Data Science Colle... · Follow publication

Member-only story

10 Papers Every Future AI Engineer Needs to Read

Breaking down the top 10 papers that shaped the field

9 min read · Oct 12, 2025

Marina Wyss - Gratitude Driven Follow

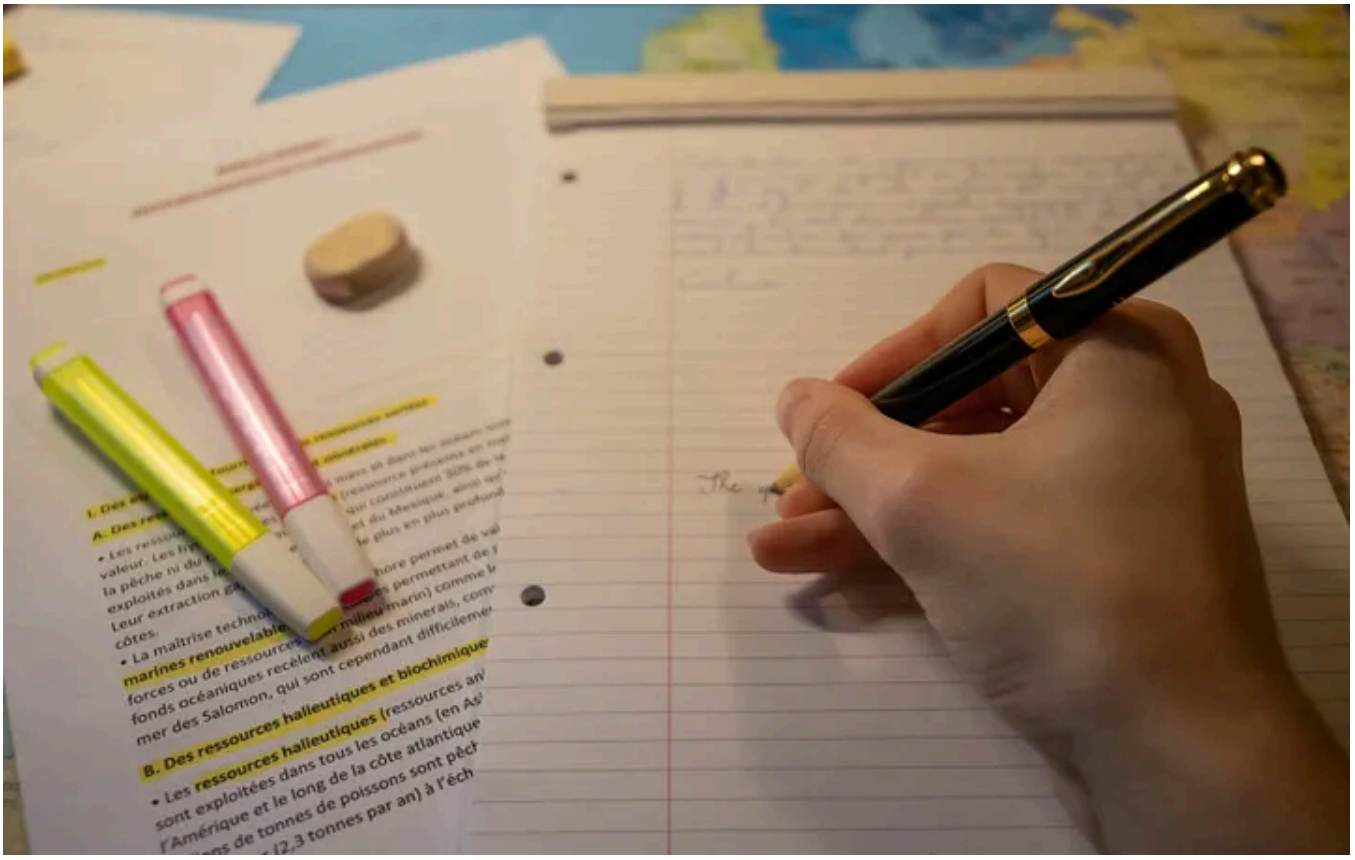

Photo by Benoît Deschasaux on Unsplash

In this post, we're covering the top 10 papers you need to read to understand AI Engineering today.

I'll go over the most important papers that shaped this new and evolving industry, from model architecture to fine-tuning to applications like RAG and Agents. I'll share a quick overview of the paper itself and the most important takeaways, so you will know how these systems work and how we got here, starting from the first breakthroughs all the way to the more recent advancements.

This is exactly the kind of thing you need to know for AI Engineering interviews, so take notes!

1. Attention Is All You Need

When most of us think about AI, we think about large language models, which are a kind of neural network. But neural networks were invented in the 1940s, so why did this become such a big deal in the last couple of years?

One of the main reasons is a breakthrough in 2017. That year, researchers from Google released the famous *Attention is All You Need* paper which introduced the Transformer architecture.

Before Transformers, we relied on recurrent and convolutional networks that processed text step by step. That meant slow training, trouble with long-range dependencies like connecting details that appear far apart in a document and difficulties parallelizing across GPUs, which matters because training speed and cost scale with how well you can split the work across many chips. Transformers replaced that with "self-attention," which lets the model look at all the words in a sentence at once and learn which ones matter to each other. Training became massively parallel, context handling improved, and scale suddenly became more favorable for engineers.

This paper was a big deal. It changed the shape of the whole field, and today almost every modern language model descends from this design.

2. Language Models are Few-Shot Learners

Now a few years later in 2020 we get another big leap with *Language Models are Few-Shot Learners*, the GPT-3 paper. The surprising finding was that if you scale Transformers up enough, they start to perform new tasks from just a few examples

in the prompt. No task-specific fine-tuning required. You can just describe what you want and show a couple of patterns, and the model figures it out surprisingly well.

The team discovered this by training a very large, decoder-only Transformer and systematically evaluating it on many tasks by only changing the text prompt: zero-shot (meaning just instructions), one-shot (one example), and few-shot (a handful of examples). The "breakthrough" wasn't a new architecture or anything. It was the demonstration that scale plus prompting unlocks "in-context learning" meaning that the model can infer a task just from patterns in the prompt.

For practitioners, this reframed how we build systems. Instead of training a new model for each task, we can often prompt a general model to do what we need.

But since then, we've learned that just scaling up forever doesn't solve all our problems. Instruction tuning and reinforcement learning from human feedback made these models far more consistent and helpful.

3. Training Language Models to Follow Instructions with Human Feedback

In 2022, researchers at OpenAI released a paper titled *Training Language Models to Follow Instructions with Human Feedback*, AKA the InstructGPT paper. This paper tried to improve models that respond in unhelpful or toxic ways by tuning the model with reinforcement learning from human feedback.

Basically, you first fine-tune on examples of good behavior (called supervised instruction tuning), then you train a small "reward model" to prefer better answers over worse ones based on human rankings, and finally you adjust the base model to produce answers the reward model prefers.

The main finding from this paper was that a smaller, aligned model could be preferred over a much larger, unaligned one, because it follows directions and respects user intent.

Since then, there have been new advances in the area of model alignment. For example, you'll hear about "DPO," or direct preference optimization, which learns directly from ranked preferences without an explicit reward model.

4. LoRA: Low-Rank Adaptation of Large Language Models

Ok, so even if we have a model that is aligned, it might not perform well on our specific tasks. For example, maybe we want to return responses in a particular

format, or use language from a specific domain like legal or medical texts. Besides in-context learning (AKA prompt engineering), another approach to teaching the model how you want it to respond is fine-tuning.

When we fine-tune, we continue training the model on examples of the behavior we want so it internalizes those patterns. Full fine-tuning updates all the weights, which is time and compute intensive.

In 2021, the LoRA paper gave us a practical way to fine-tune really large models cheaply. Instead of updating ALL the weights, you insert small "low-rank adapters", which are tiny matrices that "nudge" the big weight matrices in a low-dimensional direction, and just train those. You keep the base model frozen, which results in ~10k times fewer trainable params and ~3 times lower GPU memory than full FT on certain setups.

LoRA turned fine-tuning from a research project into something you can do on a single GPU. Over time, we've combined LoRA with quantization (coming up in paper #9) for even bigger savings.

For very complex behaviors, full fine-tuning can still win, but LoRA is the default starting point because it's fast, cheap, and good enough for a lot of use cases.

5. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks

Even once we've fine-tuned, one of the challenges from these models was having limited access to information from outside of their training data. We need some way to be able to access new information from after the training data end date, or something like private company data.

The paper *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks* from 2020 proposed a solution: before the model answers, have it retrieve relevant documents and let the model read them before responding.

This attempts to address two issues at once: old knowledge and hallucination. Instead of relying on whatever the model memorized during pretraining, you can plug it into your internal database or the public web and have it cite what it finds.

This is the pattern used by many production LLM systems today.

Over time, we've learned that retrieval quality is really important. The approaches you use for chunking, indexing, search, re-ranking, and query rewriting often

matter more than the specific base model you choose.

We've also moved from just grabbing the top-*k* best matches and hoping for the best, to multi-step pipelines that iteratively ask better questions and add together information across many sources, with evaluation methods that check faithfulness and cite sources.

6. The Rise and Potential of Large Language Model Based Agents

So now we have these powerful models and we can give them access to real data, but they still don't "do" anything by themselves. That's where Agents come in. I'm cheating a little with our list, because this next paper is a survey (*The Rise and Potential of Large Language Model Based Agents*) but it's a useful summary of the space as of 2023.

The survey lays out a simple way to think about agents:

- Brain: the LLM plans and decides what to do next.
- Perception: the agent reads what's going on — tool results, files, web pages, memory.
- Action: it takes a step — call an API, run a tool, write something — then looks at the result and repeats.

The survey walks through common setups (single-agent, multi-agent teams, and agents working with humans), plus fun ideas like "agent societies" where behaviors emerge from many agents interacting. It also notes the practical stuff that actually makes agents work like clear tool schemas, guardrails to stop runaway loops, and checks to verify the final result. Finally, it covers how to evaluate agents and lists open problems in the space.

7. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity

Let's switch gears to scale and efficiency. The paper *Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity* took the Transformer and made it sparse using a "mixture-of-experts."

Basically, imagine having many specialized mini-networks. These are the "experts." For each incoming token, a small router decides which single expert is most

relevant, and only that expert runs. You still *store* lots of parameters, but you only *use* a small fraction for any one token, so it's faster and cheaper.

This paper showed that conditional computation, or only computing what you need when you need it, can push scale much further than dense models that run every parameter on every token. That matters because it lets you build larger capacity without paying for all of it on every forward pass.

What we've learned since then is that serving sparse models is an engineering challenge because we have to think about things like balancing traffic across experts, keeping latency low, and avoiding bottlenecks. Many teams today choose moderately sized dense models plus retrieval and good tooling, because the total effort is lower.

8. DistilBERT: a distilled version of BERT (smaller, faster, cheaper and lighter)

Now let's think about making models really small. The 2019 DistilBERT paper showed that you can compress a model using "knowledge distillation." This is when we teach a smaller student model to mimic a larger teacher model. Ideally, this helps us retain most of the accuracy of the bigger model but at a fraction of the cost. The paper showed that general knowledge distillation during pre-training results in ~40% fewer params, is about ~60% faster, and retained ~97% of BERT's language understanding.

This matters for things like deployment to edge devices where you have tight latency budgets, limited memory, privacy constraints, or no internet access. A compact model you can run on a phone or small server unlocks a lot of practical use cases.

9. LLM.int8(): 8-bit Matrix Multiplication for Transformers at Scale

Another way to make models smaller is a technique called quantization. In simple terms, quantization stores numbers with fewer bits, like 8-bit integers instead of 16- or 32-bit floats, so the model takes less memory and math is faster. The challenge is doing that without hurting accuracy.

Quantization has been around for a long time, but the 2022 LLM.int8() paper was the first to show a method that keeps transformer performance intact at multi-billion-parameter scale.

It does this by being “outlier-aware.” Their key observation is that a tiny number of “outlier features,” which are individual channels with unusually large activations, especially in attention or feedforward layers, are what break naive int8 quantization. Their solution is to quantize most features at int8, and outlier features at fp16/bf16.

This mixed-precision trick preserves quality while roughly halving memory for the largest components, making single-GPU inference feasible for models that previously needed a small cluster. In practice, it lowers serving cost and accelerates prototyping, often with negligible accuracy loss. It also pairs well with parameter-efficient fine-tuning like LoRA when you need task-specific behavior to be efficient.

10. The MCP Announcement and Docs

And last but not least, I wanted to include MCP in this list, despite the fact that it was not announced via a paper. So, we’re using the announcement and docs instead.

MCP, or Model Context Protocol, is Anthropic’s 2024 open standard for connecting models and the world. The basic idea is that instead of hand-coding a one-off integration for every database, API, or developer tool you want your model to interact with, you run or connect to MCP servers that expose tools, resources, and prompts in a standard schema.

Any MCP-capable client like an IDE, an agent runtime, or a chat app can discover those capabilities, call tools, stream results, and keep shared context.

— — —

So that’s our list! It was hard to pick just 10. There are many other important papers. Some areas we didn’t talk about at all are scaling laws, infrastructure, and system design — there’s a lot going on in this space, as you can imagine. Hopefully this gives you a good starting point to this exciting field.

If you’re feeling like you need some support with your AI/ML career, here are some ways I can help:

- If you’d like to chat 1:1, you can book a call with me here.
- Subscribe to my YouTube channel for weekly videos on technical topics, interviewing strategies, and more.

- Subscribe to my newsletter for a weekly post on a mix of technical topics and mindset/motivation for challenging fields.

Ai Engineering | AI | Machine Learning | Large Language Models | Agents

Follow

Published in Data Science Collective

874K followers · Last published 1 day ago

Advice, insights, and ideas from the Medium data science community

Follow

Written by Marina Wyss - Gratitude Driven

3.9K followers · 11 following

Machine Learning, AI, and Data Science | Productivity with gratitude and purpose. www.gratitudedriven.com
All opinions are my own.

Responses (11)

Bgerby

What are your thoughts?

Elfreda
Oct 16

Thank you for sharing! Some papers are really long and difficult to digest.

I've discovered a super useful tip: I upload the PDF to ChatGOT first and let it summarize the abstract and key contributions for me, or explain any formulas I don't... more

2 Reply

Neurolov
Oct 16

Love this curation.

1 Reply

Ashraf Said
3 hours ago

Your curation of foundational AI papers provides an excellent roadmap for understanding the evolution of modern AI engineering. Starting with "Attention Is All You Need" is particularly apt - the 2017 Transformer architecture truly represents the... more

Reply

See all responses

More from Marina Wyss - Gratitude Driven and Data Science Collective

In Data Science Collective by Marina Wyss - Gratitude Driven

5 Signs of an Inexperienced, Self-Taught Machine Learning Engineer

Sign #3 is a career-killer.

Aug 27 1.7K 36

In Data Science Collective by Amanda Iglesias Moreno

NotebookLM Just Got a Serious Upgrade—Exploring NotebookLM's Newest Features and Updates

What's New and Why It Matters

Sep 29 905 23

In Data Science Collective by Andres Vourakis

AI and the Data Science Job Market: What the Hell Is Actually Happening?

What aspiring, junior, and senior data scientists should know to stay future-proof

Sep 16 1.5K 63

In Data Science Collective by Marina Wyss - Gratitude Driven

I Tried 39 AI Engineering Courses: Here Are the BEST 5

Free/low cost and high quality.

Aug 13 802 19

See all from Marina Wyss - Gratitude Driven

See all from Data Science Collective

Recommended from Medium

Tosny

7 Websites I Visit Every Day in 2025

If there is one thing I am addicted to, besides coffee, it is the internet.

Sep 23 5.3K 190

In Generative AI by Thomas Reid

Google puts another nail in the RAG coffin with URL Context Grounding

Eliminate model hallucinations when processing online data

Oct 2 375 18

Simranjeet Singh

Top 15 GenAI/ML Interview Questions asked in FAANG 2025

Top 15 FAANG interview questions for GenAI, LLM, RAG, ML, explained with practical insights, real examples, and product-focused strategies.

6d ago 124 3

In The Startup by Anirban Kar

What the Smartest People I Know Are Quietly Learning

Because hustle isn't (always) enough.

Oct 2 3.8K 120

In Data Science Collective by Marina Wyss - Gratitude Driven

ChatGPT Made a Machine Learning Career Plan... Would It Actually Work?

You might not want to trust AI on this just yet...

Oct 16 32 2

In AI Advances by Nikhil Anand

I wasted months running slow LLMs before learning this

Why your LLM is running at just 10% of its potential speed

Oct 10 827 11

See more recommendations