Transformer Reading Group

Morgan Sinclaire

Contents

The following notes grew out of an annotated list of “suggested papers” for my transformers reading group. I became a perfectionist about the annotations, and it also served as a place to make my own notes as I learned more about new developments in the field. Hence, this should be relatively self-contained (but please let me know about any mistakes).

0 Week 0 Prerequisites

Week 1 of this class will assume general familiarity with neural networks and deep learning (DL). For those with strong math background, this can be picked up fairly quickly over Winter Break with resources such as the following:

- 3Blue1Brown's "Neural Networks" Youtube playlist: Videos 1-4 cover the fundamentals of DL from scratch; videos 5-8 preview what we'll cover week 1 of the class.
  - There are a zillion other blog/Youtube introductions, e.g. this.
- Michael Nielsen's Neural Networks and Deep Learning online book: Of the 6 chapters, 1-2 cover the fundamentals with more rigor, while 3 is also helpful to consult.
- Goodfellow/Bengio/Courville Deep Learning textbook: A more comprehensive reference, but not as good for cover-to-cover reading.

1 Week 1 Introductory Concepts

Assuming familiarity with general DL, the next ideas to understand are (in order) word/semantic embeddings, the attention mechanism, and the neural network design which puts these together in an efficient way: the transformer architecture. There are a zillion blog/video explanations which may speak to different perspectives better, but in my opinion the following are the best:

- 3Blue1Brown's "Neural Networks" Youtube playlist: Videos 5-8 cover transformers, and in my opinion this is the best introduction to them I've seen, (unlike many sources from the 2010s such as Alammar's blog and the original paper, it focuses on the decoder-only architecture which is more dominant today).
- Brandon Rohrer's Transformers from Scratch: Self-contained introduction with illustrations, probably the 2nd best in my opinion. It is a bit long for a first sitting,[1] and goes into a bit more depth than 3B1B (with links in many of the later sections to read more).
- Jay Alammar's "The Illustrated Transformer": The most popularly referenced introduction, very readable. Only drawback: focuses on the obsolete encoder-decoder cross-attention setup from 2017, rather than the decoder-only self-attention architecture of GPT which I think is more natural anyways.
  - He also has "The Illustrated GPT-2", but I haven't read this myself.
- The iconic 2017 paper introducing the transformer, "Attention Is All You Need": Well-written as papers go, but I still think more modern and intuitive blogs/videos introduce it better.
  - "The Annotated Transformer": Harvard's NLP group wrote PyTorch code implementing the paper section-by-section.

2 Implementation-Level Guides

The following resources go into more depth on how to actually implement transformers and related algorithms.

- Andrej Karpathy, "Let's build GPT: from scratch, in code, spelled out." video: Walks through how to implement a GPT in PyTorch.
  - Actually, his whole playlist is excellent.

[1] Can ignore sections on nth order sequence models; these describe (skip) n-gram models, which are just a toy language model simpler than transformers.

3 Architectural Components

Diving further into the weeds of the transformer architecture, there are lots of tidbits that cause one to ask "why would you design it that way?"

- "Formal Algorithms for Transformers": Overview of algorithmic components of a transformer, with explicit detail and pseudocode for every piece.
- Weight tying paper, "Using the Output Embedding to Improve Language Models": The practice of enforcing $W_U = W_E^\top$, which is ubiquitous but I haven't found many great explanations for *why* it's done. Note the paper is from 2016 and this was originally done for RNNs.
- LayerNorm paper, "Layer Normalization": Inspired by BatchNorm in CNNs, normalizes the activations in a given layer to have mean $\mu = 0$ and variance $\sigma^2 = 1$.
- RMSNorm paper, "Root Mean Square Layer Normalization": A slightly more efficient variation of LayerNorm.

Parallelization

- Lilian Weng, "How to train really large models on many GPUs?": Guide to the different ways of parallelizing training.

4 Generative Pretraining

Early transformer models such as the original one (2017) were designed for fairly specialized tasks. GPT-1 (2018) and BERT (2018) introduced the idea of *self-supervised pretraining*: training the model for a long time on a big corpus to do a generic text-prediction task. This task is not directly useful in itself but it turns the massive sea of raw internet text into (effectively) labeled training data (hence "self-supervised"). Moreover, getting lower and lower loss on this very general task requires increasingly sophisticated abilities[2] so that after a large training run, we get a jack-of-all-trades-master-of-none type of reasoner (which we can then fine-tune to be masterful at tasks that are actually useful, hence "*pre*training").

BERT was trained to predict tokens randomly masked in the middle of text, and was initially the more influential model, but GPT-1's architecture proved to be better at scale when it grew into GPT-2 and GPT-3. It was trained specifically at *next-token* prediction: when fed some text of n tokens, it will implicitly generate $n + 1$ next-token predictions as it scans left-to-right.[3] Doing this repeatedly allows it to generate whole paragraphs of text, hence (unlike BERT's method) this is called *generative pretraining*.

- GPT-1 Paper, Improving Language Understanding by Generative Pre-Training
- GPT-2 Paper, Language Models are Unsupervised Multitask Learners

5 Data Preprocessing

A popular gloss on LLMs is that they are "trained on the whole internet". While a useful abstraction, in reality one has to carefully curate a dataset that has mostly high-quality content while

[2]For illustrative examples see Gwern's essay below on "The Scaling Hypothesis"

[3]Albeit computationally, it will do this in parallel in an atemporal, parallelized manner, which is what makes it so efficient at training compared to RNNs.

still being large. GPT-1 was trained on BookCorpus, a dataset of 7,000 books. GPT-2 and GPT-3 were primarily trained on carefully filtered/cleaned versions of Common Crawl, similar to the open-source Colossal Clean Crawled Corpus (C4). Additionally, The Pile is another open-source dataset that is commonly used.

- The Pile paper, The Pile: An 800GB Dataset of Diverse Text for Language Modeling
- C4 "paper", Documenting Large Webtext Corpora: A Case Study on the Colossal Clean Crawled Corpus: C4 was actually published without a paper or much documentation, but this paper was later written up to fill that void.

Aside from the dataset itself, one needs a way to *tokenize* the inputs into small but meaningful chunks that can be input to the transformer (visualizations: here and here). Byte pair encoding (BPE) turns out to be the *least bad* option for this.

- Andrej Karpathy, "Let's build the GPT Tokenizer" video: Quite comprehensive. 1st hour is basic examples, background on Unicode, and the BPE algorithm. 2nd hour gets into the bells and whistles one puts on top of BPE (e.g. using regexes to ensure we don't get unmeaningful tokens like "e t" despite their frequency being high).
- BPE in NLP paper, Neural Machine Translation of Rare Words with Subword Units: The BPE compression algorithm was devised in the 90s, but this 2015 paper appropriated the idea for tokenization in the NLP world, and this has been the standard since.

6 Supervised Fine-Tuning (SFT)

Besides data preprocessing, LLM training consists of 3 main phases (artist's depiction):

1. Self-supervised (generative) pretraining: This is where the *vast* majority of compute is needed, and where the core capabilities of LLMs come from. However, by itself this produces a very strange form of intelligence (the *base model*) that is difficult to interact with, unless has experience with making the carefully-targeted prompts this requires.
2. Supervised fine-tuning (SFT): To get a base model to do useful tasks at all (without skilled prompting), one has to train it on a smaller but carefully labeled dataset indicating the specific behavior you want from it.
3. RLHF: Further refines an LLM's behavior to not be insane [discussed in next section]

Terminology: both (2-3) are often called *fine-tuning*, because in both cases you are *fine*ly adjusting the behavior of your core model, *tuning* it gently towards the task(s) you want. But they are distinct stages, since the methodology is quite different (for one thing, as their names imply, SFT is supervised learning while RLHF is a form of reinforcement learning).

A base model, upon going through (2-3), becomes a *foundation model* such as GPT-4, Claude, or Gemini. This is a model which is directly useful to a consumer, but can also be further fine-tuned to be better at some specialized task.

Fine-Tuning a Foundation Model

At a high level, fine-tuning a base/foundation model M for some task T involves the following steps:

A) Obtain/create a labeled[4] dataset for T. This needn't be a *huge* dataset, but it should be substantial enough for a DL model to meaningfully learn. It also needs to be specific and targeted to your task.

B) Do supervised learning on the dataset, with some small changes from regular training.

   i) Adjust some of M's hyperparameters accordingly. For example, the learning rate should be low to make training more stable, since your model already has a lot of important but delicate information in its weights that you don't want to throw out with large updates.

   ii) Freeze all M's weights except the last layer: the early layers usually have the generalizable low-level features that don't need updating, and it's far more efficient to only retrain the last layer.

Actually, (B)(ii) pretty much describes how fine-tuning of CNNs used to be done in the 2010s. Nowadays, we have much more efficient ways of doing this, including LoRA, Quantized LoRA, and ReFT.

- Low-Rank Adaptation (LoRA) paper, LoRA: Low-Rank Adaptation of Large Language Models: If W is some $d_1 \times d_2$ weight matrix in the original model, the naïve approach is tantamount to training some ΔW of the same size (initialized to 0) such that $W + \Delta W$ are the fine-tuned weights. In LoRA, you instead train a "low-rank approximation" B, A of respective sizes $d_1 \times r$ and $r \times d_2$. Taking $r \ll \min(d_1, d_2)$, this is a lot less parameters to train.
- Representation FT (ReFT) paper, ReFT: Representation Finetuning for Language Models
- (I don't know of a good up-to-date overview of LLM fine-tuning, just a lot of fluffy blog posts and niche technical papers. This maybe comes close but I've only skimmed it.)

Fine-Tuning a Base Model

If we're starting instead from a base model and want to make a presentable consumer product, we first need to fine-tune it for instruction-following. For example, if the prompt is:

```
Prompt:  Explain the moon landing to a 6 year old.
```

Then with next-token prediction, a likely completion is:

```
Response:  Explain the theory of gravity to a 6 year old.
```

rather than a helpful explanation. To retrain it to do this, you need a bunch of humans to manually write high-quality examples of the instruction-response behavior desired. This is called *instruction-tuning*, and was introduced with InstructGPT in January 2022. Later, a similar dataset was created for conversations between a human and an AI assistant, to optimize specifically for these kinds of dialogues. The result was ChatGPT, which turned out to be unexpectedly successful.

Nowadays the LLM fine-tuning process is very proprietary (unsurprising since the simple idea of conversational fine-tuning turned out to be so $$$). But we do know there are other things modern LLMs are fine-tuned on, such as chain-of-thought reasoning and generally non-offensive outputs.

- Instruction-tuning paper, Training language models to follow instructions with human feedback, and blog post

[4]Depending on the task, sometimes you can just do next-token prediction on curated dataset, in which case it doesn't need labels.

7 Reinforcement Learning from Human Feedback (RLHF)

Imagine you're learning ballet. At first, your expert instructor demonstrates how the motions are supposed to be performed, and you try to copy them; this roughly corresponds to SFT. However, you need feedback on whether your emulations are good, or whether you're instead just picking up on more superficial aspects of what you saw. Therefore, your instructor holds up a sign 0-10 indicating the quality of your demonstrations. Initially you will likely get lots of 0's but this allows you to rule out dumber hypotheses on what you're supposed to be learning, because the point is that ballet is complicated and no expert will ever be able to explicitly write down what exactly you're supposed to learn.

Reinforcement Learning from Human Feedback (RLHF) is a type of RL because the agent is trying to maximize they're expected reward. Unlike in the traditional RL paradigm, the agent is never given an explicit reward function to maximize; instead this belongs to the paradigm of *reward modeling* where the agent is trying to build an implicit model of how the rewards work.[5] This is important because by default, an AI thinks very differently from humans, and will often learn bizarre things from SFT that fit the dataset but that humans will have never considered. RLHF is a more robust way to round out these edge cases, since if a system is doing something pathological a human has a chance to notice and downvote it.

RLHF was the main ingredient in InstructGPT, and (along with Anthropic's stricter Constitutional AI) is the main method used for doing this. As used in practice, these are known to have some annoying side effects. But even in theory, it is generally expected that these won't robustly scale to larger models, hence the quest to devise more powerful reward modeling techniques such as IDA (see below).

- RLHF paper, Deep reinforcement learning from human preferences and blog post
- Overview of the LLM training process, with somewhat more focus on the RLHF stage.
- Blog illustrating RLHF with LLMs, Illustrating Reinforcement Learning from Human Feedback (RLHF):
- Constitutional AI, paper and blog post: Anthropic's method of aligning Claude, which uses Reinforcement Learning from AI Feedback (RLAIF), a slightly stronger variant of RLHF.
- Survey of known problems with RLHF, Open Problems and Fundamental Limitations of Reinforcement Learning from Human Feedback: Paper isn't as long as it looks.

8 Prompt Engineering

As noted above, base models don't want to help you, they only "want" to do next-token prediction, e.g. if you ask a question, it will just respond with another question, so you'd need to input something more clever to get the answer you're looking for. Many people who played around with these in 2019-2022 came up with various methods for getting good/useful outputs, and this generally continued even after instruction- and conversation-tuning were introduced in 2022.

A poster child for this is chain-of-thought (CoT) prompting, where you just say "Let's think step-by-step", essentially giving the LLMs the mental scratchpad to do better multi-step reasoning (there are more precise explanations for why this helps). Another example is few-shot learning, where (e.g.) you start your prompt with a few Q-and-A pairs, then list your actual Q, at which point it completes this with an A.

[5] As such, RLHF is not an algorithm but a way of framing the abstract problem of RL (compare with traditional reward function maximization, imitation learning, or inverse RL as different paradigms). In principle almost any RL algorithm can be used to solve this problem, in practice proximal policy optimization (PPO) is a common choice.

This past year, most top labs seem to have gotten better at drilling most of these more general techniques into their fine-tuning.[6] Does this mean prompt engineering is obsolete? I don't know! But I'd be curious to see some informed, up-to-date analysis on this topic.

- The GPT-3 Paper, Language Models are Few-Shot Learners: Despite its significance, GPT-3 had almost no new architectural innovations over GPT-2, it just showcased the power of scaling it up. As such, the paper is mostly a discussion of its emergent capabilities, in particular this few-shot learning ability which basically wasn't present in GPT-2.

- Chain-of-Thought (CoT) paper, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models and blog post: The basic idea is simple and had actually been (re)discovered multiple times: on a prompt like "The 50th Fibonacci number is", this is a very hard next-token prediction task for the LLM, since it must compute it in one single forward pass through the network. There's only so much computation that can be done in a single forward pass, so by stretching the answer out into an "inner-monologue/scratchpad" of hundreds of individually easy-to-predict tokens, you give it more time to think through the answer.

9 Interpretability

A lot of prior research in CNNs involved fairly shallow notions of interpretability (e.g. saliency maps). Interpretability research in transformers is often more ambitious, in part because the discrete tokens can be traced more meaningfully than continuous pixel values. *Mechanistic interpretability (MI)* aims to reverse-engineer the algorithms learned by an LLM, and ultimately decompose the computation graph into smaller circuits that compute specific functions.[7]

In neuroscience and DL, *monosemanticity* refers to the idea that neurons and features/concepts have a 1-1 correspondence. This is cartoonishly wrong, but *if* it were true this would be great for interpretability. In reality, for efficiency reasons, neurons are *polysemantic* and correspond to multiple features at once, but we can imagine trying to unfold it into a much larger monosemantic model that we can interpret better. The *superposition hypothesis* is a proposed explanation for how neurons perform polysemanticity: it comes down to the counterintuitive geometric fact that in $\mathbb{R}^n$, we can only fit n mutually orthogonal vectors, but we can fit exponentially many near-orthogonal vectors (see 17:04 of this video for a nice visualization). A major ongoing research program in MI is to:

a) Test/investigate the superposition hypothesis.

b) Use it to do this unfolding into a more interpretable model (usually a *sparse autoencoder (SAE)*).

c) Break this down into individual circuits that can be pieced together in a reductionist manner.

There are other threads in MI, such as more concrete circuit decompositions of (parts of) existing transformers, analysis of specific attention heads (esp. *induction heads*, which are heads that apparently carry out basic inductive reasoning tasks), and applications of causal tracing.

- Zoom In: An Introduction to Circuits: An introduction to MI concepts in the context of CNNs.

- Toy Models of Superposition: More or less carries out (a) above.

[6] I'm deeply confused why it took so long; in 2023 I was still getting much better responses with simple CoT tricks long known by then.

[7] One can analogize this to how one might reverse engineer a binary computer program, except the binary is instead the NN weights.

- Towards Monosemanticity: Decomposing Language Models With Dictionary Learning: "Towards" (b) above.
- Scaling Monosemanticity: Extracting Interpretable Features from Claude 3 Sonnet: The last paper, but on bigger SOTA models.
- Fact Finding: Attempting to Reverse-Engineer Factual Recall on the Neuron Level: Compared to the attention heads, the MLP layers are even more mysterious. This is some preliminary work investigating how MLP layers store facts. (It's a series of blog posts, but the first already seems substantial enough to be a paper).
  - The high-level ideas were actually covered in this 3Blue1Brown video; probably better to watch this first.
- How to use and interpret activation patching: Exposition of activation patching aka causal tracing, a technique for figuring out which activations are responsible for a given output.
- Recent 30-page survey on MI, Mechanistic Interpretability for AI Safety – A Review
- A Mathematical Framework for Transformer Circuits: The in-depth foundational paper on MI for transformers, but it is pretty dense. Some of the technical hypotheses are considered to be overstated, but the conceptual frameworks developed (esp. in the overview) have been foundational to subsequent advances in the 3 years since.

10 Scaling Laws

The basic ideas of neural networks had been around since the dawn of AI in the 1950s, they just didn't show any interesting results until after many decades of compute growth. Indeed, the deep learning boom only arrived with AlexNet, which was notable for simply being much larger due to efficient use of GPUs rather than any major architectural innovation. In 2017, Hestness et al. did the first systematic modeling of *scaling laws*, tracing the relationship of a neural network's performance (in terms of its loss function) vs. the model's size (i.e. # of parameters) and its training dataset size. OpenAI was early to notice the implications of this and looked for ways to harness greater compute. They eventually settling on the GPT architecture, which was best able to take advantage of the internet as training data; there weren't any significant architectural changes between GPT-1 (2018) and GPT-4 (2023).

This has surprised a lot of AI researchers, but it makes more sense in the world of biology. For one thing, according to the latest neuroscience measurements, there doesn't seem to be much difference between a typical primate brain and a human brain; the latter is just a scaled-up version with a lot more neurons (and a longer "pretraining" stage). Furthermore the small size of the human genome ($\sim$3e7 non-junk base pairs vs. 8.6e10 neurons) upper-bounds the complexity of the architecture of the brain, indicating most of its magic must come from the learned connections.

- Gwern Branwen, "The Scaling Hypothesis", particularly the section "Why Does Pretraining Work?": A very lucid description of why it is not surprising that larger models are truly "smarter" and that novel capabilities emerge from simple increases in model size. Gives an intuitive explanation of why next-token prediction is (probably) AI-complete for similar reasons that general compression is Σ_1-complete. Also, the fact that while model/training size vs. perplexity is quite predictable, we know nothing about the relationship of perplexity vs. "intelligence".

    - Jacob Steinhardt, “More is Different for AI”: Series of blog posts discussing how quantitative changes in ML models lead to qualitative changes in behavior, analogous to similar phenomena in other fields (e.g. phase transitions in physics).
    - Richard Sutton, “The Bitter Lesson”: A short classic.

- Wikipedia, Neural Scaling Law along with Chinchilla (language model): A decent overview of DL/LLM scaling laws and some of the more well-known papers on it.
    - Kaplan et al. scaling paper (2020): The classic GPT-3-era scaling analysis, which did a regression analysis on several model inputs as they affected performance. Resulting equation found model size most important, followed by training set size (superseded by Chinchilla scaling).
    - Chinchilla scaling paper (2022): Reanalyzed things by training a wider variety of models of different model and training set sizes, eventually finding a model that outperformed GPT-3 with 40% of the parameters but on a data set 4 times the size. The main finding was that training set size is the most important factor in performance, and at a fixed amount of compute it's better to train a somewhat smaller for longer (compared to OpenAI's models), i.e. StAcK MoRe LaYeRs becomes PiLe MoRe DaTa.
    - DeepSeek V1 paper (2024): Noted that Kaplan/Chinchilla didn't seriously analyze how compute-optimal hyperparameters would change with scale (e.g. the learning rate or batch size) and investigated this further.

- Epoch AI, various blogs/reports: Probably the best overall resource for up-to-date scaling graphs and their analysis (this is the main thing they do), and less dry than Wikipedia/papers.

At an estimated 1-2 trillion parameters and $50-100 million training cost, GPT-4 represented the high water mark of Kaplan scaling. Following Chinchilla scaling, models released since then have actually gotten smaller. Furthermore, it seems the data is finally running out, especially high-quality data like academic journals/Wikipedia/Github. Reportedly, OpenAI recently attempted 2 large training runs of roughly half a billion dollars to make GPT-5, but performance stagnated in large part due to lack of data.

However, LLM progress itself has *not* stagnated from this, because it is driven by 3 primary factors:

1. Simple scaling: we have our simple scaling curves. Even with 0 innovation, we can simply dump in more compute to get more performance, i.e. “move along the curve.”

2. Shifting the curve: we can make efficiency improvements to the basic GPT architecture to streamline training/inference, allowing greater performance at a given level of compute.

3. New paradigms: in late 2024, the top labs finally started to succeed at using RL on CoTs at inference time to improve multi-step reasoning abilities. This opens up the new paradigm of *inference-scaling*, which will certainly have its own curve.

10.1 Inference Scaling

This new paradigm (discussed below under “Reasoning Models”) will have its own scaling laws, which are still being worked out.

- Snell et al. 2024: DeepMind's research into how to allocate a fixed amount of compute to pretraining vs. inference.

- Jones 2021: An analysis of toy models of inference-scaling, finding some conceptually interesting trends.

11 Efficiency Improvements

Behind the scenes, the top labs have been making algorithmic tweaks to make the basic GPT architecture run faster. Epoch AI estimates that "the level of compute needed to achieve a given level of performance has halved roughly every 8 months, with a 95% confidence interval of 5 to 14 months". DeepSeek's V3 model pretty much falls on this curve: at $5.3 million for pretraining, it cost an order of magnitude less than GPT-4, but this is pretty much what we'd expect when comparing it to something 2 years older. However, DeepSeek's publications have allowed us to see what these algorithmic improvements look like.

That being said, I don't think these are very important for really understanding transformers. They're basically efficiency hacks to do faster what the GPT architecture was already doing. Still, I list a few of them here.

Mixture of Experts

Mixture-of-experts (MoE) is an old idea which has found increasing use in LLM architectures. Oversimplified: our model may consist of 8 submodels (*experts*) with a smaller *routing/gating* network at the beginning, which makes a decision of which (say) 2 experts will handle a given input. Provided we're usually picking the "right" experts, performance will be comparable to the full 8, but at a quarter of the cost.

The details are more complicated (e.g. the routing happens at each layer) and there are a lot of engineering challenges involved in getting the above framework to work well. In fact, it wasn't until 2017 that Shazeer et al. got MoE to work well with any kind of neural network, and it's taken a few years to make it work with LLMs. DeepSeek pioneered this among open-souce models (it is widely believed the top closed-source models use MoE, but obviously we don't know the details).

- Cameron Wolfe, "Mixture-of-Experts (MoE) LLMs": Excellent overview of the evolution of MoE, from Shazeer et al. to the DeepSeek papers.
- DeepSeekMoE paper: Where they describe most of the MoE innovations they subsequently used in V2 and V3.

Latent Attention

Key-value caching is a standard efficiency technique for speeding up prediction of multiple serial tokens: store the key and value vectors from previous tokens rather than recomputing in each forward pass. But this uses a lot of memory, so various techniques have been proposed to reduce the KV cache. Most of these degrade performance significantly, but DeepSeek's *latent attention* does not.

- DeepSeek-V2 paper: Introduced latent attention, and also extended their MoE work.

12 Deep Reinforcement Learning (DRL) Background

Deep Reinforcement Learning (DRL) is particularly relevant to LLMs, because of 1) RLHF and 2) since late 2024, the new paradigm of LLMs such as o1 has been to use RL to train on CoTs to improve multi-step reasoning abilities (see "Inference Scaling" below). In both cases, proximal policy optimization (PPO) has been the algorithm of choice. Consequently, it can be helpful to gain more background on RL/DRL.

General Background

- OpenAI's Introduction to RL: Starts from the very basics in Part 1, has a nice taxonomy of RL methods in Part 2, and Part 3 gets in policy optimization (but not PPO specifically).
- OpenAI's RL Algorithms Docs: A sequel to the previous, which goes into more depth on policy gradient stuff (the first 3 parts, which conclude with PPO, seem appropriate for 1 week).
- Sutton & Barto is the standard textbook on RL. It was written before the DL revolution but the "D" in DRL is conceptually straightforward (just very fiddly in practice).

Proximal Policy Optimization

- PPO paper, Proximal Policy Optimization Algorithms, and blog post
- 20-minute video explainer of PPO
- Sutton & Barto Chapter 13 covers policy gradient methods, the family of algorithms that includes PPO (albeit PPO itself is newer than the book).
- GRPO paper: Group Relative Policy Optimization (GRPO) is DeepSeek's variant of PPO that they've been using instead.

13 Reasoning Models

An inherent property of the GPT architecture is that if an LLM makes a mistake somewhere, it will condition on that mistake for the rest of its CoT. However, we can have it stochastically generate multiple CoTs, evaluate which ones "worked", and use this data to retrain the model using RL. For example, AlphaGo did this for Go using *Monte Carlo tree search (MCTS)*; this general idea is sometimes called *expert iteration* or *iterated distillation and amplification*. For general background, see the AlphaGo Zero paper, the ExIt paper, or this short video explainer.

To anyone at the top labs, this has always been an obvious idea to try. But by all indications, it just didn't work since the base models weren't good enough. But in the last few months, OpenAI introduced their o1 and o3 models, closely followed by Gemini Flash Thinking and the DeepSeek-R1. What these have in common is improved prowess in domains with verifiable rewards such as math and coding. As of 2/2025 it is not (publicly) clear whether this will easily transfer over to areas without unambiguous feedback on which CoTs "worked".

Obviously, running zillions of CoTs in parallel is expensive (in solving the Abstraction and Reasoning Corpus, o3 reportedly used over \$1000 per task for 500 tasks). This is true on the user side, but also a lot of this has to be done to generate the data to train. This means that larger training runs will likely use more *test-time* or *inference-time* compute; one presumes different hardware will become more important.

- "Scaling test-time compute with open models": Hugging Face attempts to replicate the tree search strategies used by o1.
- DeepSeek-R1 paper: DeepSeek's R1 model gets performance rivaling o1 at a tiny fraction of the cost. We don't have to guess the techniques because they open-sourced everything, even reporting the unsuccessful approaches. One notable ingredient: GRPO, their own variant of PPO.

14 Risks

Concerns about the eventual consequences of creating a fully general replacement for humans go back to Turing and von Neumann. What is unique about transformers among AI techniques is that they are the only form that may plausibly realize this outcome. Indeed, many of the researchers at the top 3 labs have been rather explicit recently on the reality of a credible near-term path to autonomous systems that outperform humans at most economically valuable work (i.e. AGI).[8] Based on the new RL paradigm, they will likely be highly agentic, especially since agents have tended to be more economically valuable than inert tools. And based on the aforementioned trendlines on scaling/efficiency improvements, once we have the first AGI agent we'll likely have millions shortly afterwards. The basic concerns can then be summarized as:

1. Within the next decade, we may develop a second species with intelligence surpassing humans (analogous to how humans surpass chimps).
2. This seems dangerous.

The primary observation is that human values form a very narrow subset of the corresponding semantic embedding space, and we don't have a reliable way to get an AI's values into that tiny pocket of $\mathbb{R}^n$. This is not a problem for weak AIs, because they are not dangerous and also more moldable by simple techniques like RLHF. But if an AI's internal values aren't aligned, and we scale its capabilities past what we are capable of controlling, this becomes a problem.

Failure Modes

- *Goal misgeneralization* (description, vignette): When the AI is trained to have human values (via RLHF etc.) it usually learns something different. The difference is subtle enough that we don't notice it in training, but it can make a big difference in deployment (e.g. ChatGPT would have innocuous behavior until prompts were CaPiTaLiZeD wEiRdLy).
- *Deceptive alignment/alignment faking* (description, vignette): Sometime during pretraining, a base model might develop weird goals and crystallize these. By the time it gets to RLHF, it will be smart enough to act nice so its weights aren't modified in the "wrong" ways. It may also play nice during low-stakes deployment, only playing its hand when (e.g.) put in charge of a national economy.

Empirical Findings

In previous years, concerns about safety were based on theoretical arguments: there simply weren't any AIs with world models, coherent goals, or any ability to engage in scheming/deception when their goals were in conflict with humans. However, the late-2024 models are beginning to show signs of doing this *sometimes*.

- "Alignment faking in large language models": Anthropic found that Claude 3 Opus sometimes engages in deceptive alignment. Looking over the "hidden" CoT transcripts, it is clear that Claude was scheming in a consequentialist way based on its current values.
  - Video discussion by authors, Anthropic blog post
- "Frontier Models are Capable of In-context Scheming": Ran late-2024 models on 6 scenarios where scheming was a viable option to achieve its goals. These are still somewhat weak forms of scheming, and most of the models did not do it, but it is still *more* scheming than previous models.

[8] Notably, this group was *not* saying this a few years ago.

- Blog summary

Alignment Proposals

Paul Christiano, the main inventor of RLHF, devised it as a first stab at these hard problems and never expected it to scale well to AGI. He also invented a more advanced form of reward modeling, Iterated Distillation and Amplification (IDA), but none of the current labs use it (presumably for cost reasons). Most alignment researchers expect that IDA could eventually run into problems (Christiano left OpenAI to found his own Alignment Research Center (ARC) to do more theoretical research on overcoming these problems[9]).

- IDA paper, Supervising strong learners by amplifying weak experts and blog post, as well as an excellent 10-minute video explainer
- An overview of 11 proposals for building safe advanced AI: To my knowledge these largely remain the top approaches.

15 Historical References

These articles aren't part of "the whitelist", but it is sometimes enlightening to know about the history of ideas that led up to modern LLMs.

- Popular articles profiling the writing of "Attention Is All You Need":
  - Financial Times, "Transformers: the Google scientists who pioneered an AI revolution"
  - WIRED, "8 Google Employees Invented Modern AI. Here's the Inside Story"
  - New Yorker, "Was Linguistic A.I. Created by Accident?"
- Andrej Karpathy, "The Unreasonable Effectiveness of Recurrent Neural Networks": A nice time capsule of what the RNN-based SOTA in NLP looked like in 2015.
- "The Making of ChatGPT (35 Year History)": Half hour video walking through the history of NLP advances that led up to ChatGPT.

[9]Also Jan Leike, the other co-architect of RLHF often considered more of an optimist, quit OpenAI last year because "safety culture and processes have taken a backseat to shiny products."

Figure 1: “The document ended with a cartoony image of six Transformers in mountainous terrain, zapping lasers at one another.”
[Quote from above *WIRED* article; image made by a transformer still learning to count].