

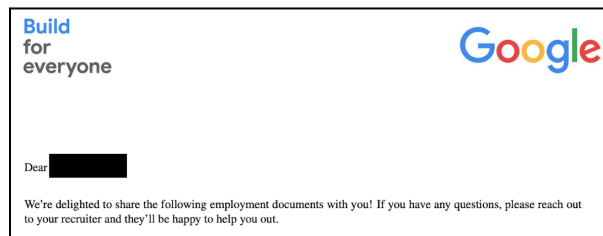


DECEMBER 2024 NEW RELEASE

Last Updated: 12/9/24

How to Ace Your FAANG Interviews

The 2024 Technical Interview Guide for AI Researchers



Re: Offer of Employment

Dear [redacted]

On behalf of OpenAI OpCo, LLC and, where applicable, its designated affiliate or subsidiary ("**OpenAI**"), I am pleased to offer you a full-time position at OpenAI pursuant to the terms and conditions set forth below and in the accompanying exhibit and contingent upon satisfactory completion of a background check. It is important that you understand clearly both what your compensation and benefits are and what OpenAI expects of you. By signing this letter and the accompanying exhibit, you will be accepting employment on the terms recited herein.

40+ FAANG researchers contributed to building the best technical interview resource for CS PhDs going into industry.

If you want to ace your FAANG research scientist technical interviews and secure 3, 4, or 5+ FAANG offers, this 62-page guide is for you!

"It's not the smartest who clears the interviews - it's the most prepared."

- Applied Scientist II @ Amazon

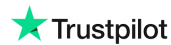


About Rora

Rora is a talent agency for elite researchers.

We've helped 1000+ CS PhDs from CMU, MIT, Stanford, and Berkeley other top universities land (and negotiate) offers from OpenAI, GDM, FAIR, Anthropic, and Microsoft Research and 100+ other companies.

We have 4.9 out of 5 stars on Trustpilot – [read reviews here!](#)



Rora's mission is to empower those who don't see the power in themselves and level the playing field between workers and corporations.

We use our profits to hire and train formerly incarcerated individuals as partners to our Career Agents – providing a higher-quality client experience.



Are you ready to recruit for industry roles?

Use our free diagnostic to evaluate
how ready you are for the job search.



[Get the Diagnostic](#)

Introduction

Welcome to the first-ever Technical Interview Guide for AI Researchers.

Interviewing for a Research Scientist role from your PhD is an opaque, intimidating process.

Why?

#1. Recruiters give generic or sometimes even incorrect answers on what to prepare (ahem, Apple), and there aren't publicly available technical interview guides for Research Scientists like there are for Data Scientists and Software Engineers. Even Leetcode doesn't have many ML questions.

This opacity hurts PhDs and post-docs from performing well in interviews and also leads companies to miss out on candidates who'd be a great fit but accidentally studied the wrong topic (for example - standard data structures instead of implementation details of optimizers and transformers).

#2. We see far too often that PhDs leave interview prep and interviewing until right at the end of their PhD and then are often up against a deadline (either self-imposed or by visa requirements).

This is because interviewing requires an entirely different skillset than your research and most PhDs are recruiting with 4+ years of research experience and 0 years of corporate interview experience. It's almost impossible to do these simultaneously.

Taking the at-the-last-second-prep-approach not only hurts your interview performance because you have less time to study and boost your confidence, but it also hurts your ability to negotiate because you won't have time to line up multiple offers - a huge point of leverage in negotiating.

The timing of interviews and offers is critical. Don't be afraid to delay interviews with certain companies to optimize the process so you're not sitting on an offer for a long time.

Tip from a Research Scientist who secured 8 (!!) offers

"I can't stress enough the value of starting to study before you start interviewing. I blocked a month for talking to recruiters and hiring managers and interview prep and this had a huge impact on my results! By the time I had my Amazon, DeepMind, and Google interviews I was super confident and prepared."

What can you expect from this guide?

Well - if we've done our job - you'll have:

1. A much clearer picture of how to study for your technical interviews



2. Access to a bank of questions you can practice with
3. Clear study schedule you can follow
4. A comprehensive list of the top resources so you can spend more time studying and less time researching
5. Guide on how to prepare for the research talks that most employers expect you give (every once in a while this is the **only** thing your interview entails)

With that - we hope you find this guide helpful! If you did (or didn't) - [tell us here](#). **The first 10 CS PhDs to give feedback will get a \$15 DoorDash gift card.**

Methodology

40+ researchers and FAANG hiring managers generously contributed their interview questions, input, and guidance over the past year.

The researchers who contributed received **more than 100 offers from 30+ companies**, including:



Help us pay it forward! [Submit your interview questions and tips here](#).



Are you ready to recruit for industry roles?

Use our free diagnostic to evaluate
how ready you are for the job search.

[→ Get the Diagnostic](#)

Table of Contents

1. Technical Interviews

- a. Components of AI Researcher technical interviews
 - i. [Amazon](#)
 - ii. [Anthropic](#)
 - iii. [Google DeepMind](#)
 - iv. [Google](#)
 - v. [Meta](#)
 - vi. [Microsoft](#)
 - vii. [OpenAI](#)
- b. Study schedules from 6 FAANG Researchers
 - i. [Study schedule of a NLP researcher who received 8 offers](#)
 - ii. [Study schedule of a CV / Deep Learning / Gen AI researcher who received 3 offers](#)
 - iii. [Study schedule of a ML researcher who received a DeepMind offer](#)
 - iv. [Study schedule of an Algorithms & ML researcher who received 5 offers](#)
 - v. [Study schedule of a NLP researcher who received 6 offers](#)
 - vi. [Study schedule of a ML & Robotics post-doc who received 2 offers](#)
- c. [What a winning solution entails](#)
 - i. [Google Qs & Solutions](#)
 - ii. [Meta Qs & Solutions](#)
- d. [Study resources from 10 FAANG researchers](#)

2. Research Talks

- a. [What to expect](#)
- b. [How to prepare](#)

3. Reverse Interview Questions

4. Question Bank by Company (over 100 questions from Apple, Meta, DeepMind, & more)

5. Appendix - The Study Topics a DeepMind Researcher Recommends

6. Appendix - Netflix Culture Interview Notes

Technical Interviews

One of the most challenging parts of the interview process is knowing what to prepare.

We've heard so many stories of hours spent studying a certain topic only to get no questions on that topic and 30 minutes of questions on a concept you know well but forgot to brush up on.

Below we'll share an overview of what you can expect when you interview at a FAANG company (or Microsoft, OpenAI, etc.), as well as study plans of several Research Scientists who secured top-tier offers.

One tip for prepping for every company you interview with – make sure to understand the company's research areas of focus. You can do this by asking questions in your early interviews, asking friends who currently work there, and even looking for posts, tweets, etc. from technical leaders in the company.

Lastly - before we dive into the interview process for each company - we wanted to address one common question we've heard: *"Will I be asked about AI topics that are important but not necessarily in my research area e.g. how much LLMs does a CV PhD need to know?"*

Typically questions will be focused on your area of research. However, the situation might depend on what type of roles you are looking for, what's your area of research, and what's hot.

For example, if you are a PhD in CV but are looking for Research Scientist jobs focused on multimodal LLMs, you will also get questioned on LLM topics. This is also because LLMs are one of the hottest topics right now.



"Don't I need interviews before I can study for them?"

Researchers who sign up to work with Rora get support landing interviews:

- A resume revamp
- 3-months of LinkedIn Premium
- Networking & outreach templates
- Hiring manager & recruiter names



[See If You Qualify](#)

Components of AI Researcher Technical Interviews - By Company

What's it like to interview with Amazon?

AWS typically conducts 5-7 rounds of interviews, with 2 coding interviews and 1-2 'systems or ML design' interviews, depending on the role.

There's also a research talk and you can also expect a research interview with more in-depth questions on your research. One researcher who received an AWS offer told us that their research interview was called an "ML depth interview" and "it felt like a research interview discussing topics specific to my experience."

The most unique thing about the Amazon interview process is how much culture plays a role in every interview – including the technical ones.

One of the researchers we interviewed who got an offer there said that 'culture' questions could take up anywhere between 30-50% of an interview.

The best way to prepare? Read [Amazon's Leadership Principles](#) and think of 3 different examples for how you embodied each principle. These examples shouldn't overlap, because you might get questioned about each principle multiple times by different interviewers.

What's it like to interview with Anthropic?

The Anthropic coding screen is thought to be extremely difficult. Not leetcode, but rather objected-oriented programming questions.

You need to get 100% correct in order to get to the next round.

We're still building out our content on the Anthropic interview process so check back soon :-)

What's it like to interview with Google DeepMind?

Google DeepMind interviews are different than Google interviews – it's a totally different hiring process.

When interviewing with DeepMind you can expect to have a:

1. Coding interview
2. 1 ML debugging interview
3. A research talk
4. 1 - 2 research interviews (with follow-up questions from your research talk)
5. A people & culture interview (i.e. behavioral interview)



In terms of DeepMind-specific prep - the researchers we spoke to said that the ML debugging interview was the most “out of distribution” interview so it’s worth preparing a bit extra for. Hint - most of the bugs here seemed to fall into the “stupid” rather than “hard” category.

What’s it like to interview with Google?

When you interview for a Research Scientist role at Google, you can expect:

- 2 coding interviews - these are a mix of open-ended questions, design, and Leetcode
 - Researchers say this is one of the most challenging parts of the Google interview process because these aren’t standard interview questions – they are open-ended, without standard interface / test cases.
 - You have to define the problem yourself and write the solution.
 - The interviewer will follow up with multiple variations of the problem, and you'll have to figure out the simplest way to change the code e.g. one line change that will make your solution work for the new variation.
 - Some of the topics covered include graph search algorithms and sampling algorithms.
- 1 hiring manager interview - this is typically fairly informal and casual
- 1 ML basics interview
 - The focus was more on fundamentals. Besides coming up with the solution, can you think about the trade-off of different data structures and algorithms?
- 1 research talk
- 2 research interviews
- 1 culture fit interview

Google Recruiters tend to be extremely helpful and can give you guidance on what to expect in each interview. If they don’t offer this proactively - make sure to ask! And review any docs they share on what to expect in each interview.

However - Google interviewers (hiring managers, peers, etc.) were generally described as intimidating or a bit cold – they don’t spend much time on intros or leave time for a final Q&A - just the interview itself. If you’re not prepared for this, it can catch you off guard.

A final note: historically Google has kept Research Scientist interviews very broad. However - in 2024 they’ve started to make interviews more AI-related. For example - they’ll pick a part of an AI pipeline, like tokenizers, and go deep on it.



What's it like to interview with Meta?

If you interview for a research role at Meta - you're likely to have 6-8 interviews.

The process will likely start with a screening interview with the hiring manager (usually a discussion of your research), followed by 6 "on-site" interviews (these may happen virtually despite the term "on-site"), and a 1-hour research talk.

For the on-site, there's typically:

- 1 coding interview (Leetcode easy to medium - should be relatively easy if you studied CS in undergrad) – "the most frequently-asked questions list on leetcode was on point and had all questions that I was asked."
- 1 ML fundamental interview (e.g. overfitting vs underfitting)
- 3 research interviews (after your research talk, generally more open-ended)
- 1 behavioral interview

A Meta Computer Vision researcher we spoke with recommended spending more time preparing the research and ML fundamental interviews – rather than Leetcode prep.

What's it like to interview with Microsoft?

The Microsoft interview process can be somewhat untraditional.

A researcher we spoke to who received an offer from Microsoft Research didn't even have a coding interview. Instead they were expected to give a research talk and then have subsequent research interviews with people who'd attended your research talk.

These are fairly informal – discussing past work, open topics, and future potential research topics. You can also expect to get some ML design questions related to your research – for example - "how would you apply this topic to XYZ real situation?"

To prepare for these you'll want to have a general understanding of the most popular research topics in the field and the key papers that support each of those topics.

In 2024, NLP (like RLHF, RAG, and even motto-editing and fine-tuning, etc.) has been a popular discussion topic during Microsoft Research interviews.

Another researcher we spoke to said that their Microsoft interview did not have any data structure and algorithm interviews, but they did have an ML design interview that felt very similar to a system design interview.



What's it like to interview with OpenAI?

Here's the OpenAI interview process, from a researcher who interviewed with them in summer 2024:

There were 6 rounds of interviews, including:

- Hiring Manager chat: A brief chat before the actual virtual onsite. Intros, informal chat about research interests, behavior questions, future directions
- Coding: A SWE interview. Not questions you can find on Leetcode. The recruiter will send over some topics to get familiarized with ahead of time. But even if you read the docs, the question is still very hard to prepare for. You basically have to be an expert in the topics.
- ML Coding: A coding question about common topics in ML. It has multiple parts, from simple to hard, including coding and open-ended questions. Having an understanding of numpy and pytorch is helpful.
- ML Debugging: This is similar to the [DeepMind debugging interview](#). In the first part, you'll see a piece of code on some neural network model, and you need to fix all bugs in it. In the second part, you might also be asked to implement some new feature in this model. Familiarity with pytorch is helpful.
- Research discussion: A 2-part interview, first discuss a paper and then your past research experience. The interviewer will send a paper a few days in advance for you to read, and you need to talk about it in terms of the overall idea, method, findings, advantages and limitations, etc. Then you'll discuss your research, the team's research, and potential interest overlaps.
- Team Lead interview: A behavioral interview with the team lead. Mostly "tell me about a time when ..." type of questions. Would also be helpful to get familiarized with the company's vision and mission. Be prepared for "big-picture" questions.

In terms of what stands out about OpenAI's interview process - the coding questions (SWE coding, not ML coding) tend to be very, very hard. If you want to work at OpenAI - you really need to be a coding machine. Unlike other companies, the interview process is much more coding-focused than research-focused.

Additionally - the recruiter will give you a two-sentence description of the specific technical interviews in your interview slate - take those seriously! They are telling you **exactly** what to study for the interview, and they expect you to actually be prepared for that specific interview.



Read every word in the description and know that each word is telling you a topic you should know something about, and think about what kinds of reasonable interview questions they might ask on those topics.

After you have those, take as much time as you need before the on-site to be prepared for those specific topics. Don't be afraid to tell the recruiter you want to schedule the interviews for a specific date a few weeks in the future so that you have time to prepare.



Are you ready to recruit for industry roles?

Use our free diagnostic to evaluate
how ready you are for the job search.



[Get the Diagnostic](#)



Study Plans from 6 FAANG Research Scientists

Your ability to get multiple, competitive offers depends in large part on how you prepare.

Below we'll share the study schedules of 5 FAANG researchers who collectively received X offers.

What did they all have in common?

They treated interviewing and studying like a full-time job and were strategic about when in their PhD they chose to interview i.e. during less research-intensive stretches. They studied for 2 weeks full-time at minimum – doing very little research during that time.

One Researcher we spoke to (who received 8 offers) mentioned that they spent an entire month studying for interviews. They dedicated about 90% of their working time to interview prep and did no coding or new experiments for their research during that time.

How long should you dedicate to studying?

As with anything - the amount of time you need to prepare will depend upon your prior technical skills and experience.

For someone without prior interview experience, you'll be in great shape if you dedicate 200 hours to studying LeetCode and ML coding problems, ML concepts, and relevant research papers.

Once you've done a few interviews and are more confident in your technical interview skills, future interviews can be prepped for much faster.



Study Schedule of a NLP Researcher Who Received 8 Offers (including Amazon, Google, Microsoft, & Databricks)

How long did you study for?

I studied 5 - 6 hours per day for 1 month.

I started to get serious about studying before I had any interviews scheduled. This turned out to be smart because - once I started connecting with recruiters and hiring managers - interviews got scheduled fairly quickly. If I had waited to start studying, I wouldn't have been nearly as prepared.

Btw - it's also worth mentioning how to schedule these interviews -- **do the ones that aren't from your dream companies first, as practice, and then you'll be much more prepared in later ones.** I did an Apple one first and quickly got rejected because I wasn't prepared enough! 😅

How did you decide what to study?

I asked a few people from my lab who were working at FAANG companies what to study, and also looked at Reddit and Quora to see what people shared about what questions they were asked during researcher interviews.

What did you do to study?

Here's how I spent my time studying:

- 2-3 hours a day for coding questions, mostly from "Cracking the Coding Interview"
- 1-2 hours to refresh my ML & DL knowledge (papers, course notes, practice questions, + blogs)
 - This also included ML design questions - where I would create a scenario (ex. A customer wants to solve X problem with ML techniques - how would you design a system and assess performance?)
 - If I was ever stuck on a topic, I'd use ChatGPT to explain it
- About 20 total hours of prep for my research talk
- I also did a sprint at the end to practice answering behavioral interview questions using the [STAR method](#). This ended up being particularly helpful for my Amazon interviews.
- In retrospect - I wish I spent more time on System Design Qs - I probably only did about 3 hours of total prep on this
- If you are interviewing for NLP roles - you should also be prepared for lots of LLM questions and coding questions on Transformers.
- I also noticed I was asked the same ML coding questions across multiple companies:



- How to implement simple neural network and training loop from scratch (sometimes with numpy)
- How to write the attention algorithm ([this blog post was particularly helpful](#))

What should you study if you only have a few hours?

I would do the most classic Leetcode questions, practice my research talk, and refresh LLM knowledge if I'm applying for NLP roles.

What resources did you find the most helpful for studying?

For coding questions: Gayle Laakmann McDowell's book, *Cracking the Coding Interview*

- Plus - [a list of Leetcode questions corresponding with the book](#)

For ML questions: Huyen Chip's book, [Introduction to the Machine Learning Interview](#)

These resources for prepping for System Design interviews:

- <https://www.youtube.com/watch?v=i7twT3x5yv8>
- <https://www.hellointerview.com/learn/system-design/>

Study Schedule of a CV / Deep Learning / Gen AI Researcher Who Received Amazon, Capital One, & Samsung Offers

How long did you study for?

I studied for 2 months – typically 3 - 4 hours per day.

How did you decide what to study?

As a researcher - my main challenge was coding so I decided to dedicate more time there.

What did you do to study?

I solved 200 Leetcode and ML questions and also searched Reddit and TeamBlind for interview questions from the companies I was interviewing with.

I also spent time practicing System Design questions and took a [“Grokking the Machine Learning Interview” course](#).

I used the rest of my time to practice my research talk and stay current on machine learning concepts. To do this - I reviewed the key papers in my field.

1. Start preparing as early as possible by scheduling a practice interview, ideally with a company you're not seriously considering joining.

What should you study if you only have a few hours?

Review core ML/AI concepts. Brush up on key algorithms.



Study Schedule of a ML Researcher Who Received a DeepMind Offer

How long did you study for?

To prepare for my first interview I studied for 3 hours a day for 3 weeks. Then - I would brush up for additional interviews the couple of days leading up to the interview.

What did you do to study?

Coding Preparation

Since I applied for research scientist positions, the coding questions were similar to easy or medium on LeetCode.

To prepare, I solved most easy-level LeetCode problems tagged by company and about 20% of medium-level ones.

To simulate actual coding interviews, I only read the problem descriptions and hide the immediate examples. I find this approach helpful, as it forced me to create my own examples and test cases. It mirrors real interviews, where interviewers verbally describe questions, and candidates ask clarifying questions and write down test cases.

ML Preparation

I reviewed my old university lecture notes on linear algebra, statistics, and machine learning. Since most positions focused on NLP, I studied transformers in detail and coded self-attention implementations.

Communication & Behavioral

I prepared examples for questions such as:

- Tell me about a time you failed at work. What did you learn?
- Describe a challenge you faced. What was your role and the outcome?
- Share a time you disagreed with a coworker, manager, or decision.
- Describe a situation where you had to make a quick decision under a tight deadline.
- Explain when you made an important decision without your boss's approval.

These questions cover most behavioral questions that I received from most companies.

What should you study if you only have a few hours?

From my experience, many research interview questions are about the implementation details of transformers, especially the attention mechanism. Nathan Lambert's blog [here](#) also mentioned this in the "Lightweight interview lessons" section.

Therefore, with only a few hours, I would take some time to review the standard transformers and the attention mechanism because questions about them can come up frequently.



Study Schedule of an Algorithms & ML Researcher Who Received 5 Offers (including Amazon, Google, & TikTok)

How long did you study for?

I studied for 2-3 weeks – about 60 hours total.

How did you decide what to study?

I talked to friends who worked at companies that I was interviewing with – primarily contacts from previous internships or alums from my PhD program.

What did you do to study?

Here's what my studying focused on:

- Pytorch basics, including tensor operations and einsum; some of the [Sasha Rush puzzles](#) would help
- Basic of deep learning ([Sasha Rush has great videos for this](#))
 - I especially focused on backprop and its implementation
- LLM-related questions
 - Transformer architecture, especially the torch implementation of the attention mechanism, and parameter counting: <https://blog.eleuther.ai/transformer-math>
 - Pre-training: tokenization, scaling laws, pre-training objective, perplexity, optimization (Adam) and its memory usage, data and model parallel training (including [Megatron](#)). These can be found in the Llama 3 paper (<https://arxiv.org/abs/2407.21783>) and various blog posts.
 - Post-training: instruction tuning, RLHF & DPO (see their respective papers)
 - Inference: pre-filling vs decoding, kv cache, sampling schemes, speculative decoding, quantization, FlashAttention
 - Some hands-on experience with prompt engineering
- Basic of RL, statistical learning theory, adversarial robustness

What should you study if you only have a few hours?

With only a few hours, I would work out a minimal implementation of transformer and understand surrounding basics about pre and post training.



Study Schedule of a NLP Researcher Who Received 6 offers (including Microsoft Research, IBM, & JPMorgan)

How long did you study for?

I studied for several weeks – about 6 hours a day.

How did you decide what to study?

Initially I focused on foundational concepts in NLP and common topics in technical interviews, and wherever I felt I was struggling, I tried to delve deeper.

What did you do to study?

When preparing for my technical interviews, I focused on both understanding foundational concepts in NLP and practicing problem-solving under timed conditions.

My approach was breaking down key areas like data structures (arrays, linked lists, trees, graphs) and algorithms (sorting, searching, dynamic programming) and tackling each one through a mix of YT tutorials and hands-on coding.

I made consistent use of platforms like LeetCode, HackerRank, and Kaggle to refine my problem-solving skills, while setting aside time for mock interviews to get comfortable thinking and explaining my thought process aloud.

Also, reviewing solutions to identify more efficient approaches and learning to articulate trade-offs were crucial steps in my opinion.

What should you study if you only have a few days?

If I had only a short time to prepare, I would take a highly structured, intensive approach.

I'd begin by reviewing the most commonly asked topics in technical interviews, like arrays, strings, recursion, and basic dynamic programming.

For the first few days, I'd focus on building a quick, deep understanding of these core concepts, especially prioritizing high-yield topics that are frequent in interviews. I would try a series of medium-level problems in each area, spending time on both speed and accuracy.

I set aside time for mock interviews to get comfortable with thinking and explaining my thought process aloud. I practiced with friends who were also preparing for interviews and also used online platforms that offer mock interview services. The questions I used for mock interviews covered common technical interview topics like arrays, strings, recursion, and basic dynamic programming.



Study Schedule of a ML & Robotics Microsoft Post-Doc Who Received 2 Offers

How long did you study for?

I studied for about 90 hours – 4 weekends of 12 hours per weekend, as well as 1.5 hours per evening for about 12 evenings.

What did you do to study?

I spent about 20% of my time doing data structures and algorithms practice questions on Leetcode. I probably did 50-80 practice questions in total, and also memorized how to quickly implement BFS, DFS, etc.

I spent about 30% of my time practicing "ML programming". I implemented simple algorithms in numpy, such as KNN, logistic regression etc. following the examples given here:

<https://github.com/alirezadir/Machine-Learning-Interviews/tree/main/src/MLC/notebooks>. I also implemented some important models in PyTorch (although I wasn't asked to use PyTorch in any interview). I practiced implementing GANs, transformers, CNN, and RNN.

I spent about 20% of my time revising generic ML/data science interview questions (e.g. normalization, regularisation, etc.). I didn't go through any textbooks, and instead went through YouTube videos for any topics I was less familiar with, and made notes.

I spent the remaining 30% of my time revising more deeply technical topics I thought would be relevant to the positions I was interviewing for. I reviewed deep reinforcement learning, transformers and related topics (like different positional encodings), and diffusion models.

For reinforcement learning I quickly skimmed through "Reinforcement Learning: An Introduction" by Sutton and Barto, and went over the most common deep RL algorithms here <https://spinningup.openai.com/en/latest/>. For reviewing transformers and diffusion models I mostly relied on YouTube.

What a Winning Solution Entails

A common question we've heard from 5th year PhDs is - "What should my solution entail? How much detail should I give?" Many researchers are capable of getting the question right – but not all are capable of answering with the appropriate level of detail and context.

Furthermore - some questions (especially 'system design' or 'ML design' questions) don't have just one right answer. Rather, companies are more looking to understand how you think.

How much detail should you give?

We've heard this question a lot from PhDs preparing to go into industry – "I can give the right answer, but how much detail or context do they want me to provide?"

Certainly - if 10 people answer a question correctly - some answers are going to feel "more" correct or impressive to interviews than others.

As a general practice - you should give a quick 1 to 2-minute answer and then ask the interviewer if they'd like you to go into more detail on anything.



"Think out loud when you're answering interview questions -- this way the interviewer can hear your thought process."

Ask clarifying questions

A hiring manager we spoke with said very few people flat-out flunk research scientist interviews. Think about it - most people who are interviewing are in fact researchers from top universities. Many of them have done FAANG internships and/or relevant research.

However - the two factors that separate a good interview from a great interview are:

1. Level of preparation
2. Taking time to truly understand the question

To point #2 - it is hard to answer these questions on the fly. The more time you take to unpack and understand the question - the better chance you have of coming up with a thoughtful answer. This can also lead to more confident answers – something interviewers also notice.

You can ask specific clarifying questions - "could you tell me more about X" - or do what's called "mirroring" where you repeat back the question to check for understanding. "Just to make sure I understand, you said XYZ XYZ." The interviewer can tell you if you're missing a piece of information, and may end up adding more context in response to what you said.



This is ESPECIALLY true for system or ML design questions. In fact - you want to ask as many questions as possible.

This is because the interviewer is actually looking to see that you ask questions – and it will actually reflect negatively on you if you don't.

These questions are designed to be ambiguous - just like many projects or directives in the workplace. Your ability to get a great result relies on your ability to ask questions and simplify the problem.

An interviewer would rather see thoughtful, strategic questions and a mediocre solution rather than a great solution without any questions to demonstrate how you thought about the problem and arrived at the solution you did.

As a Research Scientist who got offers from Amazon, DeepMind, and Google told us - “when it comes to design questions - there is truly no such thing as too many questions.”

For example - let's say you are told to “Build a summarization model.”

You should ask questions like:

- Which kinds of domains are this problem in?
- What's the system interface (expected input and output)?
- What's the availability and scale of the data?
- What's the scale of the system: say, how many users/requests per second?
- What are the most important qualities of the system we're trying to build? (say, accuracy, latency, interpretability, ..., and tradeoff between them?)

Solutions & Feedback from FAANG Researchers

Below you'll find real FAANG interview questions – along with solutions and feedback on those solutions from researchers at the company that the interview question is from.

This goal here is to help you get an understanding of what interviewers are looking for – and how to prepare to not just answer a question correctly, but provide a sufficient level of thought and detail in order to pass the interview.

Google Researcher Interview Questions

Q1: Can you describe how the transformer architecture works? What are the benefits of self-attention?

A1: The most important detail of a transformer architecture is the concept of attention. If we have a sequence of tokens (could be pixels, patches, words in a sentence etc.), imagine applying a fully connected layer to each token independently. In fact, we do this 3 times to get Query, Keys, and Values. The output of the attention layer is:

$$V_{\text{new}} = \text{softmax}(Q \cdot K' / \sqrt{d}) \cdot V$$

Where d is a scalar corresponding to the dimension of the tokens. These are essentially matrix multiplications.

Multi-headed attention leads to richer representations because we apply the attention equation over “chunks” of the QKV matrices independently.

It is critical to note that the output of an attention block is essentially a linear combination of the input tokens; the Q and K matrices compute weights of the linear combination. Softmax normalizes the weights.

Self-attention allows each token to accept information from any other token to enrich its representation. It enables a global context length, avoiding some of the issues with classic RNNs that may forget information far away from the current token. While attention is quadratic complexity in the sequence length, significant effort has been put into making its complexity approx. linear (e.g. sparse attention, axial attention) while maintaining quality.

Cross attention is very common in vision and NLP whereby the Q/K/V can come from different sources.

Positional encoding is used to indicate relative spatial information to the model (otherwise a token that appears in the first index of the sequence can't be distinguished from an identical token that appears at the end).

Feedback from a Google RS: If this is a great answer - what did they do right here?

This an average answer overall, but the answer did the following things right:

(1) mentioning the attention equation, and mentioning positional encoding, multi-headed attention;

(2) correctly mentioning one of the major advantages of self-attention: “Self-attention allows each token to accept information from any other token to enrich its representation”.



Feedback from a Google RS: What's missing from this solution for it to be a homerun?

The solution is missing some details about the transformer architecture, including details of the self-attention equation (for instance, why divide by \sqrt{d}); description of feed-forward layers; description of layer normalization; details about positional embeddings.

Regarding advantages of self-attention, the answer is missing a critical detail — self-attention allows easy parallelization across sequence length during training time, and encoding during inference time.

Feedback from a Google RS: What follow-up questions can they expect to receive?

- a) What is the reason for dividing by \sqrt{d} in the self attention equation? (answer in a footnote in the [Transformers paper](#))
- b) What is the role of the feed-forward layer in transformers?
- c) What are the popular ways of implementing positional embeddings?
- d) What is layer norm used for in transformers, and is it better to do pre-layer norm or post-layer norm?

Feedback from a Google RS: Final thoughts?

This family of questions is very common in interviews for LLM / NLP related roles.

—

Q2: What is the difference between discriminative versus generative models?

A2: Generative models attempt to model a target distribution, possibly with conditions (e.g. GANs, Diffusion models, LLMs). The output of the model is intended to be samples from the training data. In contrast, discriminative models predict useful information from data, such as features, summary statistics, class labels etc.

Feedback from a Google RS: If this is a great answer - what did they do right here?

The answer is correct intuitively, and the answer provides examples of architectures / methods for generative modeling / discriminative modeling.

Feedback from a Google RS: If not - what's missing from this solution for it to be a homerun?

The answer may seem a bit vague / hand-wavy to interviewers. Providing probabilistic definitions may impress interviewers more (generative tries to model $p(x)$ or $p(x, y)$, discriminative tries to model $p(y|x)$).

It would also be useful to provide more precise example use-cases: generative models are used to create images, new drugs, text articles. Discriminative models can be used for classifying images (like those of birds), classifying sentiment of tweets, etc.



Feedback from a Google RS: What follow-up questions could they expect to be asked after this answer?

Could you explain probabilistically, what is the difference between generative and discriminative models?

You mentioned GANs. What is a GAN, how does it work, and what is it used for? Give me details of a common implementation of GANs.

Let's talk more about diffusion models and LLMs. Which models are used for which domains, and why? Can diffusion models be used for language modeling? Why or why not?

Feedback from a Google RS: Final thoughts?

This question will likely be the start of a series of questions where the interviewer will try to drill down and dive deep into one particular technical topic. This could possibly depend on what answer the candidate provides at each step, with the interviewer asking for natural follow-ups that ask for more details / explanations for what the candidate said in the previous answer.

For instance, if the candidate mentions LLMs, the follow-up conversation could lead to a discussion of low-level implementation details of an LLM.

In summary, candidates are expected to know low-level details of everything they say.

—

Q3: What are the different ways/strategies to distribute large-model training onto multiple GPUs? What are the pros and cons of each?

A3: We can do model parallel or data parallel. Data parallel means that the whole model fits on a GPU, and we clone that model onto each GPU, feed an independent batch of samples to each GPU, perform forward and backward passes to get gradients, then synchronize (sum) the gradients across GPUs such that all models have the same weights after updating them. Most deep learning frameworks support data parallelism, it's scalable when the model fits on one GPU, and we don't need to modify the model architecture when scaling. This is most efficient when data batches are balanced across GPUs, and the communication overhead minimized.

Model parallel means we divide chunks of the model onto different GPUs. Forward passes necessarily involve a dependency where information must flow through one GPU before reaching the next; then back propagation must occur on the later GPUs before flowing back to the initial. When the model is too big to fit onto one GPU, we have no choice but to do model parallel.

For large-scale distributed training, systems rely on both principles, and the intent is always scalability, maximizing GPU utilization and minimizing communication bottlenecks. Pipeline Parallelism, Tensor Model parallelism, and ZeRO are more advanced methods that build on these principles.



Feedback from a Google RS: If this is a great answer - what did they do right here?

The answer did a great job at identifying the two major strategies: model / data parallelism. The explanations seem roughly correct. The answer also identified the optimizations done to these paradigms at scale, including pipeline parallelism and ZeRO.

Feedback from a Google RS: If not - what's missing from this solution for it to be a homerun?

I think the answer was pretty good overall. If I had to nitpick, the pros/cons of each method were not very clearly specified.

Feedback from a Google RS: What follow-up questions could they expect to receive after giving this answer?

- a) Could you describe the principles behind pipeline parallelism in more detail?
- b) How would you go about implementing model parallelism in a transformer? What are some natural locations to shard the parameters to minimize the communication bottleneck?
- c) Could you describe any strategies used to speed up inference of large language models?

(see [this paper](#) and [this paper](#) for answers / references on these topics)

Feedback from a Google RS: Final thoughts?

These kinds of questions may be common in more engineering-heavy roles which expect candidates to implement lower-level architecture details.

In addition to Google, NVIDIA also asked a lot of these kinds of questions.



“Don’t I need interviews before I can study for them?”

Researchers who sign up to work with Rora get support landing interviews:

- A resume revamp
- 3-months of LinkedIn Premium
- Networking & outreach templates
- Hiring manager & recruiter names



[See If You Qualify](#)

Meta Researcher Interview Questions

Q1: What are the different kinds of generalization errors in training?

A1: Generalization refers to how well a model performs on unseen test data, as compared with performance on training data.

There are 3 kinds of generalization errors:

[1] Overfitting. High variance - when a model is too complex, it performs well on the training data but poorly on validation/test data. The model may have learned noise or fine-grained details in the training data instead of general patterns.

[2] Underfitting. High bias - the model is too simple to capture underlying patterns in the data. Both training and validation error are high, and the model fails to generalize because it can't even fit the training data. A more complex model or getting more data can help here.

[3] Irreducible error - inherent noise in the data itself, meaning even a perfect model will not capture all the randomness or measurement error in the data.

A large train-validation gap likely indicates overfitting, meaning there is insufficient data or the model is not appropriately regularized. Dropout, L1/L2 weight decay, data augmentation, and early stopping can reduce overfitting.

Feedback from a Meta RS: If this is a great answer - what did they do right here?

This is a great answer. I first expected the interviewee to say “the model is too complex / simple”. It's a plus to explain variance and bias.

—

Q2: Please write down the solution for linear regression.

A2: Given a dataset of pairs $\{x_i, f(x_i)\}$, where each x_i is a N-D point and $f(x_i)$ is a scalar, we aim to find the best-fitting line assuming a linear relationship between the input and the output. For example, when $N=1$, our task is to determine the parameters "m" (slope) and "b" (intercept) that define this relationship:

$$f(x_i) = m * x_i + b$$

To solve this problem, we can represent the system in matrix form. First, we set up matrix "A" to hold the input vectors and account for the intercept "b". Each row of "A" corresponds to a training sample, and we append a column of 1s to incorporate the bias term:

$$A * n = f(x_i)$$

Where:

"A" is the matrix of input vectors (with an extra column of 1s),

"n" is a vector containing the parameters we are solving for ($n = [m, b]$),

" $f(x_i)$ " is the vector of outputs (the target values).

To find the parameters "n", we use the normal equation and compute the pseudo-inverse of "A". If the matrix $A' * A$ (where A' is the transpose of A) is invertible, we can find the best-fit solution:

$$n = (A' * A)^{-1} * A' * y$$

Where:

"A'" is the transpose of matrix A,

" $(A' * A)^{-1}$ " is the inverse of $A' * A$,

"y" is the vector of target values ($f(x_i)$).

This equation gives us the parameters "m" and "b" that best fit the data in the least squares sense. Note that linear regression finds the set of parameters that minimizes the MSE (L2 loss) between the model prediction and GT values.

Feedback from a Meta RS: If this is a great answer - what did they do right here?

This is a great answer. The best-fit solution is something I want to see.

Feedback from a Meta RS: What sort of follow up questions could this person expect to be asked based on their answer?

The best-fit solution only works for linear regression. As a follow-up we'd ask about gradient descent.

Feedback from a Meta RS: Any other feedback to provide?

It's rare for this question to be asked because it requires so much time to answer. I actually only saw this question once during an interview process (at Snap!).

—

Q3: What kind of data distribution is assumed by using L1 and/or L2 training losses?

A3: The L2 loss minimizes the sum of squared differences between model prediction and actual values. It is more sensitive to outliers due to the squaring of errors. L2 loss assumes that the residuals between the predicted and actual values follows an approx. normal distribution with zero mean, errors symmetrically distributed around the mean, and most errors small and larger errors occurring less often.



The L1 loss minimizes the absolute value of the difference between the model prediction and the GT value, averaged over some training set. Because errors are penalized linearly instead of quadratically, the loss is more robust to outliers. The assumed distribution for L1 is approx. Laplace distribution with zero mean, which has a sharper peak and heavier tails than the Gaussian distribution.

Feedback from a Meta RS: If this is a great answer - what did they do right here?

I want to see L1 loss is more robust to outliers.

Feedback from a Meta RS: What sort of follow up questions could this person expect to be asked based on their answer?

The follow-up question would be around how to combine these two losses.

??

“Don’t I need interviews before I can study for them?”

Researchers who sign up to work with Rora get support landing interviews:

- A resume revamp
- 3-months of LinkedIn Premium
- Networking & outreach templates
- Hiring manager & recruiter names

[→ See If You Qualify](#)

Study Resources from 30+ FAANG Researchers

We asked 30+ researchers for their most-used study resources – from coding interview prep to deep learning refreshers.

Coding Interview Prep (*don't forget to practice under time pressure* 🕒):

- Book: [Tech Interview Handbook Coding Interview Study Plan](#)
- Book: [Cracking the Coding Interview](#) (+ [corresponding Leetcode questions](#))
- Videos: [CS Dojo](#)
- Videos: [Exponent](#)

- Github: [Sasha Rush puzzles](#) to practice Pytorch basics, incl. tensor operations & einsum

- Platform: HackerRank's [Interview Prep Kits](#)
- Platform: <https://www.kaggle.com/>
- Platform for mock interviews: <https://interviewme.pro>
- Platform: [Neetcode Roadmap](#)
- Platform: [Leetcode](#)
 - Especially weekly contests for solving leetcode under time pressure

Deep Learning:

- Videos: [Sasha Rush's Basics of Deep Learning](#)
- Blog: [Deep Learning Interview Questions](#)

LLMs:

- Paper: "Attention Is All You Need" - make sure to understand EVERY SINGLE detail of it
- Blog: [Transformer Architecture](#)
 - Common transformer models: decoder-only, encoder-only, encoder-decoder
- [Parallel Training](#)
 - A common question asked is how many gpus you have used?)
- Pre-training: [tokenization](#), positional encoding, data collection & cleaning, [scaling laws](#), pre-training objective, perplexity, optimization (Adam) and its memory usage, data and model parallel training (including [Megatron](#)), long-context LLMs.



- These can be found in the [Llama 3 paper](#)
- *Note - preparing for questions about Megatron was something multiple researchers mentioned!*
- Post-training: [instruction tuning](#), LORA, RLHF & [DPO](#) (see their respective papers)
- Inference: pre-filling vs decoding kv cache, sampling schemes, speculative decoding, quantization, [FlashAttention](#), mixture of experts, RAG
 - <https://www.qualcomm.com/developer/blog/2024/03/quadruple-llm-decoding-performance-speculative-decoding-spd-microscaling-mx-formats>
- Common issues of LLMs: hallucination
- Prompting techniques
- Github: Sebastian Raschka's [LLMs-from-scratch](#)
- Github: Andrej Karpathy's [nanoGPT](#)
- Video: [Andrej Karpathy's YouTube channel](#)

ML:

- Github: Soul Machine's [Machine Learning Cheat Sheet](#)
- Github: [Machine Learning Interview Guide](#)
- Github: Youssef Hosni's [Data Science Interview Qs & Answers](#)
- Github: Ali Rezadir's [Machine Learning Interview Guide](#)
- Platform for ML concept prep & mock interviews: [MentorCruise](#)
- Platform for ML concept prep: <https://www.deep-ml.com/>
- Book: [Introduction to the Machine Learning Interview](#)
- Blog: <https://mlengineer.io/>
- Blog: <https://lilianweng.github.io/>
- Video: [The StatQuest YouTube channel](#)
- ChatGPT / Claude: Technically I used perplxity ai + Claude 3.5 for mock ML interviews since it incorporates some retrieval mechanisms. Specifically it was helpful for practicing linear regression and overfitting. The prompt I used was: *'Imagine you are an interviewer at OpenAI, and this is a ML fundamental interview. What questions will you ask?'*

ML Design:

- Book: [ML Design Interview](#)

??

“Don’t I need interviews before I can study for them?”

Researchers who sign up to work with Rora get support landing interviews:

- A resume revamp
- 3-months of LinkedIn Premium
- Networking & outreach templates
- Hiring manager & recruiter names

[→ See If You Qualify](#)

Reinforcement Learning:

- Book: Sutton and Barto’s ["Reinforcement Learning: An Introduction"](#)
- Blog: Spinning Up in Deep RL’s [Most Common Deep RL Algorithms](#)

System Design:

- Video: ByteByteGo’s [System Design Interview: A Step-by-Step Guide](#)
- Blog: Hello Interview’s [Learn System Design in a Hurry Guide](#)
- Book: Alex Xu’s ["System Design Interview - An Insider’s Guide"](#)

Behavioral + Research Interview Prep:

- Video: Cracking FAANG’s [Behavioral Interview Guide](#)
- Video: FreeCodeCamp’s [Master Behavioral Interviews \(for SWEs\)](#)
- Videos: Exponent’s [Behavioral Interview Course](#)
- Platform: Exponent’s [Top Behavioral Interview Questions](#) & [Amazon Behavioral Interview Questions](#)



- ChatGPT / Gemini: You can use these to condense long stories to short impactful better-conveyed stories
- Blog: [The STAR Method](#)
- Look at your resume + make sure you have a quick pitch for every relevant thing you have done (education, projects, papers, ...). People really like a good story!
- “The focus of behavioral interviews was to assess if I could handle ambiguity. Questions were on if you have worked with a difficult person before, have you ever mentored anyone, have you ever worked on a project that was bound to fail, talk about a failed project and how would you do things differently now.”
- [Book a free mock behavioral interview with Rora :-\)](#)

Other:

- [David Stutz's blog](#) was also super helpful - although it was written for interviews for research internships, the advice applies to full-time positions as well.
- [Algorithms Coach](#) - full time coach, lives in EU so charges less than US-based coaches
- Study + do mock interviews with friends who are also interviewing for accountability!
- Rora's Research Scientist interview bank

Research Talks

The Research Talk (also known as a “tech talk” or “job talk”) is a critical part of every Research Scientist interview. Some companies - like Microsoft Research - may only require a Research Talk and then follow-on interviews (i.e. no coding interviews).

Across the 10 FAANG researchers we interviewed for this interview guide – we heard the same sentiment phrased different ways – “if you know your research thoroughly, the research talk should be straightforward.”

What to Expect

While Research Talk formats may vary from company to company, you can generally expect to be required to give a 40 to 55-minute presentation.

The audience will typically be a group of 8 - 10 researchers at the company you’re interviewing with. However - occasionally the audiences for these can be much larger. One researcher we spoke to gave a Research Talk to two dozen people at Google. Another gave their Research Talk to 30+ researchers at Roblox.

Preparing Your Research Talk

At the highest level - you should prepare for a Research Talk the way you’d prepare to present your research at a conference.

Here’s what the researcher who got 8 different offers (including 4 FAANG offers) - did to prepare for their Research Talk:

“To prepare, I chose two papers that I’m very proud of and that I also felt like were representative of the research I’ve been doing the past four years.

I gave a high-level overview, shared my research findings, and concluded with the big picture implications and future ideas for research.

I also tailored the future ideas for research to the research areas of focus for the company I was presenting to. To figure these out, I asked questions about this in my early-round interviews and talked to a friend currently working at the company. You can also look for posts on LinkedIn from technical leaders in the company, and check out recent research papers that have been published by authors from the company.”



Here's what the researcher who got 3 offers (including one from AWS!) did to prepare for their Research Talk:

"I prepared three versions of my presentation slides for my research (I also included any mentions of notable projects or relevant internship experience):

1. **Detailed Version:** A comprehensive, in-depth presentation that explains each project or research work in full detail. This version is ideal when the interviewer requests a deep dive into a specific aspect of your research.
2. **Short Version:** A concise, 10-slide presentation that summarizes your research in 10-15 minutes. This is perfect for situations where a quick overview is required.
3. **Extended Version:** A longer, 40-50 slide presentation designed for a 45-minute talk. This format works best for group presentations or more detailed discussions.

Also - it sounds small but ensure your slides are clear, visually engaging, and well-structured.

This demonstrates professionalism and shows the hiring manager that you take your research and presentations seriously.

Including mathematical formulas in some slides is a good idea to support your ideas.

Additionally, have a slide deck prepared for any potential follow-up questions.

You can also include a few slides showcasing your collaborations and unique experiences. This can help with self-advocacy and set the stage for future negotiations if you receive an offer.

In my experience, your slides and presentation are what the team will remember. Making a strong impression can significantly influence the team's hiring vote."

Q&A

It's common that after each Research Talk there will be about 5 - 10 minutes reserved for questions. This can be the most intimidating part for many researchers since you can't anticipate exactly what they will ask. (Note - this is different from a dedicated [research interview](#) - which is common at most companies)

However - since most people in the room won't have seen your research before - they will often think of the most obvious next step. So - think back to past research presentations, every conference you presented at, etc. - what were the questions you were asked during Q&A. List those out and either answer them in your core presentation or prepare slides to use during Q&A.



Here are the questions the researchers we spoke to most commonly received during the Q&A section:

- “What would you have done if not _____?”
- “Why did you make _____ decision?”
- “What happens when you apply the same technique on an _____ domain?”
- “What is follow-up you could’ve done?”
- “What if you used xxx instead of yyy?”

“Potential Next Steps / Follow-Up Ideas” is a very common category of questioning during the Q&A section of the Research Talk.

Several researchers we spoke to recommended having a slide deck with all the possible follow-up ideas and next steps they could think of, as well as answers to commonly asked questions. One researcher even did a mini-experiment and outlined the results for the most common follow-on ideas.

Another Google researcher we spoke to (who also had Bloomberg and Allen AI offers) said that everytime someone asked them a question about their research during the interview loop, they added it to their slide deck so they would be more prepared for their next Research Talk.

Tip: Research Your Interviewer To Find Out What You’ll Be Asked

Typically, if it's an AI research interview, you'll know the name of the interviewer before the interview (if not - ask your recruiter for it). Look up their Google Scholar and read a few of their recent papers.

Not only will this help you think of questions you can ask them - but you can also get a sense of what they'll ask you. If they're an expert in the same area as you - you can keep answers to questions about that short. If they're less familiar, you should spend slightly more time to explain the high-level ideas.

FAQ

“How’d you keep up with the latest research so you were prepped for research interviews?”

*“I set **Google Scholar alerts** for specific keywords, researchers, and topics of interest.*

*I **read surveys / reviews** - these summarize recent developments & often identify trends & directions for future research.*

And, of course, conferences were also helpful (both in-person and virtual!). Many conferences also share recorded talks or proceedings online.”

- A researcher who received 3 FAANG offers

Impress Interviewers by Asking These Questions

At the end of every interview you'll typically have a chance to ask the interviewer questions. Not only is this a great way for you to learn about the company, team, and role (remember - you are interviewing the company, too!) – but it's also a way to demonstrate critical thinking and wanting to understand the broader business.

You'll want to tailor the questions to the person you're interviewing with – for example, a hiring manager, a peer, a skip-level manager, etc.

Understanding problems

- What research problems is the team working on?
- What are the biggest challenges the team is facing right now?
- What are the biggest challenges our team will face in the next 2 years?
- Where could I add the most value?
- What's something immediate that I could support you on or take over for you?
- What are examples of how direct reports haven't been successful working for you?
- Who are individuals that I should go to when you are busy or to save you time?
- If you had to imagine right now what I'd be weak at or need to improve, what would it be?
- If you had to imagine what my biggest blockers would be what would they be?

Understanding goals

- Where do you think the company is headed in the next 5 years?
- What milestones / achievements have been shipped from the team?
- What outcomes are most important for me to achieve in the first year?
- How do you want to measure my success/progress?
- What's your long term vision for this team?
- What are your most successful direct reports doing?
- How are your most successful direct reports operating?
- What are your expectations for me in the first year?

Understanding work style/preferences

- Who is someone you had a great working relationship with that reported to you and why did it work so well?
- How quickly do you expect responses to be from your requests?
- What are your typical work hours and preferences so I can adapt to this?
- What are your preferences on meetings vs. asynchronous communication?
- How do you like to do 1:1s? How often?
- What are you comfortable with me working on independently, and what would you like more visibility on?
- How do you choose projects? top-down or bottom-up?
- What's the typical cycle of a project? Can research team take risks and focus on long-term projects?
- How large is the team and what's the structure?
- Which stakeholders will a team member interact with?
- What are typical working hours?

Personal experiences

- Why did you decide to join this company over other companies?
- What's your typical day like?
- What are your favorite and least favorite things about working here?

Visibility and Growth

- How core is the team's work to [Company]'s business agenda? Why?
 - What products are supported by the team's work? How important is it?
 - What are the most important metrics for these products?
- What's the relationship between product and research? Do you publish papers and open-source code?
- What's the career path / growth trajectory like for this role?

Questions for startups

- What are your target customer groups? What are their pain points?
- How big is the market?
- What's the company's value proposition?
- What milestones / achievements have been shipped?
- How many customers does the company have now? What are they?
- Has the company reached product-market fit? why?
- What's the company's yearly revenue?
- What's the team organization?
- Who are the company's competitors?
- What are the unique advantages of the company?
- What are the biggest challenges and risks?
- Does the company have technical barriers?
- Which series of funding are you in? What's the current valuation?
- What's the company's long-term vision?
- How difficult is it to scale?
- What value can I bring to the company?

Other Good Resources For Questions

- [Reverse Interviewing Github Repo](#)
- [How to query a company's culture](#)
- [How to Reverse Interview](#)
- [Blog post on reverse interviewing from engineering manager](#)
- [40 Best Questions to Ask in an Interview](#)



At Amazon - 50% of every interview consists of behavioral interview Qs.

Don't overlook behavioral interview prep!



Sign up for a [Free Mock Behavioral Interview](#)

Question Bank

- [Amazon](#)
- [Apple](#)
- [Bloomberg](#)
- [Cohere](#)
- [Google DeepMind](#) (GDM)
- [Google](#)
- [Meta](#)
- [Microsoft](#)
- [NVIDIA](#)
- [Tesla](#)
- [TikTok](#)
- [Waymo](#)

Amazon

1. Explain different parallelism strategies for LLM (data parallelism, model parallelism).
2. Explain the basic architecture of the transformer. How does computation grow with sequence length?
3. What would be the challenges when integrating a vision model into a transformer? (e.g. the vision model is much denser in computation but smaller in #params).
4. Explain the basic architecture of the CLIP model.
5. High-level strategies to optimize LLM systems (training/inference, multi-node/single node, average resource/peak resource)?
6. What optimizations do ZeRO and FSDP have?
7. How would you model user response time and why?



At Amazon - 50% of every interview consists of behavioral interview Qs.

Don't overlook behavioral interview prep!



Sign up for a [Free Mock Behavioral Interview](#)

8. Describe bagging.
9. How to handle class-unbalanced data?
 - a. *Note: The interviewer wanted to hear metrics like F1 score, false positive, false negative, etc, together with their interpretations.*
10. Describe the differences between pretraining and instruction-following training in LLM.
 - a. *Note: I talked about the difference in masking.*
11. What metric should you watch during pretraining of an LLM?
 - a. *Note: The interviewer wanted to hear perplexity, and the definition of that.*

12. [Robotics] Describe a time when you were faced with an ambiguous problem with many solutions. Talk me through how you decided the right course of action. What was the outcome of these decisions, and how did the stakeholders react?
13. [Just Walkout Team] Can you briefly explain mixed precision training, focusing on how data type conversions occur for model weights and gradients during both forward and backward passes?

Apple MLR

1. Write K-Means clustering in Python. Lots of optimizations/vectorization questions after that.
2. Write pseudo-code / code for logistic regression.
3. Write LSTM equations, or design a long memory RNN from scratch.
4. Some questions about CNN filters and how they work.
5. Write Adam optimizer equations, or design an adaptive learning rate optimizer from scratch.
6. Explain pre-layer norm vs post-layer norm, and explain why one is better than the other in terms of gradient flow.
7. Explain mathematically why the self-attention equation has a divide by \sqrt{d}
8. Some modeling/design questions (how would you design a network to solve some speech recognition problem), nothing too complex
9. Generate a random mask given the following constraints: seq_len, num_segments, total_mask_tokens

mask tokens are represented by 1, and non-mask tokens by 0 segments start at 0 and end at 1

seq_len=12, num_seg=3, total_mask_tokens=5 Valid output: [0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 1, 1]
The three segments are: [0, 0, 1], [0, 0, 1, 1], and [0, 0, 0, 1, 1]

Generate the segment with a fully vectorized code.. i.e. no loops

Bloomberg

1. Design a logistic regression classifier (with pseudo code) for some 5-way classification task on Bloomberg data.



- a. *Note: Lots of follow up questions on imbalanced classes, data scarcity, F1 score, classification metrics, optimization, minibatching.*
2. Describe (with code / pseudo code) a non-perceptron based ML algorithm, like kNN / random forests / PCA / clustering
3. Three Leetcode-style medium questions on Bloomberg specific data, mostly involving chat transcripts. They involved BFS, Heaps, Queues (one of them per question).
 - a. *Note: Very similar to leetcode medium (probably some direct overlaps too).*
4. One more NLP-heavy question: cluster conversations in a cocktail party style chat transcript.
 - a. *Note: Sounded very similar to coref resolution to me, just replacing entities with individual messages. Since I said something about BERT in my answer, there were lots of followup questions on the specifics of BERT.*
5. Can you explain the transformer encoder block in detail?

Cohere

This next stage will consist of 3 x 45-minute technical interviews, details are as follows;

1. Live problem-solving exercise that involves language models (45 mins) In this code-pairing interview, you will screen-share a Google Colab Python notebook while implementing an evaluation task for a large language model. You'll be given a link to an academic journal article that describes the task, and the Python notebook will contain some code and comments that contextualize what you need to implement. In this interview, we are looking to understand your applied research, programming, and technical communication skills. You'll have the option to use Pytorch or Tensorflow 2; please let myself and Maria know your preferred choice.
 - a. Write code to evaluate the accuracy of a language model using a dataset that contains prompts, several possible completions per prompt, and a label indicating which completion is correct.
2. Solution Design (45 mins) You will be asked to describe an NLP solution to a particular problem. The open-ended problem will not have a single correct answer. In this interview, we are looking to gain insight into your thought process. Please walk the interviewer through your approach to pragmatic solutions and trade-offs between competing ideas. The interviewer may follow up for depth, clarity, and potential considerations. Feel free to utilize a drawing tool or a doc to share/organize/present your ideas). An example of a



question you may encounter is "Please describe your approach to creating a model that can pass the SAT/LSAT/etc."


- a. "Design a large-language-model based system that can access factual information from after the time that it was trained."
3. Machine Learning Concepts (45 mins) In this conversation, the interviewer will explore your familiarity with (modern) machine learning concepts. You will cover a range of topics such as Basic probabilistic concepts, Optimisation, Reinforcement Learning and more.
- a. What is Bayes Theorem?
 - b. What is its relevance to Machine Learning?
 - c. How can you measure the similarity of two probability distributions?
 - d. Why is KL divergence not technically a metric?
 - e. Explain what overfitting is.
 - f. Describe some techniques for regularization (especially dropout).
 - g. Describe the transformer architecture:
 - i. What are transformers used for? How do they differ from traditional RNNs?
 - ii. Describe self-attention.
 - iii. Describe positional embeddings.
 - iv. What is the difference between encoder/decoder and decoder-only architectures?
 - h. What are the stages for training a typical Large Language Model?
 - i. What are some issues with existing LLMs?



"Don't I need interviews before I can study for them?"

Researchers who sign up to work with Rora get support landing interviews:

- A resume revamp
- 3-months of LinkedIn Premium
- Networking & outreach templates
- Hiring manager & recruiter names

 [See If You Qualify](#)

DeepMind

1. What are the different ways/strategies to distribute large-model training onto multiple GPUs? What are the pros and cons of each?
2. What type of auto-encoder model is generative? Follow-ups: - Explain how a VAE works and differentiate with respect to a standard AE? - What are the two distributions you are taking the KL divergence in-between?
3. Can you describe how the transformer architecture works? What are the benefits of self-attention?
4. What do you know about DeepMind, and why are you interested in joining?
5. Are you familiar with RL?
6. How do you keep up with current research?
7. Describe a time when you changed your opinion when working on a team.
8. When does matrix inverse exist?
9. What are some matrix inversion methods?
10. What is the runtime complexity of Gaussian elimination?
11. When might matrix inversion have problems with computer implementations?
12. How could you use more memory to mitigate numerical issues?
13. What is a Taylor series?

Google

1. Write Python code to train and do inference from a bigram language model for next word prediction. Some follow-up inference optimization questions (needing binary search).
2. What is the defining characteristic of eigenvalue/eigenvector pair of a symmetric matrix?
3. Define the least squares regression problem.
4. How do you define the Bellman equation for the optimal policy?



5. When you have noisy data and you have a dataset, what is the goal, or what is the actual criteria that you are trying to optimize in Least Squares? Or what is the objective?
6. Prove that eigenvalues of a symmetric real-valued matrices are non-negative.
7. A leetcode-hard dynamic programming question: exact copy of <https://leetcode.com/problems/student-attendance-record-ii..>
8. What is the difference between discriminative versus generative models? What is the difference between logistic regression and naive Bayes? What happens when the labels are perfectly linearly separable in logistic regression?
9. What is the difference between discriminative versus generative models? What is the difference between logistic regression and naive Bayes? What happens when the labels are perfectly linearly separable in logistic regression?

Statistics position:

1. How to test for differences in preferences between two groups of annotators given some specific type of data
 - a. Data: 100 examples, each group has 10 annotators annotate a randomly chosen subset of 20 (out of 100) examples
2. Given this data, how to find “noisy” annotators that should ideally be removed from the data before use for downstream purposes
3. Given this data, what to measure to decide

Meta

1. What are the different kinds of generalization errors in training?
 - a. *Note: You aren't expected to know each by name, they prompt you: e.g. approximation error, optimizer error. You just need to give high-level explanations of each.*
2. How would you merge K-sorted lists? If given K wooden sticks of variable length, how would you compute the maximum integer length of smaller sticks that you can obtain if you are given a target number M for the number of smaller sticks required? How would you simplify a directory path?
 - a. *Note: These were coding questions testing data structure and algorithm knowledge*



3. Please write down the solution for linear regression.
4. What kind of data distribution is assumed by using L1 and/or L2 training losses?
5. [system design] How would you design a system for a social intelligent avatar?
 - a. *Note: For this kind of system design questions, the thumb of rule is: ask the interviewee for clarification: what kind of functions do you require for the system?*
 - b. *Then answer from high-level to low-level. First start with high-level design ideas: what's input/output of the system, how many components the system should include, and what's the general function for each component.*
 - c. *Then go into the details of each component (low level): what's the input/output of this module, how would you design the model, etc.*

Behavioral

1. Describe a time you anticipated a major shift in the industry. How did you stay ahead of that shift and how did you manage the change that came along with the shift?
 - a. What allowed you to be in a position to anticipate that shift in the industry or strategy?
 - b. Did you face any resistance? Anyone not willing to accept that change?
 - c. How did you bring people along as part of that change?
 - d. How did you know if you took the right approach? Tell me about adjustments you had to make and why.
2. Describe a time when the use of data or the insights from an analysis was critical to your ability to take a specific action?
 - a. What specific data or insights did you use to make this decision?
 - b. How did you ensure the data quality and the accuracy of the analyses?
 - c. What have you personally done to ensure you are basing decisions based on data?



At Meta you can expect to receive at least 10 behavioral interview Qs.

Don't overlook behavioral interview prep!



Sign up for a [Free Mock Behavioral Interview](#)

3. From start to finish, walk me through your experience leading an important project. Ideally, something where you had to create something from scratch (e.g., a “whitespace”) or turn around something that was failing.
 - a. What was the impact you set out to have with the project and what impact did it actually have? What surprised you and why?
 - b. What aspects of the project required you to create something from scratch or turn something around that was failing?
 - c. What was your approach to getting the work done?
 - d. What other approaches did you consider? Why did you settle on this one?
 - e. Who disagreed with your strategy or approach? How did you resolve that?
 - f. How did you approach getting others on board and how did you keep things on track?
 - g. What was the biggest challenge in the project? What did you do to help the team overcome it?
 - h. If you were to go back in time and lead this project all over again from the beginning, what would you do differently?
4. Why did you decide to do a PhD?
5. Give an example of where your project failed.
6. Give an example where you had to deal with a difficult person.
7. Give an example of when you had to mentor someone.
 - a. What were some difficulties you encountered in your mentoring experience?

Microsoft

1. Implement a VAE (pseudocode is enough).
2. What is p value? What is A/B testing? How do you go about selecting samples for A/B testing? How do you address the problem of missing observations? What would you do if you noticed a difference in model performance on the offline metrics and the online metrics?
3. What is the difference between encoder and decoder language models?

4. What is variational auto-encoder?
5. Write the expression for least-square regression in Pytorch and gradient descent update rule.
6. Describe the architecture of BERT.
7. Coding Q - Implement multi-head self-attention.

NVIDIA

1. Write Python code to do data deduplication.
2. Write Python code to do sampling from a huge dataset (after some tokenization + preprocessing). Some optimization questions after that.
3. Why is a decoder LLM better than encoder-decoder LLM?
4. If there is a one layer network with a million weights trained on a billion data points (till lowest possible training loss), would it underfit or overfit?
 - a. *Note: some follow up on Chinchilla laws related to repeating data vs new data (interviewer didn't know the answer he said it's speculative question)*
5. There is a neural network with 1 CNN layer with a causal filter of size 2 followed by a 1 layer transformer. There is no position embedding in the input. Does this model know positional information, and if so to what extent?
6. What is the difference between multi-query attention and multi-headed attention?
7. Explain the effect of the per-timestep loss weighting function in the diffusion objective. What weightings work well for images? What is a domain where a different weighting might be better?
8. What do you believe will be the most important applications of generative modeling over the next 5 years?
9. Does python have abstract class?
10. Explain the difference between multi head attention vs single head attention.
11. If you manage to reconstruct 3D human motions from Internet videos, and want to train a human motion model with this reconstructed motion data. However, your estimated motion can be noisy, how do you deal with this during training?

12. Given two rectangles, find the area (they can be overlapping).

a. *Note: this is on leetcode <https://leetcode.com/problems/rectangle-area/>*

13. How can we build safeguard around LLMs?

14. How does flash attention work?

15. Given two rectangles, find the area (they can be overlapping)

a. *Note: I believe this is leetcode 223*

16. How do you submit a slurm job?

17. [system design] If you want to train a human motion generation model from scratch, how would you design a system for this?

a. *Note: For this kind of system design questions, the thumb of rule is: ask the interviewee for clarification: what kind of functions do you require for the system?*

b. *Then answer from high-level to low-level. First start with high-level design ideas: what's input/output of the system, how many components the system should include, and what's the general function for each component.*

c. *Then go into the details of each component (low level): what's the input/output of this module, how would you design the model, etc.*



“Don’t I need interviews before I can study for them?”

Researchers who sign up to work with Rora get support landing interviews:

- A resume revamp
- 3-months of LinkedIn Premium
- Networking & outreach templates
- Hiring manager & recruiter names

 [See If You Qualify](#)

Tesla

1. What is your least favorite part of Python and why?
 - a. *Note: It is more difficult to state dislikes rather than likes for your primary coding language. They check what you dislike and see if you have a technical understanding of Python concepts or just a superficial one.*
2. How would you use your research project <X> to create a hardware product?
 - a. *Note: They want to understand if you can balance the research v/s product mindset, coming out of grad school.*
3. **Tesla (AutoPilot).** Solving the card game 'set'. First, provide a naive $O(n^3)$ complexity, and then exploit the problem structure to provide $O(n^2)$ algorithm.
 - a. *Note: Needs permutation and enumeration-based technique. The key to the second half of the question is that given the first 2 cards in the set, there is only one option for the third card and no search is needed.*
4. **Tesla (AutoPilot).** Implement K-means in Python. First a basic implementation with for loops. The second/main question is to implement it in a vectorized form using NumPy broadcasting (there should be only one for loop for the reassignment of points after the new centers are computed.)
 - a. *Note: They also care about documenting code side-by-side about the dimensions of the matrices being used in the NumPy operations and the resulting shapes. Additionally, take care of division by zero, and means for clusters that have no points assigned to them.*
5. **Tesla (OptimusBot).** A. If a bot is doing a random walk with probability transition p on a graph, what would be the probability of it reaching a goal state/node? B. Write a code to compute this probability.
 - a. *Note: Take care of ergodicity/recurrence and the starting position of the bot on the graph. Can make use of the NumPy libraries to compute the eigenvectors.*
6. **Tesla (OptimusBot).** Given a 1D chain with left and right transition probability and the reward function, implement tabular (a) value iteration and (b) q-learning. Discuss the limitations of Q-learning.
 - a. *Note: You need to be careful with the stopping criteria.*

7. **Tesla (OptimusBot).** 3D geometry: Translate coordinates of an object in one ego position, to the frame of reference of another agent's ego position. (a) In 2 dimensions, (b) In 3 dimensions.

TikTok

1. [E-Commerce Recommendations / Tiktok Shop] Can you explain different types of optimizers used in Deep Learning.
 - a. *Note: This question had extreme back and forth discussions comparing different types of optimizers, how the role of exponentially weighted averaging works, momentum, etc. Basically an entire run through from basic vanilla gradient descent to Adam optimizer. I was grilled on all the variables present in the equations of all these optimizers.*
2. [E-Commerce Recommendations / Tiktok Shop] What is the role of temperature in contrastive losses like InfoNCE and NT-Xent?
 - a. *Solution from a Researcher who got a TikTok offer! The temperature in contrastive losses controls the scaling of logits, helping to adjust the sharpness of the softmax distribution, making it easier to discriminate between positive and negative pairs. A lower temperature increases the penalty for dissimilar pairs, leading to stronger contrast between them.*
3. Implement a Linear Regression with SGD

Waymo

1. Is there a way to improve the efficiency of the MLP in transformers.
 - a. *Note: They're looking for the answer to be mixture of experts*
2. Implement k-means clustering.
 - a. *Note: Implement in numpy*
3. Implement logistic regression in numpy.
 - a. *Note: Implement stochastic gradient descent - similar to how it is done for linear regression*
4. How would you improve the inference speed of a large transformer?
 - a. *Note: Distillation was the answer they were expecting.*

A DeepMind Researcher's Suggested Study Topics

We asked a DeepMind Researcher what topics you should study if you're preparing to interview with DeepMind. Below is the topic list she suggested.

Computer Science

1. Data Structures
 - a. Hash Table
 - b. Linked List
 - c. Graphs
 - i. Adjacency List
 - ii. Adjacency Matrix
 - iii. Pointers and Objects
 - d. Heap
 - i. Fibonacci Heaps
 - ii. Priority Queue
 - e. Binary Tree
 - f. Binary Search Tree
 - i. Balanced Binary Search Tree
 1. Red Black Trees
 2. AVL Trees
 - g. Queue
 - h. Stack
 - i. Dequeue
 - j. Arrays
 - k. Disjoint Set
2. Algorithms (Non-ML)
 - a. Graph Algorithms
 - i. Shortest Path
 1. Dijkstra
 2. Floyd-Warshall
 3. Bellman-Ford
 4. A*
 - ii. Search
 1. BFS
 2. DFS
 - a. Topological Sort
 - b. Strongly Connected Components
 - iii. Minimum spanning Tree
 1. Kruskal
 2. Prim

- iv. Min-Cut / Max-Flow
 - 1. Ford-Fulkerson
 - 2. Maximum bipartite matching
- b. Recursion
 - i. Fibonacci
- c. Dynamic Programming / Recursion
 - i. Knapsack
 - ii. Traveling Salesman Problem
 - iii. Longest Common Subsequence
 - iv. Rod Cutting
 - v. Matrix-chain multiplication
 - vi. Optimal Binary Search Trees
- d. Divide and Conquer
 - i. Maximum-subarray
 - ii. Strassen's Algorithm
- e. Greedy
 - i. Huffman Codes
 - ii. Matroids
 - 1. Task-scheduling
- f. Sorting
 - i. $O(n \log(n))$
 - 1. MergeSort
 - 2. Quicksort
 - 3. Heapsort
 - ii. $O(n)$
 - 1. Radix Sort
 - 2. Bucket Sort
 - iii. $O(n^2)$
 - 1. Insertion Sort
 - 2. Selection Sort
 - 3. Bubble Sort
 - 4. Shell Sort
- g. Multithreaded
 - i. Multithreaded Matrix Multiplication
 - ii. Multithreaded MergeSort
- h. Linear Programming
 - i. Simplex
 - ii. Branch, Bound and Cut
- i. Fast Fourier Transform
- j. String Matching
 - i. Rabin-Karp

- ii. Knuth-Morris-Pratt
 - k. NP Completeness
- 3. Testing
- 4. Programming Languages
 - a. Functions
 - i. Passing arguments by value / reference
 - ii. Main: Handling command-line options
 - iii. Return types and the return statement
 - iv. Overloading (Differences in the input parameters determine the function called)
 - v. Polymorphism (Different behavior depending on class / type)
 - vi. Default Arguments
 - b. Types
 - c. Variables
 - i. Val vs. Var
 - d. Expressions
 - i. Order of Evaluation
 - ii. Logical and Relational Operators
 - iii. Assignment
 - iv. Increment / Decrement Operators
 - v. Conditionals
 - vi. Type Conversions
 - vii. Implicit / Explicit Conversions
 - e. Scope
 - f. Constants
 - g. Pointers, Arrays, References
 - h. Compilation
 - i. Namespaces
 - j. Error Handling
 - k. Regular Expressions
 - l. Iterators
 - m. Predicates
 - n. Resource Management
 - o. Garbage Collection vs. Reference Counting
- 5. Concurrency
 - a. Tasks and Threads
 - b. Passing Arguments
 - c. Sharing Data
 - d. Waiting for Events
 - e. Communication Tasks
- 6. Object Oriented Programming

- a. Classes (c++)
 - i. Concrete Types
 - ii. Abstract Types
 - iii. Virtual Functions (Polymorphism)
 - iv. Class Hierarchies
 - v. Copy and Move
 - vi. Constructor
 - b. Objects (Instances of classes, determining their type)
 - c. Mixins
 - d. Inheritance
 - e. Data structure framing of programming rather than logic / action based framing
 - f. Immutable State
7. Distributed Computing
- a. Map-Reduce
 - b. In-Memory Compute
 - c. How to parallelize algorithms
8. Memory Workings & Optimization
- a. Pointers
 - b. Bits
 - i. Bit Manipulation
9. System Design

Mathematics

1. Differentiation
- a. Limits & Limit Rule
 - b. Partial Differentiation
 - i. Chain Rule over several variables
 - c. Chain Rule
 - d. Product Rule
 - e. Quotient Rule
 - f. Logarithmic Differentiation
 - g. Gradient Computations
 - h. Jacobian
 - i. Hessian
 - j. Newton's Method
 - k. Convexity
 - l. Critical Points
 - m. Lagrangian Multipliers
2. Integration
- a. U-substitution (inverse chain rule)
 - b. Integration by parts (inverse product rule)
 - c. Multiple Integration

3. Functions
 - a. Exponential Functions
 - i. Exponential Manipulation Rules
 - b. Logarithm Functions
 - i. Logarithm Manipulation Rules
 - c. Series
 - i. Convergence / Divergence
 - ii. Special Series
 - iii. Power Series
 - iv. Taylor Series
 - d. Functions of Several Variables
 - i. Vector Functions
 - ii. Calculus over Vector Functions
4. Sequences and Series
 - a. Taylor Series Approximation
 - b. Summation Manipulation
5. Linear Algebra
 - a. Vector Norms
 - b. Projection
 - c. Important Matrices
 - i. Diagonal Matrices
 - ii. Positive Semi-definite Matrices
 - iii. Conjugate Matrices
 - iv. Triangular matrices
 - v. Symmetric Matrices
 - vi. Orthogonal Matrices
 - d. Inversion
 - e. Trace
 - f. Matrix Factorization
 - i. LU Decomposition
 - ii. QR Factorization
 - iii. Cholesky Decomposition
 - iv. Singular Value Decomposition (Below)
 - v. Non-Negative Matrix Factorization
 - g. Gram-Schmidt
 - h. Matrix Multiplication
 - i. Vector Spaces
 - j. Linear Independence
 - k. Basis
 - l. Linear Transformations
 - m. Determinants

- n. Eigenvalues
- o. Eigenvectors
- p. Positive Definiteness, tests
- q. Pseudoinverses
- r. Cross Product

Probability Theory

1. Expectation, Mean, Variance
2. Conditional Probability
 - a. Bayes Rule
3. Sum and Product Rule
4. Independence
5. Covariance, Correlation
6. Distributions
 - a. Discrete (Probability Masses)
 - i. Binomial
 - ii. Bernoulli
 - iii. Multinomial
 - iv. Poisson
 - b. Continuous (Probability Densities)
 - i. Gaussian
 1. Conditional Gaussian
 2. Marginal Gaussian
 3. Mixtures of Gaussians
 - ii. Student's T-distribution
 - iii. Beta
 - c. Exponential Family
 - i. Maximum likelihood for exponentials
 - ii. Conjugate priors
 - iii. Noninformative priors
7. Information Theory
 - a. Cross Entropy
 - b. Mutual Information
8. Limit Theorems
 - a. Weak Law of Large Numbers
 - b. Strong Law of Large Numbers
 - c. Central Limit Theorem
9. Bayesian Statistical Inference
 - a. Bayesian inference and the posterior distribution
 - b. Point Estimation
 - c. Hypothesis Testing
 - d. Maximum a-Posteriori Rule

10. Classical Statistical Inference

- a. Binary Hypothesis Testing
- b. Significance Testing

Machine Learning

1. Algorithms

- a. Linear Regression
 - i. Derivation of Normal Form Equations / Gradient
 - ii. L1 / L2 / Elastic Net Regularization
 - iii. Bayesian Linear Regression
- b. Logistic Regression
 - i. Derivation from probability theory
 - ii. Derivation of Gradient, Hessian
 - iii. Multiclass vs. Binary logistic regression
 - iv. L1 / L2 / Elastic Net Regularization
 - v. Bayesian Logistic Regression
 - 1. Laplace Approximation
- c. Discriminant Functions
 - i. Linear Discriminant Analysis
 - ii. Fisher's linear discriminant
 - iii. The Perceptron Algorithm
- d. Neural Networks
 - i. Chain rule, Backprop derivation
 - ii. Feedforward Neural Networks
 - 1. Layer Types
 - 2. Activations
 - 3. Softmax
 - iii. Convolutional Neural Networks
 - 1. Convolution Operation
 - 2. Pooling
 - 3. Assumptions & Properties
 - a. Invariance to location of features (Equivariant to translation)
 - b. Weight Sharing for efficiency
 - c. Sparse weights / Sparse connectivity
 - i. Data Locality (Infinitely strong prior on local interactions mattering)
 - iv. Recurrent Neural Networks
 - 1. Structure
 - v. Word2Vec / Embeddings
 - vi. Autoencoders
 - vii. Batch Normalization
 - viii. Regularization

1. Early Stopping
 2. Weight Decay
 3. Dropout
 4. Data Augmentation
- ix. Bayesian Neural Networks
- e. Decision Trees
 - i. Random Forests
 1. Properties around bias-variance tradeoff
 2. Ensemble Modeling
 3. Extremely Random Forests
 - ii. Gradient Boosting
 1. Nature of Boosting
 - iii. Feature Importance
 - iv. Regularization
 1. Pruning
 2. Bootstrap
 3. Randomness
 - v. Decision Trees for Regression
- f. Maximum Margin Classifiers
 - i. Support Vector Machine
 1. Kernel Trick
 2. For Regression
 3. Optimizers
 - a. Quadratic Programming
 - b. Pegasus
- g. Naive Bayes
- h. KNN
- i. Splines and Piecewise Polynomials
 - i. MARS
- j. Principal Component Analysis
 - i. Maximum variance formulation
 - ii. Minimum error formulation
 - iii. Probabilistic PCA
 - iv. Kernel PCA
- k. Clustering
 - i. K-Means
 1. K-Means ++
 2. K-Medoids
 - ii. Hierarchical Clustering
 1. Varying linkage criterion - Single, Complete, Average Linkage
 - iii. Spectral Clustering

- I. Hidden Markov Model
 - m. Gaussian Mixture Model
 - i. Expectation-Maximization
 - n. Generative vs. Discriminative Models
 - o. Gaussian Processes
 - p. Variational Inference
 - q. MCMC
 - i. Markov Chains
 - ii. Metropolis-Hastings
 - iii. Gibbs Sampling
 - r. Graphical Models
 - i. Bayesian Networks
 - ii. Markov Random Fields
 - iii. Inference over Graphical Models
 - s. Optimizers
 - i. Gradient Descent
 - 1. Momentum
 - a. Nesterov Momentum
 - 2. RMSProp
 - 3. Adadelta
 - 4. Adagrad
 - 5. Adam
 - ii. Newton's Method
 - 1. Trust Newton
 - iii. BFGS
 - 1. L-BFGS
 - iv. Iteratively Reweighted Least Squares
 - v. Conjugate Gradients
 - vi. Line Search
2. Evaluation
 - a. Loss Functions
 - i. KL Divergence
 - ii. Cross Entropy
 - 1. Negative Log Likelihood
 - b. Validation
 - i. Cross Validation
 - 1. Leave one out
 - 2. Situations where valuable
 - ii. Proper validation methodology
 - c. Regression
 - i. RMSE

- ii. MAE
 - iii. Median
 - iv. R^2
 - v. Visualization (Especially of large errors)
 - d. Classification
 - i. ROC Curve
 - ii. Confusion Matrix
 - iii. F1 Score
 - iv. Heat Map
 - v. Overall accuracy rate
 - vi. Kappa Statistic
 - vii. Sensitivity
 - viii. Specificity
 - ix. AUC
 - x. Visualization (Especially of errors)
- 3. Testing
 - a. t-test
 - b. F-test
 - c. A-B Testing
- 4. Conceptual
 - a. Bias-Variance Tradeoff
 - b. Softmax and its properties
- 5. Classification Class Imbalance
 - a. Model Tuning (Tune Parameters For Sensitivity)
 - b. Alternate Cutoffs (Using ROC Curve)
 - c. Adjusting Prior Probability
 - d. Unequal Class Weights
 - e. Down Sampling
 - f. Up Sampling
 - g. Alter Cost Function
 - h. Dynamic Structure (Cascade of classifiers)
- 6. Hyperparameter Tuning
 - a. Cross Validation
 - b. Bootstrap
 - c. Grid Search
 - d. Random Search
 - e. Bayesian Optimization
- 7. Procedural Data Science
 - a. Preprocessing
 - b. Exploratory Data Analysis
 - c. Feature Evaluation



- i. Coefficients in Linear Models
- ii. Random Forest Importances (variance for regression, information gain for classification)
- iii. Pearson Correlation with Outcome
- iv. Maximal Information Coefficient (MIC)
- v. Distance Correlation ([code](#))
- vi. Model with/without feature
- vii. Randomly shuffle the feature between data points, check difference in model quality
- viii. Lasso Automatic Selection
- ix. Mean Decrease Accuracy ([code](#))
- x. Stability Selection
- xi. Recursive Feature Elimination

??

“Don’t I need interviews before I can study for them?”

Researchers who sign up to work with Rora get support landing interviews:

- *A resume revamp*
- *3-months of LinkedIn Premium*
- *Networking & outreach templates*
- *Hiring manager & recruiter names*

[→ See If You Qualify](#)

Appendix: Notes for the Netflix Culture Interview

Here are my notes for the Netflix culture interview, written in hindsight. Culture is about 50% of the interview process, and also a huge part of working at Netflix. I sent this to a friend before her interviews.

Culture memo summary: <https://jobs.netflix.com/culture>

These are the critical parts of the culture memo (must discuss)

- Freedom and responsibility
- Context, not control
- Honest, Productive Feedback

Other key aspects of culture:

- Informed captains
- Disagree, then commit
- Highly Aligned, Loosely Coupled

The “valued behaviors” from the culture memo are important too.

Mention sharing context a lot. It’s a key part of their culture. Managers view it as a primary part of their job, so you will make their job easier if you see it as part of your role as well.

Possible interview questions:

Are there any parts of the Netflix culture you think will be difficult?

- Option: Freedom and responsibility is an easy for this question
- Teams heading in different directions causing varying tech stacks
- Option: Working on an innovative new idea, as part of F+R, but needing to share context with colleagues to ensure alignment and that people aren’t headed in different directions or duplicating work



Tell me about a time you had to give a coworker negative/critical feedback?

- Traits from Steve's manager readme: Direct, specific, timely, balanced, open
- Disagree, then commit culture value If you're not the decision maker

Tell me about a time you received negative/critical feedback?

- Important: what did you do differently as a result, and how does it relate to your current work?
- Answer option: Seek to understand first (context)

Tell me about at time you disagreement you had with a difficult coworker?

- Answer option: "let me tell you about how I approach this in general, and I will give an example"
- Get to know coworker, their experiences, and their context for their view