

Q1: Explain the primary differences between TensorFlow and PyTorch. When would you choose one over the other?

Solution

TensorFlow and PyTorch are both powerful deep learning frameworks with distinct characteristics:

Tensor Flow

- Originally used static computation graphs (define-then-run)
- Strong production deployment capabilities with Tensor Flow Serving, Lite, and JS
- Comprehensive ecosystem (Keras integrated, Tensor Board for visualization)
- Industry preference for production systems

PyTorch

- Uses dynamic computation graphs (define-by-run)
- More Pythonic and intuitive for research and prototyping
- Excellent debugging capabilities with standard Python tools
- Research community preference

When to choose:-

1. Tensor Flow – Production deployment, mobile/edge devices, when using Tensor Flow Extended (TFX) for MLOps
2. PyTorch - Research projects, rapid prototyping, when dynamic graph flexibility is needed

Q2: Describe two use cases for Jupyter Notebooks in AI development.

Solution

1. Interactive Model Prototyping and Experimentation:- Jupyter Notebooks allow data scientists to iteratively develop and test models, immediately visualizing results and adjusting parameters without rerunning entire scripts.
2. Educational Demonstrations and Collaborative Research:- Notebooks combine code, visualizations, and explanatory text in a single document, making them ideal for teaching concepts, sharing research findings, and collaborative debugging.

Q3: How does spaCy enhance NLP tasks compared to basic Python string operations?

Solution

SpaCy provides sophisticated linguistic capabilities that basic string operations lack:

- **Contextual Understanding:** Recognizes entities, parts of speech, and dependencies based on linguistic context rather than just pattern matching
- **Pre-trained Models:** Comes with industrial-strength models trained on large corpora
- **Efficiency:** Optimized Cython implementation for processing large volumes of text
- **Linguistic Features:** Handles lemmatization, named entity recognition, and dependency parsing that would be extremely complex with basic string operations

Scikit-learn and Tensor Flow Comparison

Aspect	Scikit-learn	Tensor Flow
Target Applications	Classical ML algorithms (SVM, decision trees, clustering)	Deep learning and neural networks
Ease of Use for Beginners	Very easy with consistent API and minimal setup	Steeper learning curve, especially with low-level API
Community Support	Excellent documentation and large community for traditional ML	Massive ecosystem with strong industry and research backing
Performance	Optimized for CPU, good for small-medium datasets	GPU acceleration, scales to large datasets and complex models
Deployment	Simple models, REST APIs	Comprehensive deployment options (mobile, web, edge)