Build
for
everyone

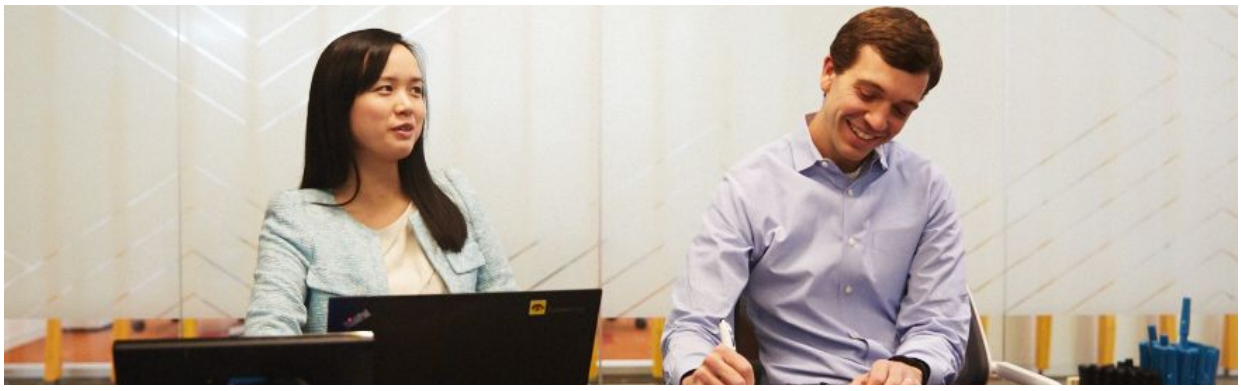
Google Interview Prep Guide
Machine Learning Software Engineer

What's a Machine Learning Software Engineer (SWE)?

Machine Learning Engineers at Google develop the next-generation technologies that change how millions of users connect, explore and interact with information and one another. Google believes the impact of AI will be most powerful when everyone can use it. As a Machine Learning SWE, you will specialize in supervised/unsupervised learning and apply techniques to various problems that scale to millions of users.

Why Google? Impact.

Google is and always will be an engineering company. We hire people with a broad set of technical skills who are ready to tackle some of technology's greatest challenges and make an impact on billions of users. At Google, engineers not only revolutionize search, they routinely work on massive scalability and storage solutions, large-scale applications and develop entirely new platforms around the world. From Assistant to Ads, Chrome to CloudAI, Tensorflow to Search, YouTube to Maps, Google Engineers are changing the world one technological achievement after another.



General Interview Tips

Explain - We want to understand how you think, so explain your thought process and decision making throughout the interview. Remember we're not only evaluating your technical ability, but also how you solve problems. Explicitly state and check assumptions with your interviewer to ensure they are reasonable.

Clarify - Many questions will be deliberately open-ended to provide insight into what categories and information you value within the technological puzzle. We're looking to see how you engage with the problem and your primary method for solving it. Be sure to talk through your thought process and feel free to ask specific questions if you need clarification.

Improve - Think about ways to improve the solution you present. It's worthwhile to think out loud about your initial thoughts to a question. In many cases, your first answer may need some refining and further explanation. If necessary, start with the brute force solution and improve on it — just let the interviewer know that's what you're doing and why.

Practice - You won't have access to an IDE or compiler during the interview so practice writing code on paper or a whiteboard. Be sure to test your code and ensure it's easily readable without bugs. Don't stress about small syntactical errors like which substring to use for a given method (e.g. start, end or start, length) — just pick one and let your interviewer know.

The Technical Phone Interviews

Your phone interview will cover data structures and algorithms. Be prepared to write around 20-30 lines of code in your strongest language. Approach all scripting as a coding exercise — this should be clean, rich, robust code.

1. You will be asked an open-ended question. Ask clarifying questions, devise requirements.
2. You will be asked to explain it in an algorithm.
3. Convert it to a workable code. Hint: Don't worry about getting it perfect because time is limited. Write what comes but then refine it later. Also make sure you consider corner cases and edge cases, production ready.
4. Optimize the code, follow it with test cases and find any bugs.

The Coding & Algorithm Interviews

Coding - You should know at least one programming language really well, preferably C++, Java, Python, Go, or C. You will be expected to know APIs, Object Oriented Design and Programming, how to test your code, as well as come up with corner cases and edge cases for code. Note that we focus on conceptual understanding rather than memorization.

Algorithms - Approach the problem with both bottom-up and top-down algorithms. You will be expected to know the complexity of an algorithm and how you can improve/change it. Algorithms that are used to solve Google problems include sorting (plus searching and binary search), divide-and-conquer, dynamic programming/memoization, greediness, recursion or algorithms linked to a specific data structure. Know Big-O notations (e.g. run time) and be ready to discuss complex algorithms like Dijkstra and A*. We recommend discussing or outlining the algorithm you have in mind before writing code.

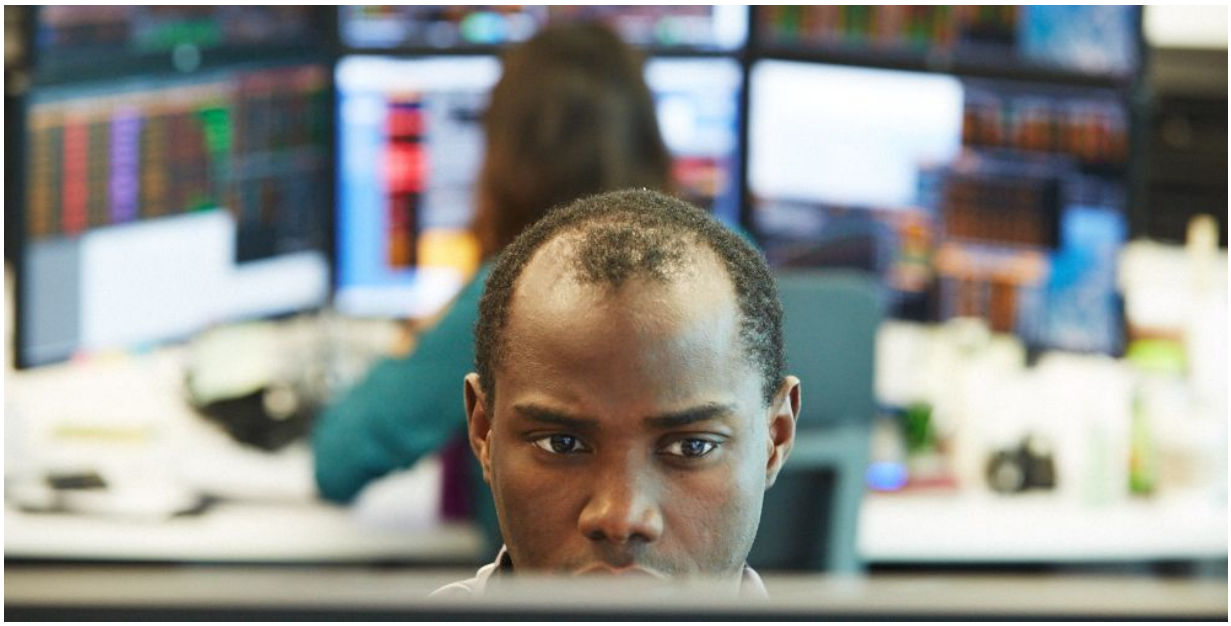
Sorting - Be familiar with common sorting functions and on what kind of input data they're efficient on or not. Think about efficiency means in terms of runtime and space used. For example, in exceptional cases insertion-sort or radix-sort are much better than the generic QuickSort/MergeSort/HeapSort answers.

Data structures - You should study up on as many data structures as possible. Data structures most frequently used are arrays, linked lists, stacks, queues, hash-sets, hash-maps, hash-tables, dictionary, trees and binary trees, heaps and graphs. You should know the data structure inside out, and what algorithms tend to go along with each data structure.

Mathematics - Some interviewers ask basic discrete math questions. This is more prevalent at Google than at other companies because counting problems, probability problems and other Discrete Math 101 situations surround us. Spend some time before the interview refreshing your memory on (or teaching yourself) the essentials of elementary probability theory and combinatorics. You should be familiar with n-choose-k problems and their ilk.

Graphs - Consider if a problem can be applied with graph algorithms like distance, search, connectivity, cycle-detection, etc. There are three basic ways to represent a graph in memory (objects and pointers, matrix, and adjacency list) — familiarize yourself with each representation and its pros and cons. You should know the basic graph traversal algorithms, breadth-first search and depth-first search. Know their computational complexity, their tradeoffs and how to implement them in real code.

Recursion - Many coding problems involve thinking recursively and potentially coding a recursive solution. Use recursion to find more elegant solutions to problems that can be solved iteratively.



Onsite Machine Learning Technical Prep

At least one interview will be focused on your expertise within Machine Learning. Please work directly with your recruiter to understand the panel of interview topics. General knowledge of the field and its main concepts should be demonstrated throughout the interview, such as supervised learning, unsupervised learning, overfitting, boosting, regularization. Experience with common learning algorithms such as logistic regression, decision trees, PCA, k-means and understanding/practice of how to apply those techniques to various problems is important.

Practice solving the coding problems yourself, entering them into a compiler to verify your solutions work and are bug-free. Remember: you won't have the benefit of using a compiler in your interview, so it's important to keep practicing until you can solve problems (bug-free!).

Think about solutions that can scale. Practice mapping algorithms to one or more Machine Learning paradigms and tuning/training them. Make sure you are able to analyze data and realize patterns in them to support solutions.

The interview will focus on three main areas:

- Computer Science Fundamentals
- Coding ability
- Basics of Machine Learning

In addition to the basics of Machine Learning, if you have a focus area - we can ask questions based on your expertise in the following topic(s):

Recommendations/Ranking/Prediction
Experience developing software that generates suggestions based on various input and output targets, ex. book or movie recommendations based on previous selections.

Computer Vision
Knowledge of methods for acquiring, processing, analyzing, and understanding images and, in general, high-dimensional data from the real world in order to produce numerical or symbolic information. Familiarity with one or several sub-domains: tracking, pose estimation, compression

Natural Language Processing
Knowledge and experience about processing/understanding natural language including main concepts like word embeddings, parse trees, parts of speech and the main tools like LSTM, attention, bidirectional models, beam search. Also understand the pro and cons of Machine Learning approaches compared to more classical approaches like ngram models.

Deep Learning/Neural Networks
Knowledge of Deep Learning algorithms inclusive of examples and features. Understanding of Convolutional Neural Network, Recurrent Neural Networks, optimization algorithms (Stochastic Gradient Descent, Backpropagation through time) and other aspects of the design/training of those networks.

Speech/Audio
Knowledge of technology designed to duplicate and respond to the human voice. Subfields include: speech synthesis, speech recognition, speaker recognition, speaker verification, speech/audio encoding, multimodal interaction

Reinforcement Learning
Knowledge of an area of machine learning about sequential decision making, in which an agent takes actions in an environment, trying to maximize the sum of expected rewards. This includes sequential environment modeling and control. This requires knowledge of the main concepts (Markov Decision Processes, value functions, Bellman equations, policy gradient, Q-learning, and actor-critic.) Be able to cast concrete problems into a RL formulation.

The Large Scale Design Interview

In this interview, you should be able to combine your current Machine Learning knowledge, theories, experience and judgement toward solving a real-world engineering problem. Questions are generally open ended so you can dive deeper into the solution. It is important to design a solution thinking about data analysis, potential use cases, the users, scalability, limitations and tradeoffs.

Resources

Books

Cracking the Coding Interview
Gayle Laakmann McDowell

Programming Interviews Exposed: Secrets to Landing Your Next Job
John Mongan, Eric Giguere, Noah Suojanen, Noah Kindler

Programming Pearls
Jon Bentley

Introduction to Algorithms
Thomas Cormen, Charles Leiserson, Ronald Rivest, Clifford Stein

Interview Prep

How we hire

Interviewing @ Google

Candidate Coaching Session: Tech Interviewing

CodeJam: Practice & Learn

Technical Development Guide

About Google

Company - Google

The Google story

Life @ Google

Google Developers

Open Source Projects

Github: Google Style Guide

Google Publications

The Google File System

Bigtable

MapReduce

Google Spanner

Google Chubby