MentorCruise

Search

2025 40 curated interview questions

40 PyTorch Interview Questions

Master your next PyTorch interview with our comprehensive collection of questions and expert-crafted answers. Get prepared with real scenarios that top companies ask.

Start Studying Questions | Book Interview Coaching

PYTORCH INTERVIEW QUESTIONS FROM HIRING MANAGERS AT

amazon Microsoft airbnb NETFLIX Uber

Study Mode

List View | Flashcards

1. How do you set and use learning rate schedulers in PyTorch?

In PyTorch, learning rate schedulers adjust the learning rate during training, which can help the model converge faster and better. You first need to setup your optimizer, like so: `python optimizer = torch.optim.Adam(model.parameters(), lr=0.001)` Once the optimizer is in place, you can define a scheduler. For instance, if you want to use a StepLR scheduler which decays the learning rate by a factor every few epochs, you'd

set it up like this: `python scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=30, gamma=0.1)` Then, in your training loop, simply step the scheduler at the end of each epoch: `python for epoch in range(num_epochs): train(...) # Training step validate(...) # Validation step scheduler.step() # Update the learning rate` That's it! You can choose different schedulers like ExponentialLR, ReduceLROnPlateau, etc., depending on your needs.

Hide Answer

2. How does PyTorch handle automatic differentiation?

PyTorch uses a feature called Autograd for automatic differentiation. Autograd records all the operations that you perform on tensors in a dynamic computation graph. When you want to compute gradients, you simply call the `.backward()` method on a tensor that represents a scalar value, and PyTorch traverses this graph in reverse to calculate and store the gradients of all tensors involved. This process is efficient and allows for flexibility in building and modifying neural networks on the fly.

Hide Answer

3. Can you explain what a computational graph is and how it is used in PyTorch?

A computational graph is a representation of the mathematical operations that occur within a neural network. It's essentially a graph where nodes represent operations (like addition, multiplication) or variables, and edges represent the dependencies between these operations. This graph structure allows for efficient computation of derivatives, which is crucial for gradient-based optimization techniques in training neural networks.

In PyTorch, the computational graph is dynamic, meaning it's built on-the-fly as you perform operations on tensors. This is different from static graphs in other frameworks like TensorFlow, where the graph is defined and then executed. The dynamic nature of PyTorch's computational graph makes it more intuitive and easier to debug because it

reflects the actual code execution flow. When you call the `.backward()` method on a tensor, PyTorch traverses this graph to compute gradients for all the tensors involved, which are then used to update the model parameters during backpropagation.

Hide Answer

4. How do you manually compute gradients for a simple operation in PyTorch?

To manually compute gradients in PyTorch, you need to use `requires_grad=True` when defining your tensors. Let's say you have a simple operation, like (z = x^2 + y^2). You'd define your tensors `x` and `y` with `requires_grad=True` , perform the operation, and then call `backward()` on the result to compute the gradients.

Here’s a quick example: ```python import torch

Define tensors

x = torch.tensor(2.0, requires_grad=True) y = torch.tensor(3.0, requires_grad=True)

Perform operation

z = x**2 + y**2

Compute gradients

z.backward()

Access gradients

print(x.grad) # Should output tensor(4.0) print(y.grad) # Should output tensor(6.0) ``` **In this case, the gradients are the partial derivatives of \(z \) with respect to** x **and** y **. After calling** backward() **,** x.grad **will be 4 (because \(\partial z/\partial x = 2x \)) and** y.grad` will be 6 (because (\partial z/\partial y = 2y)).

Hide Answer

5. What is a neural network module in PyTorch and how do you define one?

In PyTorch, a neural network module is essentially a building block for constructing neural networks. It's represented by the **`torch.nn.Module`** class, which you can subclass to create your custom network architectures. When you define a new neural network module, you typically implement two main components: the **`__init__`** method, where you define the layers, and the **`forward`** method, where you specify the forward pass or how data flows through the network.

Here's a simple example: ```python import torch import torch.nn as nn

class SimpleNet(nn.Module): def **init**(self): super(SimpleNet, self).**init**() self.fc1 = nn.Linear(10, 5) # Layer 1: Fully connected layer from 10 to 5 nodes self.fc2 = nn.Linear(5, 2) # Layer 2: Fully connected layer from 5 to 2 nodes

```
def forward(self, x):
    x = torch.relu(self.fc1(x))  # Apply ReLU activation after Layer 1
    x = self.fc2(x)  # No activation after Layer 2
    return x
```

Create an instance of the network

model = SimpleNet() `` **In this example,** SimpleNet **is a neural network module with two fully connected layers. The** forward **method defines how the input tensor** x` is transformed as it passes through these layers.

Hide Answer

6. How do you load and preprocess data for training a model in PyTorch?

Loading and preprocessing data in PyTorch usually involves using the **torchvision** library for common datasets and the **Dataset** and **DataLoader** classes for custom data. To start, you typically define a custom dataset by subclassing **torch.utils.data.Dataset** and implementing the **__len__** and **__getitem__** methods. The **__getitem__** method is where you'd handle any preprocessing or transformations, which can be facilitated by **torchvision.transforms** .

Once your dataset is ready, you pass it to a **DataLoader** , which will handle batching, shuffling, and parallel loading of data. You can specify parameters like batch size, number of worker processes for data loading, and whether to shuffle the data at each epoch. This setup makes it efficient to load data on the fly during training.

Hide Answer

7. What is PyTorch and why would you choose it over other deep learning frameworks like TensorFlow or Keras?

PyTorch is an open-source machine learning framework developed by Facebook's AI Research lab. It's known for its dynamic computation graph, which allows for more flexible model design and easier debugging compared to static graphs used in frameworks like TensorFlow. This makes PyTorch particularly user-friendly and intuitive, especially for research purposes and rapid prototyping.

One reason you might choose PyTorch over TensorFlow or Keras is its dynamic graph construction, which can be altered during runtime, versus the static graph of TensorFlow that you need to define and then execute. This dynamic aspect simplifies the creation and modification of complex models. Also, PyTorch's syntax tends to be more Pythonic, making it easier for those already familiar with Python to pick up. The strong community support and extensive libraries built around PyTorch are another big plus.

Hide Answer

8. Explain the difference between a tensor and a NumPy array.

A tensor and a NumPy array are similar in that they both represent n-dimensional arrays of data. However, there are some key differences. Tensors are a core feature of PyTorch and are designed to work seamlessly with GPUs, which is a huge advantage for deep learning tasks that require significant computational resources. This means you can move tensors between CPU and GPU effortlessly and perform operations that are optimized for either hardware.

On the other hand, NumPy arrays are the backbone of the NumPy library which is well-suited for general-purpose numerical computations, but it doesn't natively support GPU acceleration. Another difference is that PyTorch tensors provide automatic differentiation, which is crucial for training neural networks. PyTorch's autograd system records operations on tensors to calculate gradients during the backward pass, a feature not available in NumPy arrays.

Hide Answer

Master Your PyTorch Interview

Essential strategies from industry experts to help you succeed

Research the Company

Understand their values, recent projects, and how your skills align with their needs.

Practice Out Loud

Don't just read answers - practice speaking them to build confidence and fluency.

Prepare STAR Examples

Use Situation, Task, Action, Result format for behavioral questions.

Ask Thoughtful Questions

Prepare insightful questions that show your genuine interest in the role.

9. What is the purpose of the `autograd` module in PyTorch?

Show Answer

10. Can you explain the role of `DataLoader` and `Dataset` classes in PyTorch?

Show Answer

11. How do you use GPUs to accelerate computations in PyTorch?

Show Answer

12. Explain the difference between `torch.optim.SGD` and `torch.optim.Adam`.

Show Answer

13. Describe what an activation function is and give examples of commonly used activation functions in PyTorch.

Show Answer

14. Explain the concept of batch normalization and how it is implemented in PyTorch.

Show Answer

15. How do you handle and avoid overfitting in PyTorch models?

Show Answer

16. How do you create a tensor in PyTorch, and what are some of the common functions to initialize a tensor?

Show Answer

17. Explain the difference between `torch.Tensor` and `torch.Variable`.

Show Answer

18. How do you perform element-wise operations with PyTorch tensors?

Show Answer

19. What is the `optimizer` in PyTorch and how do you set it up?

Show Answer

20. How do you save and load a trained model in PyTorch?

Show Answer

21. What is the purpose of the `torch.nn.functional` module and how is it different from the `torch.nn` module?

Show Answer

22. What is a loss function, and how do you implement custom loss functions in PyTorch?

Show Answer

23. How do you use the `nn.Module` class to build custom neural networks in PyTorch?

Show Answer

24. Explain the significance of the `forward` method in PyTorch `nn.Module` class.

Show Answer

25. Describe what `torch.no_grad()` does and when you would use it.

Show Answer

Show 15 more questions

Get Interview Coaching from PyTorch Experts

Knowing the questions is just the start. Work with experienced professionals who can help you perfect your answers, improve your presentation, and boost your confidence.

Tracy Pham
Data Science Manager
★★★★★ (17)

I have been a mentor for more than 2 years on MentorCruise and other online platforms. I find myself loving to help people with their ...

Natural Language Processing Deep Learning Machine Learning

View Profile

Nima Tajbakhsh

Deep Learning Lead @ Nvidia

★★★★★ (67)

Free introductory call Choosing the right mentor is a crucial decision. You want to ensure that your mentor is a great fit for your needs ...

Generative AI LLM Multimodal LLM

View Profile

Siddharth Agrawal

Principal Research Scientist @ Motorola Solutions

★★★★★ (34)

I'm a Principal Research Scientist at Motorola Solutions, where I work on cutting-edge deep learning/computer vision algorithms for the security industry. I have 7+ years ...

AI Deep Learning Computer Vision

View Profile

Chris Hammerschm...

ML Research Scientist and Founder @ APTA Technologies

★★★★★ (25)

As a mentor with a background in both research and industry, I have a wealth of experience of 10+ years to draw upon when guiding ...

Machine Learning Data Science Product Market Fit

View Profile

Azzam Alwan
Sr. Data scientist | 2xFounder | TOP 8 inventors in MENA | Phd. ML/DL @ Liricare
★★★★★ (61)

Hello! Check my video first 🙂 I'm a research scientist with a deep-seated passion for making sense of data and a flair for solving complex ...

Data Science Software Engineering Machine Learning

View Profile

Reza Fazeli
Machine Learning Engineer @ IBM, Ex-SoundHound
★★★★★ (23)

Need help with data science and machine learning skills? I can guide you to the next level. Together, we'll create a personalized plan based on ...

Machine Learning Natural Language Processing Computer Vision

View Profile

Browse All Interview Coaches

Still not convinced? Don't just take our word for it

We've already delivered 1-on-1 mentorship to thousands of students, professionals, managers

Farzad
Leadership Mentee

*"**Naz** is an amazing person and a wonderful mentor. She is supportive and knowledgeable with extensive practical experience. Having been a manager at Netflix, she also knows a ton about working with teams at scale. Highly recommended."*

and executives. Even better, they've left an average rating of 4.9 out of 5 for our mentors.

Get Interview Coaching

Rao
Engineering Mentee

*"**Brandon** has been supporting me with a software engineering job hunt and has provided amazing value with his industry knowledge, tips unique to my situation and support as I prepared for my interviews and applications."*

Clara
Web Development Mentee

*"**Sandrina** helped me improve as an engineer. Looking back, I took a huge step, beyond my expectations."*

Complete your PyTorch interview preparation

Comprehensive support to help you succeed at every stage of your interview journey

PyTorch Resume Review

Get your resume reviewed by industry experts to make sure it gets past ATS systems and impresses hiring managers.

PyTorch Mock Interviews

Practice with experienced professionals who can simulate real interview conditions and provide immediate feedback.

PyTorch Salary Negotiation

Learn how to negotiate your salary and benefits package effectively with guidance from seasoned professionals.

MentorCruise

Your trusted source to find highly-vetted mentors & industry professionals to move your career ahead.