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?

- Computation Style:
    - *TensorFlow* originally used static computation graphs, where you define the model first, then run it. However, with TensorFlow 2.x, it now supports eager execution, making it more flexible.
    - *PyTorch,* on the other hand, uses dynamic computation graphs. This means the graph is built on the fly as operations are run, making it more intuitive.

- Development Background:
    - TensorFlow, developed by Google, is widely used in production environments.
    - PyTorch is developed by Meta (formerly Facebook) and is a favorite among researchers due to its flexibility and ease of use.
- Deployment Tools:
  TensorFlow shines in deployment, offering tools like TensorFlow Serving, TensorFlow Lite (for mobile), and TensorFlow.js (for web).
- PyTorch has TorchServe, but its production ecosystem is still maturing

When to Choose:

- Use TensorFlow if you're building large-scale AI applications, especially for enterprise or mobile deployment.
  Go with PyTorch if you're working in academia, research, or need to rapidly prototype models.

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

Jupyter Notebooks are incredibly useful tools in AI/ML development. Since it enhances:

1. Interactive Learning and Prototyping
   You can write and run code step-by-step, visualize results instantly, and tweak models in real-time. This makes Jupyter great for exploration, learning, and debugging.

2. Collaborative Research and Teaching

   Since notebooks support code, comments, images, and charts all in one document, they're perfect for collaborating with team members, sharing experiments, or even teaching AI concepts.

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

- Advanced Linguistic Understanding: spaCy comes with pre-trained models that can:
  - Detect Part-of-Speech (POS) tags
  - Parse sentence structures
  - Recognize Named Entities like people, places, dates
- Efficiency and Speed: It's optimized for performance, so it can handle large volumes of text very fast.
- Extra Features
  - Built-in tokenization
  - Lemmatization finds the root forms of words
  - Support for word vectors and similarity checks
  - Integration with transformer models like BERT

    Visual tools like displaCy to explore syntax trees and named entities

2. Comparative Analysis

- Compare Scikit-learn and TensorFlow in terms of:
  - Target applications (e.g., classical ML vs. deep learning).
  - Ease of use for beginners.
  - Community support.

Aspect	**Scikit-learn**	**TensorFlow**
Target use	Classical ML, e.g., regression, clustering	Deep Learning (e.g., CNNs, RNNs, transformers)
Ease of use	Beginner-friendly, clean, and simple API	Steeper learning curve, but Keras API helps simplify usage

Best for	Small to medium-sized datasets, quick ML models	Complex deep learning tasks and large-scale applications
Community support	Strong among data scientists and traditional ML users	Large global community, industry-grade tools, and docs

When to choose:

- Use Scikit-learn when you want to solve problems like classification, regression, or clustering using classical ML on moderate datasets.
- Choose TensorFlow when building neural networks or deploying scalable AI systems to production or cloud platforms.

Primary sources:

References used in Week 3 content: Google, AI

TensorFlow Official Documentation https://www.tensorflow.org/guide

PyTorch Official Documentation https://pytorch.org/docs/stable/index.html
Comparative Blog Articles:

- PyTorch vs TensorFlow in 2024 https://towardsdatascience.com/pytorch-vs-tensorflow-in-2024-c246b7434f0e

Part 3: Ethics & Optimization.

1. Ethical Considerations

Bias in MNIST Digit Classification Model.

The MNIST dataset has potential biases due to the limited demographics of its contributors, mainly American high school students and census employees. This can affect te model's ability to generalize to handwriting styles from other populations. In addition, if certain digits like '9' or '8' are underrepresented in the training set, the model may perform poorly on them, leading to class imbalance bias.

Bias in Amazon Reviews Sentiment Model

The model trained on Amazon product reviews can exhibit:

- Sentiment Polarity Bias: If the dataset has more positive reviews, the model may be biased toward positive predictions.
- Language and Tone Bias: Informal language, sarcasm, and dialects may be misinterpreted by the model, resulting in incorrect sentiment classification.

Bias Mitigation Techniques

TensorFlow Fairness Indicators:

- A powerful tool to evaluate model performance across different groups.
- Helps identify if the model behaves unfairly for certain labels (e.g., digits 0–9, or specific user groups).
- Can visualize metrics like accuracy or false-positive rates per subgroup.

spaCy Rule-Based Systems:

- Rule-based logic allows manual crafting of conditions, reducing the model's dependence on biased data.
- Useful for identifying known sentiment patterns or specific named entities without requiring extensive training data.

Troubleshooting Challenge: Debugging a TensorFlow Script

Key Debugging Tips for TensorFlow Models

1. Check Input and Output Shapes

- Use shape on your datasets and model.summary() to verify input/output dimensions.
- This helps prevent common errors like mismatched dimensions between layers or

2. Use the Right Loss Function

- Choose a loss function based on your task and label format:
    - **binary_crossentropy**: For binary classification (2 classes, one-hot or 0/1 labels).
    - **sparse_categorical_crossentropy**: For multi-class classification using integer labels.
    - **categorical_crossentropy**: For multi-class classification using one-hot encoded labels.

3. Apply Appropriate Activation Functions

- The final layer should reflect the nature of the output:
    - **softmax** for multi-class classification.
    - **sigmoid** for binary classification.

4. Normalize Your Input Data

- Scaling your input features improves model convergence and accuracy.
- For image data, normalize pixel values from [0, 255] to [0, 1].