**Part 1: Theoretical Understanding**

**1. Short Answer Questions**

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

TensorFlow (developed by Google) uses a graph-based execution model, where computations are defined as a static graph (though it supports eager execution for dynamic graphs since version 2.0). It emphasizes production deployment with tools like TensorFlow Serving and TensorFlow Lite for mobile/edge devices. PyTorch (developed by Meta) is built around dynamic computation graphs (imperative programming), making it more intuitive for rapid prototyping and debugging, as code runs line-by-line like standard Python.

Key differences:

* **Graph Execution**: TensorFlow's static graphs optimize for performance in large-scale deployments but can be less flexible. PyTorch's dynamic graphs allow easier modifications during runtime.
* **API Style**: TensorFlow has a more declarative API (define-then-run), while PyTorch is imperative (define-by-run).
* **Ecosystem**: TensorFlow has stronger support for distributed training and production tools; PyTorch excels in research with seamless integration to libraries like Hugging Face.

Choose TensorFlow for production environments, scalability in enterprise (e.g., recommendation systems at scale), or when needing built-in deployment tools. Choose PyTorch for research, experimentation, or when flexibility and ease of debugging are priorities (e.g., custom model architectures in academia).

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

1. **Interactive Data Exploration and Prototyping**: Jupyter allows mixing code, text, and visualizations in one document. For AI, this is ideal for loading datasets (e.g., via pandas), preprocessing, training small models, and iterating quickly—e.g., experimenting with hyperparameters in a PyTorch model and plotting loss curves with matplotlib inline.
2. **Collaboration and Documentation**: Teams can share notebooks with embedded explanations, results, and code. In AI projects, this supports reproducible research, like documenting a TensorFlow pipeline for image classification, including model summaries, evaluation metrics, and visualizations, making it easy for others to review or extend.

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

Basic Python string operations (e.g., split(), find(), regex via re module) are low-level and require manual implementation for tasks like tokenization or entity extraction, often leading to inefficient, error-prone code for complex text.

spaCy enhances NLP by providing:

* **Pre-trained Pipelines**: Industrial-strength models for tokenization, part-of-speech tagging, dependency parsing, and named entity recognition (NER) out-of-the-box, trained on large corpora for accuracy.
* **Efficiency**: Optimized Cython-based implementation for speed on large texts, unlike slow pure-Python loops.
* **Extensibility**: Easy to add custom rules or integrate with ML models (e.g., for sentiment analysis).
* **Visualization**: Built-in tools like displacy for rendering parse trees or entities.

For example, extracting entities from a review is a one-liner in spaCy (doc.ents), versus writing custom regex patterns that miss nuances like context or variations.

**2. Comparative Analysis**

| **Aspect** | **Scikit-learn** | **TensorFlow** |
| --- | --- | --- |
| **Target Applications** | Classical ML tasks like classification, regression, clustering (e.g., decision trees on tabular data, SVMs for spam detection). Not suited for deep learning. | Deep learning and neural networks (e.g., CNNs for image recognition, RNNs for sequences). Can handle classical ML but overkill for it. |
| **Ease of Use for Beginners** | High: Simple, consistent API with minimal boilerplate. Quick to learn for non-DL tasks (e.g., fit() and predict()). | Moderate: Steeper curve due to graph concepts and verbosity, but Keras API simplifies it for beginners. |
| **Community Support** | Strong: Vast tutorials, Stack Overflow answers for ML basics. Integrated with pandas/numpy ecosystems. | Excellent: Backed by Google, huge resources for DL (e.g., official guides, TensorFlow Hub for pre-trained models). Active forums and research papers |