context window assigns to a full sequence w as

$$p_\theta(w_1 w_2 \cdots w_d) = \prod_{i=1}^{n} p_\theta(w_i \mid w_{i-k} w_{i-k+1} \cdots, w_{i-1}).$$

Taking logarithms around the product on the right-hand side and multiplying by $-1/n$ gives us the *negative average log-likelihood* loss function on the sequence w:

$$\mathcal{L}(\theta; w) = -\frac{1}{n} \sum_{i=1}^{n} \log p_\theta(w_i \mid w_{i-k} w_{i-k+1} \cdots, w_{i-1}).$$

Here, the vector θ represents the model parameters, k is the context length, and $n = |w|$ is the sequence length. The loss corresponds to the negative average of the logarithms of the model's probabilities on the correct next tokens across the sequence. Given a distribution D over sequences, we analogously define the expected negative average log-likelihood as

$$\mathcal{L}(\theta) = - \mathop{\mathbb{E}}_{w \sim D} \left[\frac{1}{|w|} \sum_{i=1}^{|w|} \log p_\theta(w_i \mid w_{i-k} w_{i-k+1} \cdots w_{i-1}) \right].$$

This loss corresponds to picking a random sequence, picking a random position in the sequence and penalizing the model by the logarithm of the inverse next-token probability. The smaller the probability on the next token, the higher the loss.

Recalling the cross entropy loss from Chapter 2, we can see that minimizing this objective function is equivalent (up to scaling) to minimizing the cross entropy loss in a prediction problem. This has two implications. First, each gradient update on the negative log-likelihood nudges the model's probabilities toward the target token. Second, the unconstrained optimal solution equals the underlying distribution. This is a property of the cross entropy

loss. We should therefore expect a large enough well-trained language model to be a lot like the distribution it's trained on.

The main surprise in language modeling was just how far next-token prediction could take us.

Get the perplexity down!

Perplexity is a popular metric to evaluate the quality of a language model for next-token prediction. Perplexity is closely related to the training objective, the negative average log-likelihood. The early days of sequence learning were all about *getting the perplexity down* on toy datasets.

To evaluate the perplexity of a model on a dataset, we first concatenate and tokenize all sequences in the dataset into one long stream $w = w_1 \cdots w_D$ of D tokens. The number of tokens D could be millions or even billions of tokens depending on the dataset. We can define the perplexity of a model p on the whole sequence as

$$\text{PPL} = p(w)^{-1/D}.$$

Although succinct, it's more common to express perplexity as the exponential of the negative average log-likelihood:

$$\text{PPL} = \exp(\mathcal{L}(p; w)) = \exp\left(-\tfrac{1}{D}\textstyle\sum_{i=1}^{D} \log p(w_i \mid w_1 w_2 \cdots w_{i-1})\right).$$

By convention in language modeling, the logarithm and exponential have base e. But perplexity is base-invariant, so long as we use the same base for the logarithm and exponential.

To convince yourself that the two definitions of perplexity are equivalent, it's helpful to take an intermediate step and express perplexity as the geometric mean of the inverses of the next-token probabilities. Letting p_i denote the i-th next-token probability of the model p_θ on the sequence w,

$$\text{PPL} = \left(\prod_{i=1}^{D} \frac{1}{p_i}\right)^{1/D} = \left(\frac{1}{p(w)}\right)^{1/D}.$$

An inverse next-token probability corresponds to the effective number of choices a model has in predicting the next token. Explaining where the name perplexity comes from, this measures how surprised the model is by the text it encounters. Smaller perplexity is better.

Perplexity is never smaller than 1 but it could be infinite if the model assigns zero probability to a token that occurs. Most model evaluation metrics in machine learning are averages of a bounded quantity. Accuracy, for example, is the sample average of the error indicator. Perplexity is rather different. It's the exponential of an average of potentially unbounded quantities. As a result, perplexity doesn't enjoy the strong concentration of measure results we worked out in Chapter 3. Concentration of perplexity is weaker and depends on additional assumptions.

If somebody told you they got perplexity 7.28 on some dataset, you wouldn't know much from the number alone.

One reason is that perplexity is specific to a tokenizer. If we tokenize into a vocabulary of size V, the perplexity of the uniform distribution, i.e., random guessing, is $\exp(\log V) = V$. As a result, perplexity is generally smaller over smaller alphabets. Furthermore, a language model that successfully narrows down the next token to one of two equally likely options always has the same perplexity, namely, $\exp(-\log(1/2)) = 2$. Such a model is no better than random guessing when the dictionary is binary, but the guarantee is non-trivial when the dictionary is large. In other words, perplexity is insensitive to the hardness of the next-token prediction task.

Hoping to mitigate some of the quirks of perplexity, researchers often recommend *bits-per-byte* (BPB):

$$\mathrm{BPB} = -\frac{1}{B}\sum_{i=1}^{D} \log_2 p(w_i \mid w_1 w_2 \cdots w_{i-1}).$$

Here, B is the total number of bytes of the dataset (before tokenization) and the logarithm is base 2. Bits-per-byte relates to perplexity as

$$\mathrm{BPB} = \frac{D}{B} \cdot \frac{\ln \mathrm{PPL}}{\ln 2} = \frac{D}{B} \cdot \log_2 \mathrm{PPL}\,.$$

Bits-per-byte measures how good the language model is at compressing the dataset. Indeed, standard tools, such as *arithmetic coding*, can use the model's probabilities to compress the dataset down to $B \cdot \mathrm{BPB}$ bits up to a few bits of overhead.

A model that at each step assigns probability 1/2 to each of two tokens has D/B bits-per-byte. This number is lower (better) the smaller the number of tokens D is. A larger vocabulary will generally mean that we need fewer tokens to encode the dataset. This means the metric, unlike perplexity,

does recognize that next-token prediction is harder over larger vocabularies. However, if the tokenizer itself is already good at compression, we might misattribute this compression ability to the language model. Bits-per-byte confounds the quality of the tokenizer and the quality of the language model. For this reason, a bits-per-byte number isn't exactly easy to interpret either.

To make matters worse, both perplexity and bits-per-byte are sensitive to the ordering of data points in a dataset. Imagine a dataset that consists of a corpus of millions of tweets. If we arrange tweets by topic, the language model might generally have more useful context to work with, resulting in better numbers. Context switching drives up perplexity. If we randomly arrange the tweets, the previous tweet the model encountered might be misleading for whatever follows. Of course, absolute perplexity numbers can also differ sharply from one dataset to the other.

Both metrics are slow to evaluate on large datasets. Practitioners therefore often only evaluate next-token probabilities every so many tokens (called *stride*). These implementation details again influence what absolute number we end up with.

If perplexity is kind of a strange metric that's hard to interpret, why was it so popular? It remained popular because improvements in perplexity reliably tracked real improvements in language modeling more broadly. Getting the perplexity down on any large enough toy dataset usually indicated improvements in other downstream tasks.

Attention Is All You Need contains a table showing numerous different transformer architectures. For each architecture it lists the perplexity achieved by the model and its BLEU score, a metric for machine translation. Plotting perplexity and BLEU score in a scatterplot, we see a clear trend: The smaller the perplexity, the better the BLEU score.

The perplexity numbers in this plot are *word-piece* perplexities. Don't look at the absolute numbers. They don't directly compare to other perplexity numbers. It's the rank correlation that matters.

Getting the perplexity down was a signal model builders trusted.

The dataset didn't have to be fancy or realistic. Popularized in 2010 as a perplexity benchmark, the Penn Treebank dataset was one of the standard choices. Derived from a 1993 syntax benchmark, it featured text from Wall Street Journal articles, with fewer than 1M words in the training set and 82,000 words in the test set. Like CIFAR-10 for image classification, toy datasets like Penn Treebank were part of many language model evaluation

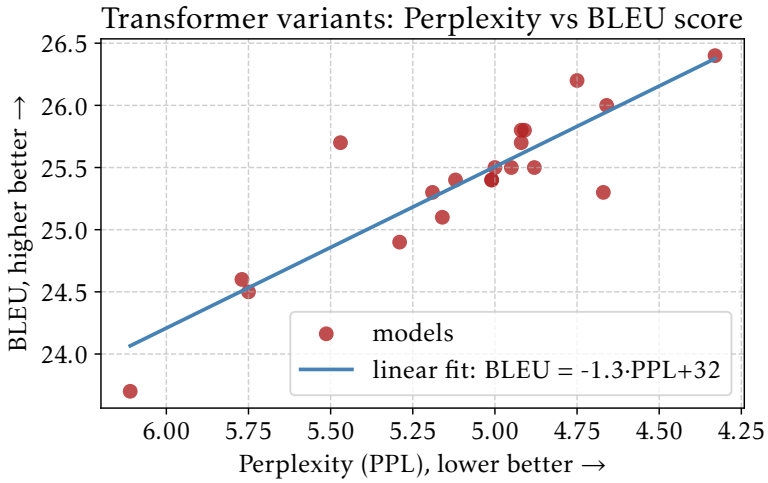

Figure 10.1: Scatterplot of transformer model variants from Attention Is All You Need: Lowest perplexity has highest BLEU score.

pipelines. Even in 2024, researchers continued to observe that perplexity tracked downstream benchmark performance.[5]

Researchers rekindled their faith in the external validity of the iron rule: Improvements in perplexity on any sufficiently large dataset implied—hopefully, at least—improvements elsewhere, too. The number was irrelevant. Beating the previous best was all that mattered.

And so the competition began.

10.2 Scaling

The original transformer architecture from 2017 sported 65 million parameters in its base model. It reached state-of-the-art on a machine translation benchmark after training for twelve hours on eight GPUs.

A year later, researchers at Google released BERT, an improved transformer model with 110 million parameters (340 million in its large version) trained on 3.3 billion words of text from *BookCorpus* and English Wikipedia.[6] BERT was trained on *masked language modeling*: Rather than predicting the next token, the objective is to predict randomly masked tokens given the whole sentence. In addition, BERT trained on *next sentence prediction*: a binary classification problem to distinguish the actual next sentence from a random sentence. This way of training resulted in good text embeddings broadly

useful for downstream language tasks. A valuable open-source tool, BERT created its own lineage featuring RoBERTa, DistilBERT, ALBERT, BigBird, and DeBERTa, among many others, as well as domain-specific variants such as BioBERT, SciBERT, LegalBERT, and CodeBERT.

Around the same time as BERT, OpenAI released GPT-1, a transformer model with 117 million parameters trained on 5 GB text from *BookCorpus*. Unlike BERT, GPT-1 trained on the basic next-token prediction objective. Like BERT, GPT-1 needed supervised fine-tuning on downstream tasks to achieve strong performance. At the time, however, researchers generally preferred BERT over GPT-1. GPT-1 seemed like an interesting but less powerful transformer model.

Nevertheless, OpenAI continued to scale up the GPT architecture to a 1.5 billion-parameter model, dubbed GPT-2, released in 2019.[7] To train GPT-2, OpenAI scraped about 8 million outgoing Reddit links with at least 3 *karma* points, resulting in roughly 40 GB of text from the internet.

With some *prompt-engineering*, GPT-2 generated plausible-looking pieces of text on its own without the need for additional fine-tuning. Evaluation without additional supervision earned the name *zero-shot evaluation*. This ability to solve tasks without fine-tuning *emerged* on its own with enough training data. The training objective remained next-token prediction with no explicit cues about any downstream test task. GPT-2's outputs were often quirky, sometimes on point, sometimes far off. Still, the model jolted researchers' assumptions, adding evidence that large language models were on to something.

What other abilities would emerge with additional scale?

In 2020, OpenAI released GPT-3, a model with 175 billion parameters trained to predict the next token on about 400 billion byte-pair encoded tokens of web-crawled text.[8] GPT-3 could tackle new tasks by learning from a few examples provided in the prompt, a technique called *few-shot prompting*. This ability—known as *in-context learning*—emerged with sufficient scale (training data, model size, and compute).

GPT-4 arrived three years after GPT-3, rumored to have 1.8 trillion parameters in a mixture of experts architecture.[9] If this number is fact, the relative increase in model size from GPT-3 to GPT-4 is more modest than the step from GPT-2 to GPT-3. Increasingly, companies invested not only in scaling the model but also in finding clever ways to fine-tune and prompt the *base* model.

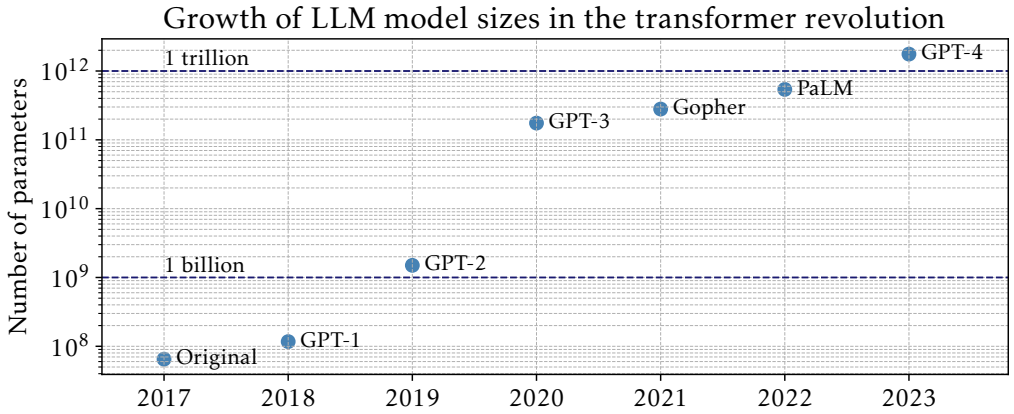

Figure 10.2: Model size growth in the transformer revolution. GPT-4 model size according to an unconfirmed estimate.

But what exactly are the limits to scaling?

In the early transformer days, it looked like hardware might become the bottleneck to continued scaling. Scaling required unprecedented compute resources, typically thousands of GPUs running for months to train a single model. Training GPT-3 reportedly took about 3×10^{23} floating point operations (FLOPs). Those consumed more than one million kilowatt-hours of electricity[10], roughly like the energy your body would use in 1200 years. And that's just one model—an old one at that—and only one training run. By training AI models and serving them on digital platforms, tech companies have been growing an enormous carbon footprint.[11–13]

For decades, computer scientists had relied on Moore's Law, the empirical trend that transistor counts on chips doubled roughly every two years. But this pace began to falter in the 2010s. Yet, what enabled the transformer revolution despite the end of Moore's Law was special-purpose hardware. GPU manufacturer NVIDIA kept finding ingenious ways to squeeze more floating point operations per second out of special-purpose designs aimed at deep learning broadly, and later transformers specifically. This maintained—and often bested—the gains of Moore's Law. In fact, the computing capacities of tech companies increased to the point that a different bottleneck appeared: Model builders might run out of high-quality data.[14]

Training data

Founded in 2007, Common Crawl is a non-profit organization—small in staff size—that's had an outsized impact on the development of artificial intelligence.[15] Common Crawl maintains petabytes of openly accessible web data. It didn't take long before researchers realized that Common Crawl could be a source of "dirt cheap" training data NLP applications.[16] Many training and benchmark datasets now come from filtering Common Crawl sources.

A single monthly Common Crawl snapshot contains hundreds of terabytes of data from billions of pages. Unfortunately, much of it has limited value for model training. Much of Common Crawl is boilerplate, menus, ads, login pages, duplicate content, junk sites, and spam.

Creating a good training set requires filtering much of the available data. Facebook's CCNet dataset, derived from Common Crawl, retained only 3.2 TB of compressed text starting from the February 2019 monthly Common Crawl snapshot.[17] Google's C4 dataset, created for the T5 model family, fished out around 750 GB of text from an initial 20 TB.[18] To train GPT-3, OpenAI filtered 570 GB of text from an initial 45 TB of Common Crawl data covering the years 2016 to 2019.[8]

Among the high-quality parts of the internet are the pages of Wikipedia, Reddit, Stack Exchange, GitHub, ArXiv, Project Gutenberg, and various news sites. Project Gutenberg, a collection of more than 75,000 free books online, adds up to tens of gigabytes of uncompressed text. The same is true for the English text portion of Wikipedia. These precious data sources have long been in language model training sets.

It's a safe bet that the latest large language models have trained on these high-quality parts of the accessible internet.[19] There's more data, however, in closed sources, behind paywalls and logins, in private chats and mail accounts, and across company intranets and government servers. AI companies are scrambling to find additional resources for training.[20,21] The data squeeze in turn had companies engage in more aggressive and questionable data acquisition practices.[13]

There are several ongoing lawsuits and debates about what data may be used for training large generative models.[22,23] The legal bases for these lawsuits include claims of copyright infringement, trademark dilution and violation, breach of licenses, in addition to privacy and data-protection claims. More broadly, content creators argue that companies take their

intellectual labor without attribution or permission to build commercial products. These products, like chatbots and assistants, largely exclude content creators from the revenue stream.[24–30]

Aside form its serious ethical dimension, the scarcity of accessible high-quality training data raises a practical question: How large a model can we train with a given amount of data? In other words, how much data is needed to best utilize an available compute budget? *Scaling laws* attempt to give an answer.

Scaling laws

Scaling laws refer to empirical relationships between test loss (or perplexity) and resource increases in terms of model size, dataset size, and compute. In 2020, researchers at OpenAI published a scaling law for training large language models:

> Language modeling performance improves smoothly as we increase the model size, dataset size, and amount of compute used for training. For optimal performance all three factors must be scaled up in tandem. Empirical performance has a power-law relationship with each individual factor when not bottlenecked by the other two.[31]

A scaling law predicts the test loss of a model trained at larger scale from smaller scale training runs. In addition, the authors claim that performance primarily depends on scale and less on architecture design:

> Performance depends strongly on scale, weakly on model shape: [...] Within reasonable limits, performance depends very weakly on other architectural hyperparameters such as depth vs. width.[31]

It therefore seemed to some as though architecture search was essentially over: It only remained to scale up the same kind of transformer architecture. Contrast this with the ImageNet era or the RNN days, where the competition was all about the model architectures. Now it looked like one architecture was good enough and scale was the one remaining dimension of competition. Consequently, the faith in scaling laws licensed massive investments in compute resources to scale up essentially the same kind of model in the years that followed. Looking back, transformers didn't end architecture search. There is much active research on alternative architectures. There are good reasons to believe that other architectures might be equally good or better with sufficient scale and optimization.

Building a scaling law. Let's dig into the details of a scaling law. A typical scaling law has four key parameters:

- Compute budget C (in FLOPs)
- Number N of model parameters
- Dataset size D (in number of tokens)
- Final cross-entropy training loss $L(N, D)$ of a stochastic gradient method making a full pass over the data

We can eliminate one parameter with the FLOPs heuristic

$$C = 6ND.$$

The equality derives from a useful oversimplification. In a transformer, the main computation involves matrix-vector multiplication between a weight matrix and a vector-encoded token. Therefore each model weight and token contribute one addition and one multiplication. Computing the model output across all tokens therefore costs $2ND$ FLOPs. This is the *forward pass* in a gradient based optimization method that computes the loss values. The *backward pass* computing the gradient is about twice as expensive as the forward pass, adding another $4ND$ operations. The two add up to $6ND$ FLOPs. For long sequences, the cost of computing the quadratic number of attention interactions dominates. But the simple heuristic is good enough for a scaling law for typical sequences.

Next we need to model the loss $L(N, D)$. This is where the power law comes in. Assume the loss has the functional form:

$$L(N, D) = E + AN^{-\alpha} + BD^{-\beta}.$$

This form is loosely inspired by a learning-theoretic heuristic. We can decompose the test loss into the sum of the Bayes optimal error (cf. Chapter 2), a *model approximation error*, and a *stochastic optimization error*. The Bayes optimal error is a constant E independent of N and D that depends only on the population. The model approximation error measures how well the model approximates the Bayes optimal predictor. It's plausible to assume that it's only a function of the model size N and follows a power law $AN^{-\alpha}$, for constants A and $\alpha > 0$. Similarly, the optimization error measures how close to optimal a single pass over the data with a stochastic gradient method gets us. It's not unreasonable to imagine that this is a only a function $BD^{-\beta}$ of the dataset size D.

With enough compute resources, we can now fit the scaling law empirically by collecting data from many training runs with varying parameters of N

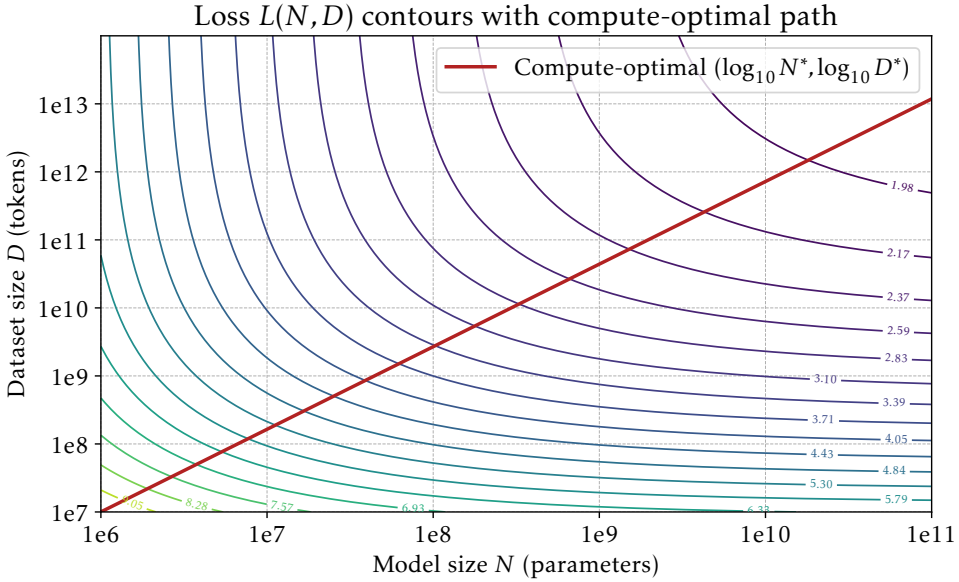

Figure 10.3: Contours of the estimated loss function in terms of the number of model parameters and number of training tokens. As you vary the compute budget, the parameters that minimize the loss for the given compute budget form line in the log-log plot.

and D. Following the first scaling law, researchers at Google proposed an improved scaling law, known as Chinchilla[32] scaling, that has the empirical power-law relationship with coefficients $E = 1.69$, $A = 406.4$, $B = 410.7$ and exponents $\alpha = 0.34$ and $\beta = 0.28$.

A scaling law has at least two applications. First, it lets us predict the performance of large models from smaller scale experiments. Second, the functional form of the scaling law, lets us answer the question we started out from: Given a compute budget, how much data do we need and how large a model should we train?

Compute-optimal scaling. To find the best settings of the parameters N and D for a given compute budget of C FLOPs, we can minimize the loss $L(N, D)$ subject to the constraint that $C = 6ND$.

The resulting *compute-optimal* values N_{opt} and D_{opt} have power laws

$$N_{\text{opt}} \sim C^a \quad \text{and} \quad D_{\text{opt}} \sim C^b,$$

respectively, with exponents $a = 0.46$ and $b = 0.54$. The fact that the two

exponents are close means that we should scale N and D proportionately with our compute budget. In fact, Chinchilla recommends $D \approx 20N$. Loosely speaking, a factor four increase in compute budget should come with a factor two increase in both model size and dataset size. If we use fewer model parameters than that, we risk *overtraining* with diminishing returns. If we use fewer data points than that, we risk *undertraining*, the opposite problem where additional data would give solid improvements.

Chinchilla scaling contradicted the earlier estimates by Kaplan et al. The latter suggested that dataset size D should scale as $D \sim N^{0.74}$ at optimality. This relationship means that ten times the parameters needs about 5.5 times the data, a more "compute hungry" scaling law where model size outgrows data. Chinchilla scaling is more "data hungry" by comparison. It suggests that data should grow on the same order as model size. A consequence of Chinchilla scaling is that models like GPT-3 were likely too large. For the same compute budget, a smaller model trained on more data would've been better.

In a world where we are compute-bound, Chinchilla's law lets us use more data before we run out of compute. If we're data-starved, the situation is the other way around. We wish that we needed less data to fully utilize our available compute budget. It wasn't clear from the beginning where we'd end up. Compute looked like it was going to be the bottleneck. But many years of rapid improvements in GPU hardware, software to utilize such hardware, and unprecedented scaling of compute facilities by tech companies have put us in a different world.

But this raises the question how much of a *law* is a scaling law anyway?

The limits of scaling laws

Researchers have fit power laws to large datasets for some time now. In the heyday of social network research, reports of power laws in social graphs were a recurring spectacle in prestigious journals. The degree distribution, reshare counts, community sizes, and post frequencies all looked like power laws. Retrospectively, however, Clauset, Shalizi, and Newman demonstrated that many other distribution families fit the data just as well as the power-law distribution.[33] The work tells a cautionary tale about the statistical application of power laws to real-world data. But this statistical complication isn't the only caveat.

Scaling laws aren't physical laws that nature handed to us. Scaling laws

reflect statistical regularities within a given engineering setup, not laws of nature. They are patterns that come from the particular ways that people at OpenAI, Meta, and Google build large transformer-based language models. This doesn't mean they aren't useful. Holding various engineering practices fixed, they let you extrapolate training loss under scaling of model parameters and dataset size.

In a sense, scaling laws are self-fulfilling prophecies. By following the prescriptions of a scaling law, researchers and engineers make the law more true. In fact, this is a well-known phenomenon with financial models such as the Black-Scholes model for pricing options. When introduced in 1973, the model was a theoretical abstraction whose assumptions did not have much empirical support. However, traders began to use the Black-Scholes model to price options, because it was simple, elegant, and eventually became standard. The sociologist Donald MacKenzie worked out how this broad use of the Black-Scholes model became self-fulfilling.[34] As traders embedded the model in financial practice, market behavior began to conform to the model's assumptions. That was at least until faith in those kinds of assumptions declined sharply with the collapse of a major hedge fund (LTCM) in 1998, and ultimately the *Great Recession* that followed the financial crisis in 2007.

Finally, perhaps the biggest caveat about scaling laws is that they only hold for test loss (or perplexity) but not necessarily for downstream task performance. This problem relates to debates about *emergent abilities*: Performance in some downstream tasks can pick up *unpredictably* at a certain large enough model scale.[35] Such emergent capabilities are in a sense counterexamples to scaling laws for downstream tasks. Emergence has been the source of much debate[36] among experts and anxiety about the potential risks of large language models.[29] The next chapter digs deeper into these problems and, in particular, the relationship between perplexity and downstream tasks.

10.3 Early NLP benchmarks

From the get-go, the transformer revolution had one stark difference with the deep learning revolution in computer vision. Unlike image classification in the 2010s, natural language understanding (NLU) had no single central benchmark. Whereas ImageNet was the direct target of the competition over the best deep convolutional models, language benchmarks appeared more downstream of the action. If anything, benchmarks struggled to cope with the rapid advances in transformer models. To better understand the

problem, it's worth taking a look at the earlier NLP benchmarks that were around in the 2010s when transformers came up. In the following chapters, we'll move on to more recent benchmarks.

Text comprehension. First released in 2016, the Stanford Question Answering Dataset (SQuAD) is a reading comprehension benchmark consisting of around 100,000 question-answer pairs.[37] The answer is contained in the provided context. The challenge is to extract the correct answer from the text snippet. Here's an example:

```
Context: The university is the major seat of the Congregation of Holy
    Cross (albeit not its official headquarters, which are in Rome).
    Its main seminary, Moreau Seminary, is located on the campus across
     St.  Joseph lake from the Main Building. Old College, the oldest
    building on campus and located near the shore of St. Mary lake,
    houses undergraduate seminarians. Retired priests and brothers
    reside in Fatima House (a former retreat center), Holy Cross House,
     as well as Columba Hall near the Grotto. The university through
    the Moreau Seminary has ties to theologian Frederick Buechner.
    While not Catholic, Buechner has praised writers from Notre Dame
    and Moreau Seminary created a Buechner Prize for Preaching.

Question: What is the primary seminary of the Congregation of the Holy
     Cross?

Answer: Moreau Seminary
```

Early transformer models reliably solved SQuAD with fine-tuning. SQuAD 2.0, released in 2018, made the benchmark harder by adding unanswerable questions after accuracy numbers on the original benchmark had saturated quickly in the BERT model era.[38]

Another popular natural language understanding benchmark, SNLI[39] (Stanford Natural Language Inference) targets natural language *entailment*: The goal is to classify the logical relationship of two consecutive sentences as *neutral*, *contradiction*, and *entailment*.

```
A person on a horse jumps over a broken down airplane.
A person is training his horse for a competition.
1 neutral

A person on a horse jumps over a broken down airplane.
A person is at a diner, ordering an omelette.
```

```
2 contradiction

A person on a horse jumps over a broken down airplane.
A person is outdoors, on a horse.
0 entailment
```

Just like SQuAD, the SNLI benchmark and its extension MultiNLI quickly became too easy for large transformer models. These benchmarks were high-quality efforts testing important linguistic competencies. Large language models solved them with ease, sparking debates about what it is that language benchmarks actually test for.[40]

A test less subject to abuse. In an ambitious proposal, the creators of the Winograd Schema Challenge (WSC) aimed at an alternative to Turing Test.[41] The authors start from a compelling analysis of the shortcomings of Turing's Imitation Game as a measure of intelligence. Among its issues is the fact that the test encourages "deception and trickery". The authors therefore articulate desiderata for a better benchmark "that is less subject to abuse". Specifically, a benchmark should have

> the subject responding to a broad range of English sentences; native English-speaking adults can pass it easily; it can be administered and graded without expert judges; no less than with the original Turing Test, when people pass the test, we would say they were thinking.[41]

Of the four criteria, the last is the most ambitious. A good test should require *thinking*. Inspired by textual entailment benchmarks, the authors propose a type of commonsense reasoning challenge. Consider the sentence:

```
The delivery truck zoomed by the school bus because it was going so
    slow.
```

The pronoun *it* is ambiguous in this sentence. It could refer to either the delivery truck or the school bus. But we can tell from *commonsense reasoning* that it probably refers to the school bus. Change the last word of the sentence to *fast* and you get a different resolution:

```
The delivery truck zoomed by the school bus because it was going so
    fast.
```

The pronoun now refers to the delivery truck. This pair of sentences is called a *Winograd schema*. WSC turned 136 such schemas into 273 test questions such as:

```
The city councilmen refused the demonstrators a permit
because they feared violence. Who feared violence?
A. The city councilmen
B. The demonstrators
```

The WSC creators assumed perfect human accuracy, but humans score a few accuracy points lower than that.[42]

Much thought, expertise, and ambition went into the design of WSC and the plan to replace the Turing Test with a more trustworthy benchmark. But WSC didn't escape the fate of all other benchmarks around the time. A fine-tuned RoBERTa model achieved well over 90% accuracy in 2019. The benchmark was soon considered solved.

WSC illustrates a robust lesson from benchmark design. Intended as measurements of latent constructs such as *intelligence*, benchmarks tend to fail, even if carefully devised. The creators assumed that solving WSC would necessarily involve some form of reasoning. But the success of models like BERT shook this assumption.

Inspired by WSC, researchers introduced *Winogrande*, a larger version of WSC filtered to eliminate statistical biases that may let a model shortcut the commonsense reasoning task.[43] Performance on Winogrande plateaued near human performance with GPT-4 and subsequent models.

Back to perplexity? Popularized by the release of GPT-2, LAMBADA (LAnguage Modeling Broadened to Account for Discourse Aspects) is a word prediction benchmark testing relatively long-range discourse comprehension.[44]

```
Context: He shook his head, took a step back and held his hands up as
   he tried to smile without losing a cigarette. "Yes you can," Julia
   said in a reassuring voice. "I've already focused on my friend. You
    just have to click the shutter, on top here."

Target sentence: He nodded sheepishly, through his cigarette away and
   took the _____.

Word: camera
```

By design of the benchmark, humans can easily guess the last word when seeing the whole passage, but struggle to do so when they only see the target sentence.

The OpenAI version of LAMBADA stuck around for some time as a lan-

guage modeling benchmark. Although the corpus is far from comprehensive, model builders trusted the signal that perplexity improvements on the benchmark provided. Indeed, Google evaluated its 2022 PaLM model with more than 500 billion parameters on LAMBADA.

Looking back, perplexity on relatively toy datasets was arguably the main benchmark for much of the transformer revolution. It was around for the discovery of the transformer architecture, it supported scaling laws, and it continued to support model comparisons in an increasingly chaotic benchmarking ecosystem.

Multi-task benchmarks. As different benchmarks for natural language understanding proliferated, it made sense to combine many of these into a single meta benchmark. And so the idea of a *multi-task benchmark* caught on.

The General Language Understanding Evaluation (GLUE) benchmark[45] grouped nine established natural language understanding benchmarks into one benchmark with a public leaderboard. Tasks include grammar checking, sentiment analysis, paraphrasing, sentence similarity, and natural language inference problems. All tasks are single sentence or sentence pair classification problems.

But aggregating many benchmarks into one didn't solve the issues with individual benchmarks. In a position paper from 2021, Bowman and Dahl reflect on the trouble with benchmarking in light of the transformer revolution:

> Performance on popular benchmarks is extremely high, but experts can easily find issues with high-scoring models. The GLUE benchmark, a compilation of NLU evaluation tasks, has seen performance on its leaderboard approach or exceed human performance on all nine of its tasks. The follow-up SuperGLUE benchmark project solicited dataset submissions from the NLP research community in 2019, but wound up needing to exclude the large majority of the submitted tasks from the leaderboard because the BERT model was already showing performance at or above that of a majority vote of human crowdworkers. Of the eight tasks for which BERT did poorly enough to leave clear headroom for further progress, all are now effectively saturated.[46]

Transformers shook the benchmarking enterprise and eroded trust in an institution that had delivered results for decades. As transformers models progressed, a benchmarking crisis loomed.

10.4 CLIP and a final look at ImageNet

The transformer revolution also came around to image classification. Rather than curating a labeled datasets, such as ImageNet, researchers found that they could train image representations on web crawled data in an unsupervised manner:

> [T]he simple pre-training task of predicting which caption goes with which image is an efficient and scalable way to learn SOTA image representations from scratch on a dataset of 400 million (image, text) pairs collected from the internet.[47]

The quote is from the abstract of the paper that introduced OpenAI's CLIP model at the beginning of 2021. CLIP trained on a simple unsupervised objective that easily scaled to web crawled data without costly human supervision. By consuming image-caption pairs, the model jointly learns an image representation and a text representation that talk to each other. Text that suits an image well, like a good caption, ends up close to the image in embedding space.

A piece of pseudocode from the original paper best illustrates the core idea.

```
# image_encoder - ResNet or Vision Transformer
# text_encoder - CBOW or Text Transformer
# I[n, h, w, c] - minibatch of aligned images
# T[n, l] - minibatch of aligned texts
# W_i[d_i, d_e] - learned proj of image to embed
# W_t[d_t, d_e] - learned proj of text to embed
# t - learned temperature parameter

# extract feature representations of each modality
I_f = image_encoder(I) #[n, d_i]
T_f = text_encoder(T) #[n, d_t]

# joint multimodal embedding [n, d_e]
I_e = l2_normalize(np.dot(I_f, W_i), axis=1)
T_e = l2_normalize(np.dot(T_f, W_t), axis=1)

# scaled pairwise cosine similarities [n, n]
logits = np.dot(I_e, T_e.T) * np.exp(t)

# symmetric loss function
```

```
labels = np.arange(n)
loss_i = cross_entropy_loss(logits, labels, axis=0)
loss_t = cross_entropy_loss(logits, labels, axis=1)
loss = (loss_i + loss_t)/2
```

CLIP was a bit like the GPT of image representations. Like GPT, CLIP solved no fixed task; the model applies to various tasks via zero-shot prompting. The model accepts both text and image inputs. To have it classify *cats* versus *dogs*, you give it the image and check whether the embedding of the text *a photo of a cat* is closer to the image embedding than the embedding of *a photo of a dog*.

What you need to train a model like CLIP are tons of image-caption pairs, which the internet abundantly supplies. The dataset OpenAI crawled and used for CLIP training was never available. However, a grassroots initiative called LAION created a massive dataset of image-caption pairs by filtering Common Crawl sources, culminating in nearly 6 billion image-caption pairs.[48] With the help of LAION, researchers successfully created an open-source replication of CLIP.[49]

Off the line?

The allure of CLIP compared to earlier ImageNet models is its apparent robustness to changing domains. In particular CLIP's image representations transfer more gracefully from one domain to another compared with its earlier competitors trained on ImageNet. Where those models' accuracies drop sharply when going from ImageNet to other domains, CLIP's accuracy is about the same.[47] In Chapter 7, we saw that in a typical scatterplot of in-domain and out-of-domain accuracies, models trained on ImageNet cluster on a line below the main diagonal. CLIP defied the line. It landed solidly above the line.

The creators of CLIP attribute its robustness to the zero-shot paradigm: You can't overfit to a dataset, if you zero-shot your model predictions. CLIP trained on image-caption pairs from the internet, favoring no particular domain. ImageNet models, on the other hand, trained on one specific dataset and therefore suffered elsewhere.

But if CLIP trained on the internet, isn't it possible that it has already seen many of the test cases before? After all, various benchmark datasets also originate from the same kind of web crawls. Did CLIP perhaps train on the test set, thus committing the cardinal sin of machine learning benchmarks?

Following this suspicion, researchers asked: "Does CLIP's Generalization Performance Mainly Stem From High Train-Test Similarity?"[50] Betteridge's law of headlines commands that the answer to any question in a headline is *no*. Indeed, the study finds that the situation is not so simple:

> "[. . .] it is questionable how meaningful CLIP's high zero-shot performance is as it seems likely that web-scale datasets like LAION simply contain many samples that are similar to common OOD benchmarks originally designed for ImageNet. To test this hypothesis, we retrain CLIP on pruned LAION splits that replicate ImageNet's train-test similarity with respect to common OOD benchmarks. While we observe a performance drop on some benchmarks, surprisingly, CLIP's overall performance remains high."

There is no smoking gun evidence that CLIP's transfer performance is due to training on the test set. The problem is more subtle. By training on the internet broadly, CLIP already encountered the domains that it was later evaluated on. Whereas a model trained on ImageNet had never seen cartoon versions of various objects, for example, CLIP must have seen lots of them during training:

> Indeed, Mayilvahanan et al. (2023) revealed that CLIP's training data contains exact or near duplicates of samples of many OOD datasets. Yet, they showed that CLIP still generalizes well when this sample contamination is corrected. However, their analysis failed to account for domain contamination. In contrast to sample contamination, domain contamination does not care about duplicates of specific data points but instead checks whether crucial aspects of a test domain are included in the training domain, e.g., by including images with different content but similar style to test samples.[51]

Training on the internet poses a challenge to benchmarking. Most commercial model providers give little to no insight into training datasets. As a result, the evaluator doesn't know what data a model has already encountered. There is no way to rule out that a model hasn't already seen the kind of data that's in any particular benchmark. Comparisons of models trained on different training data are never apples-to-apples comparisons. One model may have trained on data more similar to the test task than the other model. This problem became only more severe when evaluating large language models. The next chapter continues this thread.

Notes

Language modeling in its current form of next-word prediction dates back to Shannon's 1950s work.[52] Shannon estimated the entropy of the English language to be between 0.6 and 1.3 bits per character. The estimate comes from how well humans can predict the next word in English sentences. He compared this to how well *n*-gram models do.

The Computer History Museum conducted an interview with Jakob Uszkoreit in 2024 that conveys a wealth of intuition and background about the development of the transformer architecture.[3] Phuong and Hutter give a helpful precise description of transformers.[53] Alammar's *The Illustrated Transformer* adds valuable visual intuition.[54] Sasha Rush's *The Annotated Transformer* walks through the implementation of a transformer model.[55] For a thorough introduction to NLP and large language models, consider the textbook by Jurafsky and Martin.[56]

Coining the term *foundation models*, Bommasani et al. provide a comprehensive overview, background, and discussion of large language models in the paper *On the opportunities and risks of foundation models*.[29]

Mikolov et al. introduced popularized the Penn Treebank dataset for language modeling.[57] The dataset is the UPenn Treebank portion of the WSJ corpus comprising 930K words in the training set, 74K words in a validation set and 82K words in the test set. It became a standard corpus that many researchers reported perplexity numbers on.

Scaling laws have been the subject of much research beyond the scope of this chapter. One line of work studies the possibility of predicting downstream task performance—rather than test loss—from model scale. Evidence against it comes from the observation of *emergent capabilities*: Downstream task performance can vary unexpectedly with model scale.[35] However, in 2024, Gadre et al. show a power-law relationship between perplexity and the average benchmark performance in a suite of downstream tasks.[5] In the opposite direction, Maor, Carmon, and Berant argue that scaling laws for downstream tasks requires additional computational work.[58] In addition, Lourie, Hu, and Cho demonstrate numerous cases where training loss fails to predict downstream task performance.[59] The evidence is that there are no reliable scaling laws for downstream performance. The next chapter sheds more light on this topic.

Varoquaux, Luccioni, and Whittaker discuss the negative consequences of AI scaling.[13] Not all problems require scale or benefit equally from scale.

Benchmarks overemphasize the benefits of scaling. Continued scaling is environmentally unsustainable. In addition, scaling may lead to a concentration of power as fewer companies are able to compete over large models. Finally, scaling forces more aggressive data collection practices that lead to several problems. Paullada et al. survey ethical pitfalls in the development of machine learning datatasets with a focus on NLP benchmark datasets.[60]

Liu et al.[61] give a help history of the ImageNet model development leading up to CLIP era models. Fang et al. demonstrate that the increase in robustness of the CLIP model is due to greater diversity in the training data.[62]

Bibliography

1. Sutskever, I., Vinyals, O. & Le, Q. V. *Sequence to sequence learning with neural networks* in *Neural Information Processing Systems (NeurIPS)* (2014) (↑ 2).
2. Vaswani, A. *et al. Attention is all you need* in *Neural Information Processing Systems (NeurIPS)* (2017) (↑ 2).
3. Uszkoreit, J. & Museum, C. H. *Chatbots Decoded Interview: Jakob Uszkoreit* Video interview, Computer History Museum "Chatbots Decoded" exhibition. Published on YouTube, created for the CHM "Chatbots Decoded" exhibit, edited. 2024 (↑ 3, 26).
4. Pavlus, J. *When* ChatGPT *Broke an Entire Field: An Oral History* Accessed 2025-06-09. https://www.quantamagazine.org/when-chatgpt-broke-an-entire-field-an-oral-history-20250430/ (↑ 3).
5. Gadre, S. Y. *et al.* Language models scale reliably with over-training and on downstream tasks. *arXiv:2403.08540* (2024) (↑ 10, 26).
6. Devlin, J., Chang, M.-W., Lee, K. & Toutanova, K. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding* in *Proc. 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)* (2019), 4171–4186 (↑ 10).
7. Radford, A. *et al.* Language models are unsupervised multitask learners. *OpenAI blog* **1,** 9 (2019) (↑ 11).
8. Brown, T. *et al. Language models are few-shot learners* in *Neural Information Processing Systems (NeurIPS)* (2020), 1877–1901 (↑ 11, 13).
9. Schreiner, M. *GPT-4 architecture, datasets, costs and more leaked* Updated 2023-07-11. https://the-decoder.com/gpt-4-architecture-datasets-costs-and-more-leaked/ (2025) (↑ 11).
10. De Vries, A. The growing energy footprint of artificial intelligence. *Joule* **7,** 2191–2194 (2023) (↑ 12).
11. Strubell, E., Ganesh, A. & McCallum, A. *Energy and policy considerations for modern deep learning research* in *Proc. AAAI conference on artificial intelligence* **34** (2020), 13693–13696 (↑ 12).
12. Luccioni, A. S., Viguier, S. & Ligozat, A.-L. Estimating the carbon footprint of bloom, a 176b parameter language model. *Journal of machine learning research* **24,** 1–15 (2023) (↑ 12).
13. Varoquaux, G., Luccioni, S. & Whittaker, M. *Hype, Sustainability, and the Price of the Bigger-is-Better Paradigm in AI* in *ACM Conference on Fairness, Accountability, and Transparency (FAccT)* (2025), 61–75 (↑ 12, 13, 26).

14. Villalobos, P. *et al.* Will we run out of data? an analysis of the limits of scaling datasets in machine learning. *arXiv:2211.04325* **1** (2022) (↑ 12).
15. Baack, S. & Insights, M. Training data for the price of a sandwich. *Retrieved May* **9,** 2024 (2024) (↑ 13).
16. Smith, J. R. *et al. Dirt cheap web-scale parallel text from the common crawl* in (2013) (↑ 13).
17. Wenzek, G. *et al.* CCNet: Extracting high quality monolingual datasets from web crawl data. *arXiv:1911.00359* (2019) (↑ 13).
18. Raffel, C. *et al.* Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research* **21,** 1–67 (2020) (↑ 13).
19. Villalobos, P. *et al. Will we run out of data? Limits of LLM scaling based on human-generated data* 2024. arXiv: 2211.04325 [cs.LG]. https://arxiv.org/abs/2211.04325 (↑ 13).
20. O'Brien, M. *AI 'gold rush' for chatbot training data could run out of human-written text* The Associated Press. https://apnews.com/article/ai-artificial-intelligence-training-data-running-out-9676145bac0d30ecce1513c20561b87d (2025) (↑ 13).
21. Chowdhury, H. & Langley, H. *The AI world's most valuable resource is running out, and it's scrambling to find an alternative: 'fake' data* https://www.businessinsider.com/ai-synthetic-data-industry-debate-over-fake-2024-8 (2025) (↑ 13).
22. Practical Law Intellectual Property & Technology. Key Rulings on GenAI Training and Copyright Fair Use. *Practical Law The Journal: Litigation.* AI Monitor, July 2025 issue. https://www.reuters.com/practical-law-the-journal/litigation/key-rulings-genai-training-copyright-fair-use-2025-07-01/ (2025) (July 2025) (↑ 13).
23. The Authors Guild. *Understanding the AI Class Action Lawsuits* Authors Guild. https://authorsguild.org/news/ai-class-action-lawsuits/ (2025) (↑ 13).
24. Terranova, T. in *Digital labor* 33–57 (Routledge, 2012) (↑ 14).
25. Srnicek, N. *Platform capitalism* (John Wiley & Sons, 2017) (↑ 14).
26. Zuboff, S. in *Social theory re-wired* 203–213 (Routledge, 2023) (↑ 14).
27. Bender, E. M., Gebru, T., McMillan-Major, A. & Shmitchell, S. *On the dangers of stochastic parrots: Can language models be too big?* in *ACM Conference on Fairness, Accountability, and Transparency (FAccT)* (2021), 610–623 (↑ 14).
28. Crawford, K. *The atlas of AI: Power, politics, and the planetary costs of artificial intelligence* (Yale University Press, 2021) (↑ 14).
29. Bommasani, R. *et al.* On the opportunities and risks of foundation models. *arXiv:2108.07258* (2021) (↑ 14, 18, 26).
30. Bender, E. M. & Hanna, A. *The AI Con: How to fight big tech's hype and create the future we want* (Random House, 2025) (↑ 14).
31. Kaplan, J. *et al.* Scaling laws for neural language models. *arXiv:2001.08361* (2020) (↑ 14).
32. Hoffmann, J. *et al.* Training compute-optimal large language models. *arXiv:2203.15556* (2022) (↑ 16).
33. Clauset, A., Shalizi, C. R. & Newman, M. E. Power-law distributions in empirical data. *SIAM review* **51,** 661–703 (2009) (↑ 17).
34. MacKenzie, D. *An engine, not a camera: How financial models shape markets* (Mit Press, 2008) (↑ 18).
35. Wei, J. *et al. Emergent Abilities of Large Language Models* 2022. arXiv: 2206.07682 [cs.CL]. https://arxiv.org/abs/2206.07682 (↑ 18, 26).

36. Schaeffer, R., Miranda, B. & Koyejo, S. *Are emergent abilities of large language models a mirage?* in *Neural Information Processing Systems (NeurIPS)* (2024) (↑ 18).
37. Rajpurkar, P., Zhang, J., Lopyrev, K. & Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250* (2016) (↑ 19).
38. Rajpurkar, P., Jia, R. & Liang, P. Know what you don't know: Unanswerable questions for SQuAD. *arXiv:1806.03822* (2018) (↑ 19).
39. Bowman, S. R., Angeli, G., Potts, C. & Manning, C. D. A large annotated corpus for learning natural language inference. *arXiv:1508.05326* (2015) (↑ 19).
40. Bender, E. M. & Koller, A. *Climbing towards NLU: On meaning, form, and understanding in the age of data* in *Proc. 58th annual meeting of the association for computational linguistics* (2020), 5185–5198 (↑ 20).
41. Levesque, H., Davis, E. & Morgenstern, L. *The winograd schema challenge* in *Thirteenth international conference on the principles of knowledge representation and reasoning* (2012) (↑ 20).
42. Bender, D. *Establishing a Human Baseline for the Winograd Schema Challenge.* in *MAICS* (2015), 39–45 (↑ 21).
43. Sakaguchi, K., Bras, R. L., Bhagavatula, C. & Choi, Y. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM* **64,** 99–106 (2021) (↑ 21).
44. Paperno, D. *et al.* The LAMBADA dataset: Word prediction requiring a broad discourse context. *arXiv:1606.06031* (2016) (↑ 21).
45. Wang, A. *et al. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding* in *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP* (2018), 353–355 (↑ 22).
46. Bowman, S. R. & Dahl, G. E. What will it take to fix benchmarking in natural language understanding? *arXiv:2104.02145* (2021) (↑ 22).
47. Radford, A. *et al. Learning transferable visual models from natural language supervision* in *International Conference on Machine Learning (ICML)* (2021), 8748–8763 (↑ 23, 24).
48. Schuhmann, C. *et al. Laion-5b: An open large-scale dataset for training next generation image-text models* in *Neural Information Processing Systems (NeurIPS)* (2022), 25278–25294 (↑ 24).
49. Ilharco, G. *et al. OpenCLIP* version 0.1. July 2021. https://doi.org/10.5281/zenodo.5143773 (↑ 24).
50. Mayilvahanan, P., Wiedemer, T., Rusak, E., Bethge, M. & Brendel, W. Does CLIP's Generalization Performance Mainly Stem from High Train-Test Similarity? *arXiv:2310.09562* (2023) (↑ 25).
51. Mayilvahanan, P. *et al.* In search of forgotten domain generalization. *arXiv:2410.08258* (2024) (↑ 25).
52. Shannon, C. E. Prediction and entropy of printed English. *Bell system technical journal* **30,** 50–64 (1951) (↑ 26).
53. Phuong, M. & Hutter, M. Formal algorithms for transformers. *arXiv:2207.09238* (2022) (↑ 26).
54. Alammar, J. The illustrated transformer. *The Illustrated Transformer–Jay Alammar–Visualizing Machine Learning One Concept at a Time* **27,** 1–2 (2018) (↑ 26).
55. Rush, A. M. *The annotated transformer* in *Workshop for NLP open source software (NLP-OSS)* (2018), 52–60 (↑ 26).
56. Jurafsky, D. & Martin, J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language*

Models 3rd. Online manuscript released August 24, 2025. https://web.stanford.edu/~jurafsky/slp3/ (2025) (↑ 26).
57. Mikolov, T., Karafiát, M., Burget, L., Cernockỳ, J. & Khudanpur, S. *Recurrent neural network based language model.* in *Interspeech* **2** (2010), 1045–1048 (↑ 26).
58. Ivgi, M., Carmon, Y. & Berant, J. Scaling laws under the microscope: Predicting transformer performance from small scale experiments. *arXiv:2202.06387* (2022) (↑ 26).
59. Lourie, N., Hu, M. Y. & Cho, K. Scaling Laws Are Unreliable for Downstream Tasks: A Reality Check. *arXiv:2507.00885* (2025) (↑ 26).
60. Paullada, A., Raji, I. D., Bender, E. M., Denton, E. & Hanna, A. Data and its (dis) contents: A survey of dataset development and use in machine learning research. *Patterns* **2** (2021) (↑ 27).
61. Liu, Z. *et al. A convnet for the 2020s* in *Conference on Computer Vision and Pattern Recognition (CVPR)* (2022), 11976–11986 (↑ 27).
62. Fang, A. *et al. Data determines distributional robustness in contrastive language image pre-training (clip)* in *International Conference on Machine Learning (ICML)* (2022), 6216–6234 (↑ 27).