Ten Noteworthy AI Research Papers of 2023

SEBASTIAN RASCHKA, PHD
DEC 30, 2023

This year has felt distinctly different. I've been working in, on, and with machine learning and AI for over a decade, yet I can't recall a time when these fields were as popular and rapidly evolving as they have been this year.

To conclude an eventful 2023 in machine learning and AI research, I'm excited to share 10 noteworthy papers I've read this year. My personal focus has been more on large language models, so you'll find a heavier emphasis on large language model (LLM) papers than computer vision papers this year.

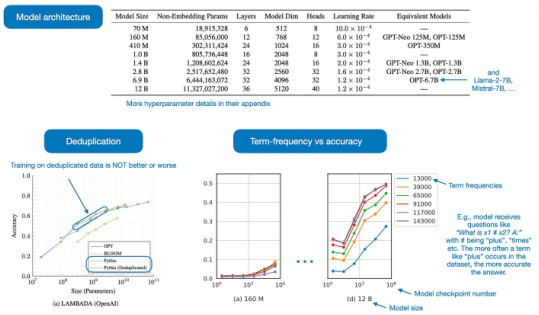
I resisted labeling this article "Top AI Research Papers of 2023" because determining the "best" paper is subjective. The selection criteria were based on a mix of papers I either particularly enjoyed or found impactful and worth noting. (The sorting order is a recommended reading order, not an ordering by perceived quality or impact.)

By the way, if you scroll down to the end of this article, you'll find a little surprise. Thanks for all your support, and I wish you a great start to the new year!

1) Pythia — Insights from Large-Scale Training Runs

With ***Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling***, the researchers originally released 8 LLMs ranging from 70M to 12B parameters (with both weights and data publicly released, which is rare).

But in my opinion, the standout feature of this paper is that they also released the training details, analyses, and insights (some of them shown in the annotated figure below).



Annotated charts from the Pythia paper, https://arxiv.org/abs/2304.01373

Here are some questions that the Pythia paper addresses:

1. Does pretraining on duplicated data (i.e., training for >1 epoch) make a difference? It turns out that deduplication does not benefit or hurt performance.
2. Does training order influence memorization? Unfortunately, it turns out that it does not. "Unfortunately," because if this was true, we could mitigate undesirable verbatim memorization issues by reordering the training data.
3. Does pretrained term frequency influence task performance? Yes, few-shot accuracy tends to be higher for terms that occur more frequently.
4. Does increasing the batch size affect training efficiency and model convergence? Doubling the batch size halves the training time but doesn't hurt convergence.

Today, only six months later, the LLMs are by no means groundbreaking. However, I am including this paper because it not only tries to answer interesting questions about training settings but is also a positive example regarding details and transparency. Moreover, the small LLMs in the <1B range are nice templates for small studies and tinkering, or starters for pretraining experiments (here's a link to their GitHub repository).

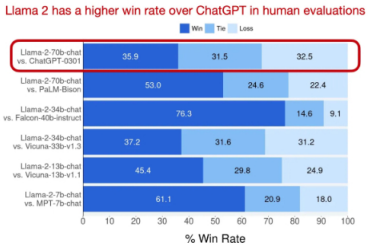
My wish for 2024 is that we see more studies like this and well-written papers in the coming year!

2) Llama 2: Open Foundation and Fine-Tuned Chat Models

Llama 2: Open Foundation and Fine-Tuned Chat Models is the follow-up paper to Meta's popular first Llama paper.

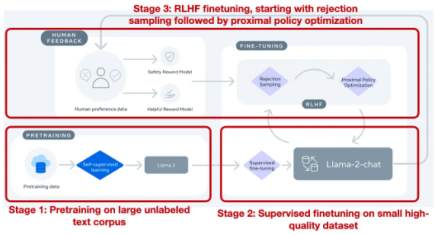
Llama 2 models, which range from 7B to 70B parameters, are one of the reasons this paper made it onto this list: these are still among the most capable and widely used openly available models. Worth noting is that the Llama 2 license also permits use in

commercial applications (see the Request to Access page for details).



Annotated figure from Llama 2 paper (https://arxiv.org/abs/2307.09288) comparing Llama 2 models to ChatGPT

On the model side, what differentiates the Llama 2 suite from many other LLMs is that the models come as standard pretrained models and chat models that have been finetuned via reinforcement learning with human feedback (RLHF, the method used to create ChatGPT) to follow human instructions similar to ChatGPT — RLHF-finetuned models are still rare.



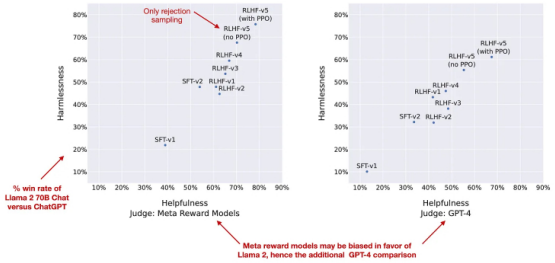
Annotated figure from Llama 2 paper (https://arxiv.org/abs/2307.09288) summarizing the RLHF process for instruction finetuning.

For more details on RLHF and how it's used in Llama 2, see my more comprehensive standalone article below.

LLM Training: RLHF and Its Alternatives
SEBASTIAN RASCHKA, PHD · SEPTEMBER 10, 2023
Read full story →

Next to the fact that Llama 2 models are widely used and come with RLHF instruction-finetuned variants, the other reason I decided to include the paper on this list is the accompanying in-depth 77-page research report.

Here, the authors also nicely illustrated the evolution of the Llama 2 70B Chat models, tracing their journey from the initial supervised finetuning (SFT-v1) to the final RLHF finetuning stage with PPO (RLHF-v5). The chart reflects consistent improvements in both the harmlessness and helpfulness axes, as shown in the annotated plots below.



Annotated figure from Llama 2 paper (https://arxiv.org/abs/2307.09288) showing the performance progression from the first iteration of the supervised finetuned model (SFT-1) to the final RLHF-finetuned chat model (RLHF-v5).

Even though models such as Mistral-8x7B (more later), DeepSeek-67B, and YI-34B top the larger Llama-2-70B models in public benchmarks, Llama 2 still remains a common and popular choice when it comes to openly available LLMs and developing methods on top of it.

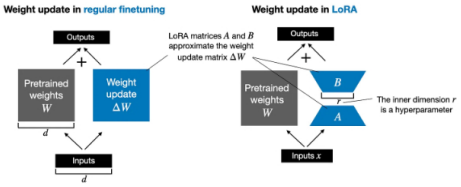
Furthermore, even though some benchmarks indicate that there may be better models, one of the bigger challenges this year has been the trustworthiness of benchmarks. For instance, how do we know that the models haven't been trained on said benchmarks and the scores aren't inflated? In classic machine learning, when someone proposed a new gradient boosting model, it was relatively easy to reproduce the results and check. Nowadays, given how expensive and complex it is to train LLMs (and the fact that most researchers either don't disclose the architecture or the training data details), it is impossible to tell.

To conclude, it's refreshing to see Meta doubling down on open source even though every other major company is now rolling out its own proprietary large language models (Google's Bard and Gemini, Amazon's Q, and Twitter/X's Grok, and OpenAI's ChatGPT).

3) QLoRA: Efficient Finetuning of Quantized LLMs

Quantized LLMs

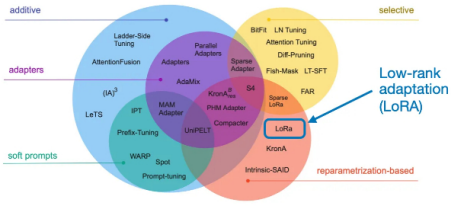
QLoRA: Efficient Finetuning of Quantized LLMs has been one of the favorite techniques in the LLM research and finetuning community this year because it makes the already popular LoRA (low-rank adaptation) technique more memory efficient. In short, this means that you can fit larger models onto smaller GPUs.



A short visual summary of regular LoRA

QLoRA stands for quantized LoRA (low-rank adaptation). The standard LoRA method modifies a pretrained LLM by adding low-rank matrices to the weights of the model's layers. These matrices are smaller and, therefore, require fewer resources to update during finetuning.

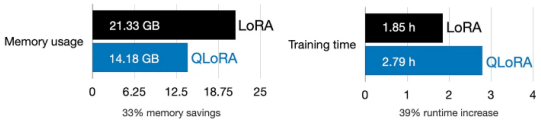
In QLoRA, these low-rank matrices are quantized, meaning their numerical precision is reduced. This is done by mapping the continuous range of values in these matrices to a limited set of discrete levels. This process reduces the model's memory footprint and computational demands, as operations on lower-precision numbers are less memory-intensive



Among the many efficient finetuning methods for LLMs, LoRA is among the most popular and widely used ones. Annotated figure from the excellent Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning survey.

According to the QLoRA paper, QLoRA reduces the memory requirements of a 65B Llama model to fit onto a single 48 GB GPU (like an A100). The 65B Guanaco model, obtained from quantized 4-bit training of 65B Llama, maintains full 16-bit finetuning task performance, reaching 99.3% of the ChatGPT performance after only 24 hours of finetuning.

I've also run many QLoRA experiments this year and found QLoRA a handy tool for reducing GPU memory requirements during finetuning. There's a trade-off, though: the extra quantization step results in an additional computation overhead, meaning the training will be a bit slower than regular LoRA.



Excerpt from my LoRA & QLoRA experiments that I wrote about previously here

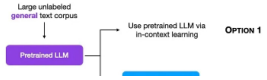
LLM finetuning remains as relevant as ever as researchers and practitioners aim to create custom LLMs. And I appreciate techniques like QLoRA that help make this process more accessible by lowering the GPU memory-requirement barrier.

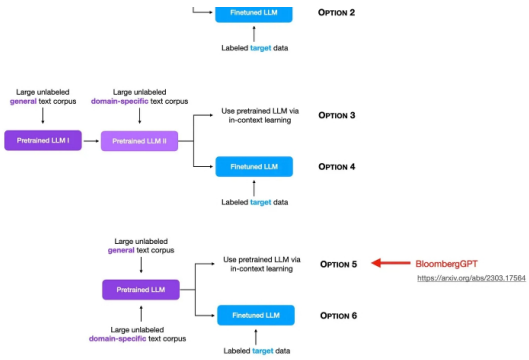
4) BloombergGPT: A Large Language Model for Finance

Looking at all the papers published this year, ***BloombergGPT: A Large Language Model for Finance*** may look like an odd choice for a top-10 list because it didn't result in a groundbreaking new insight, methodology, or open-source model.

I include it because it's an interesting case study where someone pretrained a relatively large LLM on a domain-specific dataset. Moreover, the description was pretty thorough, which is becoming increasingly rare. This is especially true when it comes to papers with authors employed at companies -- one of the trends this year was that major companies are becoming increasingly secretive about architecture or dataset details to preserve trade secrets in this competitive landscape (PS: I don't fault them for that).

Also, BloombergGPT made me think of all the different ways we can pretrain and finetune models on domain-specific data, as summarized in the figure below (note that this was not explored in the BloombergGPT paper, but it would be interesting to see future studies on that).





The different ways of pretraining and finetuning LLMs.

In short, BloombergGPT is a 50-billion parameter language model for finance, trained on 363 billion tokens from finance data and 345 billion tokens from a general, publicly available dataset. For comparison, GPT-3 is 3.5x larger (175 billion parameters) but was trained on 1.4x fewer tokens (499 billion).

Why did the authors use an architecture with "only" 50 billion parameters since GPT-3 is 3.5x larger? That's easier to answer. They adopted the Chinchilla scaling laws and found this to be a good size given the available size of the finance data.

Is it worth (pre)training the LLM on the combined dataset from scratch? Based on the paper, the model performs really well in the target domain. However, we don't know whether it's better than a) further pretraining a pretrained model on domain-specific data or b) finetuning a pretrained model on domain-specific data.

Despite the little criticism above, overall, this is an interesting paper that serves as an interesting case study and example for domain-specific LLMs; plus, it leaves room for further research on pretraining versus finetuning to instill knowledge into an LLM.

(PS: For those curious about a comparison to finetuning, as Rohan Paul shared with me, the "small" AdaptLLM-7B model outperforms BloombergGPT on one dataset and nearly matches its performance on three other finance datasets. Although BloombergGPT appears to be slightly better overall, it's worth noting that training AdaptLLM-7B cost about $100, in contrast to BloombergGPT's multi-million dollar investment.)

5) Direct Preference Optimization: Your Language Model is Secretly a Reward Model

Before discussing the ***Direct Preference Optimization: Your Language Model is Secretly a Reward Model*** paper, let's take a short step back and discuss the method it aims to replace, Reinforcement Learning from Human Feedback (RLHF).

RLHF is the main technique behind ChatGPT and Llama 2 Chat models. In RLHF, which I described in more detail in a separate article, we use a multi-step procedure:

1. Supervised finetuning: The model is initially trained on a dataset containing instructions and the desired responses.
2. Reward modeling: Human raters provide feedback on the model's outputs. This feedback is used to create a reward model, which learns to predict what kinds of outputs are to be preferred.
3. Proximal policy optimization (PPO): The model generates outputs, and the reward model scores each output. The PPO algorithm uses these scores to adjust the model's policy toward

generating higher-quality outputs. (This is a reinforcement learning algorithm used to finetune the model's policy.

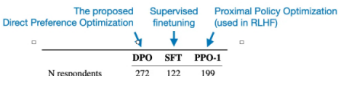
```
{
    "instruction": "Write a limerick about a
                    pelican.",
    "input": "",
    "output": "There once was a pelican so fine,
              \nHis beak was as colorful as
              sunshine,\nHe would fish all day,\nIn
              a very unique way,\nThis pelican was
              truly divine!\n\n\n"
},


{
    "instruction": "Identify the odd one out from
                    the group.",
    "input": "Carrot, Apple, Banana, Grape",
    "output": "Carrot\n\n"
},
```

Example of two training examples from a dataset for the supervised instruction finetuning step. Note that the "input" is optional.

While RLHF is popular and effective, as we've seen with ChatGPT and Llama 2, it's also pretty complex to implement and finicky.

The Direct Preference Optimization (DPO) paper introduces an algorithm that optimizes language models to align with human preferences **without** explicit reward modeling or reinforcement learning. Instead, DPO uses a simple classification objective.



GPT-4 (S) win %	47	27	13
GPT-4 (C) win %	54	32	12
Human win %	58	43	17
GPT-4 (S)-H agree	70	77	86
GPT-4 (C)-H agree	67	79	85
H-H agree	65	-	87

Annotated figures from the DPO paper, https://arxiv.org/abs/2305.18290

In DPO, we still keep the supervised finetuning step (step 1 above), but we replace steps 2 and 3 with a single step to further finetune the model on the preference data. In other words, DPO skips the reward model creation required by RLHF entirely, which significantly simplifies the finetuning process.

How well does it work? There haven't been many models trained with DPO until very recently. (This makes sense because DPO is also a relatively recent method.) However, one recent example is the Zephyr 7B model described in *Zephyr: Direct Distillation of LM Alignment*. Zephyr-7B is based on a Mistral-7B base LLM that has been finetuned using DPO. (There will be more on Mistral later.)

As the performance tables below reveal, the 7B-parameter Zephyr model outperformed all other models in its size class at the time of its release. Even more impressively, Zephyr-7B even surpassed the 10 times larger 70B-parameter Llama 2 chat model on the conversational MT-Bench benchmark as well.

Conversational performance benchmarks

Model	Size	Align	MT-Bench (score)	AlpacaEval (win %)
StableLM-Tuned-α	7B	dSFT	2.75	-
MPT-Chat	7B	dSFT	5.42	-
Xwin-LM v0.1	7B	dPPO	6.19*	$87.83_{1.15}$
Mistral-Instruct v0.1	7B	-	6.84	-
Zephyr	7B	dDPO	**7.34**	**$90.60_{1.03}$**
Falcon-Instruct	40B	dSFT	5.17	$45.71_{1.75}$
Guanaco	65B	SFT	6.41	$71.80_{1.59}$
Llama2-Chat	70B	RLHF	6.86	$92.66_{0.91}$
Vicuna v1.3	33B	dSFT	7.12	$88.99_{1.10}$
WizardLM v1.0	70B	dSFT	**7.71**	-
Xwin-LM v0.1	70B	dPPO	-	**$95.57_{0.72}$**
GPT-3.5-turbo	-	RLHF	7.94	$89.37_{1.08}$
Claude 2	-	RLHF	8.06	$91.36_{0.99}$
GPT-4	-	RLHF	**8.99**	**$95.28_{0.72}$**

Zephyr outperforms all LLMs of its size class

Zephyr is even on par with a 70B-parameter Llama 2 model

SFT = supervised finetuning
dSFT = distilled supervised finetuning
dPPO = distilled proximal policy optimization
dDPO = distilled direct preference optimization
RLHF = reinforcement learning with human feedback

Knowledge-based benchmarks

Model	Size	Align	ARC	Hella Swag	MMLU	Truthful QA
StableLM-Tuned-α	7B	dSFT	31.91	53.59	24.41	40.37
MPT-Chat	7B	dSFT	46.50	75.51	37.62	40.16
Xwin-LM v0.1	7B	dPPO	56.57	79.40	49.98	47.89
Mistral-Instruct v0.1	7B	dSFT	54.52	75.63	55.38	56.28
Zephyr	7B	dDPO	**62.03**	**84.52**	**61.44**	**57.44**
Falcon-Instruct	40B	dSFT	61.60	84.31	55.45	52.52
Guanaco	65B	SFT	65.44	86.47	62.92	52.81
Llama2-Chat	70B	RLHF	67.32	87.33	69.83	44.92
Vicuna v1.3	33B	dSFT	62.12	83.00	59.22	56.16
WizardLM v1.0	70B	dSFT	64.08	85.40	64.97	54.76
Xwin-LM v0.1	70B	dPPO	70.22	87.25	69.77	59.86

Zephyr outperforms all LLMs of its size class

Annotated Benchmarks of the Zephyr model (a DPO-finetuned LLM) from https://arxiv.org/abs/2310.16944

In summary, the appeal of the DPO paper lies in the simplicity of its method. The scarcity of chat models trained using RLHF, with Llama 2 as a notable exception, can likely be attributed to the complexity of the RLHF approach. Given this, I think it's reasonable to anticipate an increase in the adoption of DPO models in the coming year.

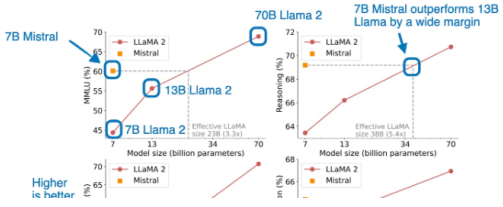
6) Mistral 7B

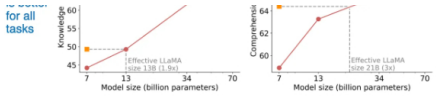
I must admit that the ***Mistral 7B paper*** wasn't among my favorites due to its brevity. However, the model it proposed was quite impactful.

I decided to include the paper on this list because the Mistral 7B model was not only very popular upon release, but also served as the base model, leading to the development of two other notable models: Zephyr 7B and the latest Mistral Mixture of Experts (MoE) approach. These models are good examples of the trend I foresee for small LLMs in (at least) the early half of 2024.

Before we discuss the Zephyr 7B and Mistral MoE models, let's briefly talk about Mistral 7B itself.

In short, The Mistral 7B paper introduces a compact yet powerful language model that, despite its relatively modest size of 7 billion tokens, outperforms its larger counterparts, such as the 13B Llama 2 model, in various benchmarks. (Next to the two-times larger Qwen 14B, Mistral 7B was also the base model used in the winning solutions of this year's NeurIPS LLM Finetuning & Efficiency challenge.)



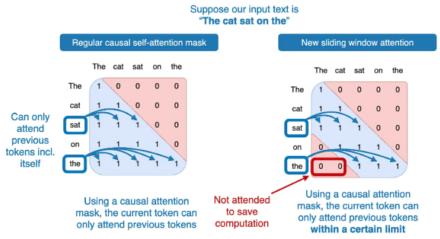


Annotated figure from https://arxiv.org/abs/2310.06825 comparing Mistral 7B and Llama 13B performances

Why exactly it is so good is unclear, but it might likely be due to its training data. Neither Llama 2 nor Mistral discloses the training data, so we can only speculate.

Architecture-wise, the model shares group-query attention with Llama 2. While being very similar to Llama 2, one interesting addition to the Mistral architecture is sliding window attention to save memory and improve computational throughput for faster training. (Sliding window attention was previously proposed in Child et al. 2019 and Beltagy et al. 2020.)

The sliding window attention mechanism used in Mistral is essentially a fixed-sized attention block that allows a current token to attend only a specific number of previous tokens (instead of all previous tokens), which is illustrated in the figure below.



Annotated figure from https://arxiv.org/abs/2310.06825 explaining sliding window attention.

In the specific case of 7B Mistral, the attention block size is 4096 tokens, and the researchers were training the model with up to 100k token context sizes. To provide a concrete example, in regular self-attention, a model at the 50,000th token can attend all previous 49,999 tokens. In sliding window self-attention, the Mistral model can only attend tokens 45,904 to 50,000 (since 50,000 - 4,096 = 45,904).

However, sliding window attention is mainly used to improve computational performance. The fact that Mistral outperforms larger Llama 2 models is likely not because of sliding window attention but rather despite sliding window attention.

Zephyr and Mixtral

One reason Mistral 7B is an influential model is that it served as the base model for Zephyr 7B, as mentioned earlier in the DPO section. Zephyr 7B, the first popular model trained with DPO to outperform other alternatives, has potentially set the stage for DPO to become the preferred method for finetuning chat models in the coming months.

Another noteworthy model derived from Mistral 7B is the recently released Mistral Mixture of Experts (MoE) model, also known as Mixtral-8x7B. This model matches or exceeds the performance of the larger Llama-2-70B on several public benchmarks.

Datasets	Mode	Mistral-7B-v0.1	Mixtral-8x7B	Llama2-70B
MMLU	PPL	64.1	71.3	69.7
BIG-Bench-Hard	GEN	56.7	67.1	64.9
GSM-8K	GEN	47.5	65.7	63.4
MATH	GEN	11.3	22.7	12.0
HumanEval	GEN	27.4	32.3	26.2
MBPP	GEN	38.6	47.8	39.6
ARC-c	PPL	74.2	85.1	78.3
ARC-e	PPL	83.6	91.4	85.9
CommonSenseQA	PPL	67.4	70.4	78.3
NaturalQuestion	GEN	24.6	29.4	34.2
TriviaQA	GEN	56.5	66.1	70.7
HellaSwag	PPL	78.9	82.0	82.3
PIQA	PPL	81.6	82.9	82.5
SIQA	GEN	60.2	64.3	64.8

OpenCompass benchmarks via https://github.com/open-compass/MixtralKit. Blue boxes highlight the best results in each row.

For more benchmarks, also see the official Mixtral blog post announcement. The team also released a Mixtral-8x7B-Instruct model that has been finetuned with DPO (but as of this writing there are no benchmarks comparing it to Llama-2-70-Chat, the RLHF-finetuned model).

```
{
    "dim": 4096,
    "n_layers": 32,
    "head_dim": 128,
    "hidden_dim": 14336,
    "n_heads": 32,
    "n_kv_heads": 8,
    "norm_eps": 1e-05,
    "vocab_size": 32000,
    "moe": {
        "num_experts_per_tok": 2,
        "num_experts": 8
    }
}
```

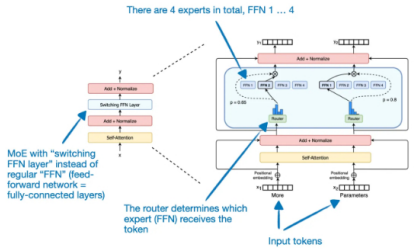
Mixtral architecture overview based on the param.json file that the Mistral team originally shared via a magnet link on social media

GPT-4 is also rumored to be an MoE consisting of 16 submodules. Each of these 16

submodules is rumored to have 111 billion parameters (for reference, GPT-3 has 175 billion parameters). If you read my AI and Open Source in 2023 article approximately two months ago, I mentioned that "It will be interesting to see if MoE approaches can lift open-source models to new heights in 2024". It looks like Mixtral started this trend early, and I am sure that this is just the beginning.

Mixture of Experts 101

If you are new to MoE models, here's a short explanation.



Annotated figure from Switch Transformers paper (https://arxiv.org/abs/2101.03961) that explains mixture-of-experts architectures

The figure above shows the architecture behind the Switch Transformer, which uses 1 expert per token with 4 experts in total. Mixtral-8x-7B, on the other hand, consists of 8 experts and uses 2 experts per token.

Why MoEs? Combined, the 8 experts in a 7B model like Mixtral are still ~56B parameters. Actually, it's less than 56B, because the MoE approach is only applied to the FFN (feed forward network, aka fully-connected) layers, not the self-attention weight matrices. So, it's likely closer to 40-50B parameters.

Note that the router reroutes the tokens such that only <14B parameters (2x <7B, instead of all <56B) are used at a time for the forward pass, so the training (and especially inference) will be faster compared to the traditional non-MoE approach.

If you want to learn more about MoEs, here's a reading list recommended by Sophia Yang:

- The Sparsely-Gated Mixture-of-Experts Layer (2017)
- GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding (2020)
- MegaBlocks: Efficient Sparse Training with Mixture-of-Experts (2022)
- Mixture-of-Experts Meets Instruction Tuning (2023)

Furthermore, if you are interested in trying MoE LLMs, also check out the OpenMoE repository, which implemented and shared MoE LLMs earlier this year.

Other Small but Competitive LLMs

Mistral 7B, Zephyr 7B, and Mixtral-8x7B are excellent examples of the progress made in 2023 with small yet capable models featuring openly available weights. Another notable model, a runner-up on my favorite papers list, is Microsoft's phi series.

The secret sauce of phi is training on high-quality data (referred to as "textbook quality data") obtained by filtering web data.

Released in stages throughout 2023, the phi models include phi-1 (1.3B parameters), phi-1.5 (1.3B parameters), and phi-2 (2.7B parameters). The latter, released just two weeks ago, is already said to match or outperform Mistral 7B, despite being only half its size.

Comparison between the 1.3B parameter phi-1.5 model and various 7B parameter models (via the phi-1.5 paper, https://arxiv.org/abs/2309.05463)

For more information about the phi models, I recommend the following resources:

- Textbooks Are All You Need -- the phi-1 paper
- Textbooks Are All You Need II: phi-1.5 Technical Report
- The Phi-2: The Surprising Power of Small Language Models announcement

7) Orca 2: Teaching Small Language Models How to Reason

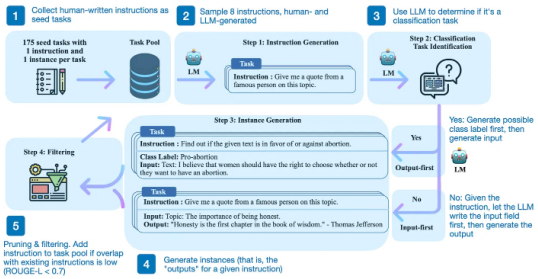
Orca 2: Teaching Small Language Models How to Reason is a relatively new paper, and time will tell whether it has a lasting impact on how we train LLMs in the upcoming months or years.

I decided to include it because it combines several concepts and ideas.

One is the idea of distilling data from large, capable models such as GPT-4 to create a synthetic dataset to train small but capable LLMs. This idea was described in the Self-Instruct paper, which came out last year. Earlier this year, Alpaca (a Llama model finetuned on ChatGPT outputs) really popularized this approach.

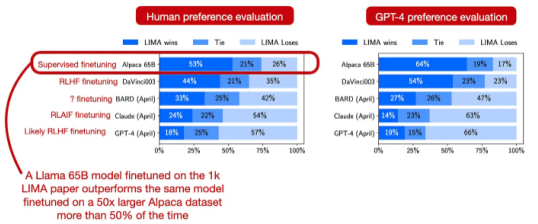
How does this work? In a nutshell, it's a 4-step process:

1. Seed task pool with a set of human-written instructions (175 in this case) and sample instructions;
2. Use a pretrained LLM (like GPT-3) to determine the task category;
3. Given the new instruction, let a pretrained LLM generate the response;
4. Collect, prune, and filter the responses before adding them to the task pool.



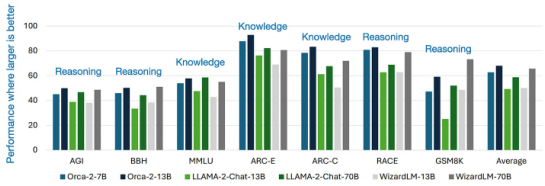
Annotated figure based on the Self-Instruct paper, https://arxiv.org/abs/2212.10560

The other idea may not be surprising but worth highlighting: high-quality data is important for finetuning. For instance, the LIMA paper proposed a human-generated high-quality dataset consisting of only 1k training examples that can be used to finetuning to outperform the same model finetuned on 50k ChatGPT-generated responses.



Annotated figure from the LIMA paper, https://arxiv.org/abs/2305.11206

Unlike previous research that heavily relied on imitation learning to replicate outputs from larger models, Orca 2 aims to teach "small" (i.e., 7B and 13B) LLMs various reasoning techniques (like step-by-step reasoning, recall-then-generate, etc.) and to help them determine the most effective strategy for each task. This approach has led Orca 2 to outperform similar-sized models noticeably and even achieve results comparable to models 5-10 times larger.



A subset of the many benchmarks tasks evaluated in the Orca 2 paper, https://arxiv.org/abs/2311.11045

While we haven't seen any extensive studies on this, the Orca 2 approach might also be able to address the issue of using synthetic data that was highlighted in the The False Promise of Imitating Proprietary LLMs paper. Here, the researchers investigated the finetuning weaker language models to imitate stronger proprietary models like ChatGPT, using examples such as Alpaca and Self-Instruct. Initially, the imitation models showed promising results, performing well in following instructions and receiving competitive ratings from crowd workers compared to ChatGPT. However, more follow-up evaluations revealed that these imitation models only seemed to perform well to a human observer but often generated factually incorrect responses.

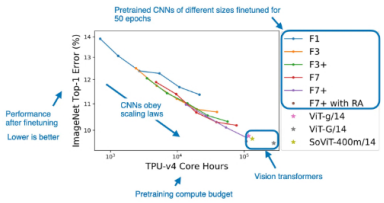
8) ConvNets Match Vision Transformers at Scale

In recent years, I've almost exclusively worked with large language transformers or vision transformers (ViTs) due to their good performance.

Switching gears from language to computer vision papers for the last three entries, what I find particularly appealing about transformers for computer vision is that pretrained ViTs are even easier to finetune than convolutional neural networks. (I summarized a short hands-on talk at CVPR earlier this year here:

https://magazine.sebastianraschka.com/p/accelerating-pytorch-model-training).

To my surprise, I stumbled upon the ***ConvNets Match Vision Transformers at Scale*** paper showing that convolutional neural networks (CNNs) are in fact, competitive with ViTs when given access to large enough datasets.



Annotated figure from ConvNets Match Vision Transformers at Scale (https://arxiv.org/abs/2310.16764) paper

Here, researchers invested compute budgets of up to 110k TPU hours to do a fair comparison between ViTs and CNNs. The outcome was that when CNNs are pretrained with a compute budget similar to what is typically used for ViTs, they can match the performance of ViTs. For this, they pretrained on 4 billion labeled images from JFT and subsequently finetuned the models on ImageNet.

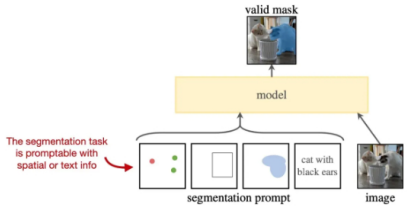
9) Segment Anything

Object recognition and segmentation in images and videos, along with classification and generative modeling, are the main research fields in computer vision.

To briefly highlight the difference between these two tasks: object detection about predicting bounding boxes and the associated labels; segmentation classifies each pixel to distinguish between foreground and background objects.

Object detection (top) and segmentation (bottom). Figures from YOLO paper (https://arxiv.org/abs/1506.02640) and Mask R-CNN paper (https://arxiv.org/abs/1703.06870v3)

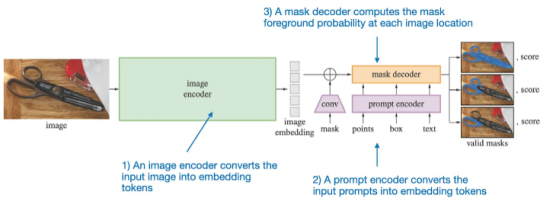
Meta's ***Segment Anything*** paper is a notable milestone for open source and image segmentation research. The paper introduces a new task, model, and dataset for image segmentation. The accompanying image datasets the largest segmentation dataset to date with over 1 billion masks on 11 million images.



The Segment Anything Model (SAM) is designed for efficient, prompt-based image segmentation. Annotated screenshot from the Segment Anything paper, https://arxiv.org/abs/2304.02643

However, what's rare and especially laudable is that the researchers used licensed and privacy-respecting images, so the model can be open-sourced without major copyright concerns.

The Segment Anything Model (SAM) consists of three main components, as summarized in the annotated figure above.



The three main components of the Segment Anything Model via https://arxiv.org/abs/2304.02643

In slightly more details, the three components can be summarized as follows:

1. An image encoder utilizing a masked autoencoder based on a pretrained vision transformer (ViT) that can handle high-resolution inputs. This encoder is run once per image and can be applied before prompting the model.

2. A prompt encoder that handles two types of prompts: sparse (points, boxes, text) and dense (masks). Points and boxes are represented by positional encodings combined with learned embeddings for each prompt type. And free-form text uses an off-the-shelf text encoder from CLIP. Dense prompts, i.e., masks, are embedded using convolutions and summed element-wise with the image embedding.
3. A mask decoder maps the image embedding, prompt embeddings, and an output token to a mask. This is a decoder-style transformer architecture that computes the mask foreground probability at each image location.

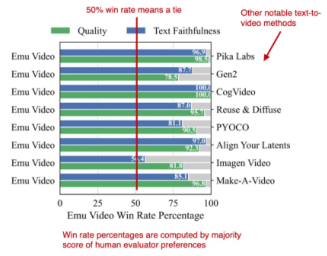
Image segmentation is important for applications like self-driving cars, medical imaging, and many others. In the short amount of 6 months, the paper has already been cited more than 1500 times, and there have already been many projects that have been built on top of this paper.

10) Align your Latents: High-Resolution Video Synthesis with Latent Diffusion Models

Emu Video: Factorizing Text-to-Video Generation by Explicit Image Conditioning is another notable computer vision project from Meta's research division.

Emu is a text-to-video model that can generate entire videos from text prompts.

While it's not the first model for impressive text-to-video generation, it compares very favorably to previous works.



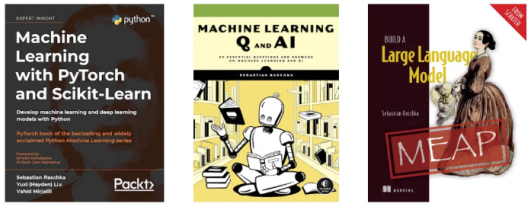
Performance comparison between Emu and other text-to-video models via https://arxiv.org/abs/2311.10709

As the authors note, the Emu architecture setup is relatively simple compared to previous approaches. One of the main ideas here is that Emu factorizes the generation process into two steps: first, generating an image based on text (using a diffusion model), then creating a video conditioned on both the text and the generated image (using another diffusion model).

2022 has been a big year for text-to-image models like DALL-E 2, Stable Diffusion, and Midjourney. While text-to-image models remain very popular in 2023 (even though LLMs got most of the attention throughout the year), I think that text-to-video models are just about to become more prevalent in online communities in the upcoming year.

Since I am not an image or video designer, I don't have use cases for these tools at the moment; however, text-to-image and text-to-video models are nonetheless interesting to watch as a general measure of progress regarding computer vision.

This magazine is personal passion project that does not offer direct compensation. However, for those who wish to support me, please consider purchasing a copy of one of my books. If you find them insightful and beneficial, please feel free to recommend them to your friends and colleagues.



Machine Learning with PyTorch and Scikit-Learn, Machine Learning Q and AI, and Build a Large Language Model (from Scratch)

Your support means a great deal! Thank you!

---

344 Likes · 39 Restacks

344 32 Share

32 Comments

Write a comment...

Sahar Mor AI Tidbits Dec 31, 2023 Liked by Sebastian Raschka, PhD

I'd add the recent Medprompt paper that demonstrated how effective prompting strategies can enable a generalized model like GPT-4 to outperform a specialized fine-tuned model such as Google's Med-PaLM https://arxiv.org/abs/2311.16452

It shows the potential we have yet to explore with such LLMs that can be applied to smaller models as well, substantially boosting their performance at a fraction of the size, cost, and latency.

LIKE (7) REPLY SHARE

1 reply by Sebastian Raschka, PhD

Richard Hackathorn Dec 30, 2023 Liked by Sebastian Raschka, PhD

On the Bloomberg piece... It was confusing to me why Option 3 was different than Option 5. I sense that I am missed a key contrast, perhaps between full-from-scratch-training and fine-tuning. Good practical point about $100 versus $millions. 👍

PS: SUPER!!! Another most-excellent textbook from SR. I got it! Minor note... Your 45% discount was not accepted since Manning already discounts the ebook by 50%.

PSS: You are missing an opportunity with this new textbook. What about a chapter on 'Beyond Language To Multi-Modal'? The term LLM is aging; it should LxM for both pretraining inputs and generative outputs.

LIKE (6) REPLY SHARE

8 replies by Sebastian Raschka, PhD and others

30 more comments...

Top Latest Discussions

Understanding Large Language Models
A Cross-Section of the Most Relevant Literature To Get Up to Speed
APR 16, 2023 • SEBASTIAN RASCHKA, PHD

Understanding ... Attention, Cros...
This article will teach ... architectures and larg...
JAN 14

Practical Tips fo... Adaptation)
Things I Learned From...
NOV 19, 2023 • SEBASTIAN...

See all