1. Short Answer Questions

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

  a) Execution Model
  - TensorFlow: Static computation graph (define-then-run)
  - PyTorch: Dynamic computation graph (define-by-run, more intuitive)

  b) Debugging
  - TensorFlow: Harder to debug due to static graph
  - PyTorch: Easier debugging with native Python tools (e.g., print, pdb)

  c) Deployment
  - TensorFlow: Better for production, mobile, and cross-platform deployment
  - PyTorch: Improving, but traditionally more research-focused

  d) Community & Adoption
  - TensorFlow: Backed by Google, widely used in industry
  - PyTorch: Backed by Meta, dominant in academia and research

  When to Choose
  - Choose TensorFlow:
    - Need production-ready tools
    - Targeting mobile/web deployment
    - Using Google Cloud ecosystem
  - Choose PyTorch:
    - Rapid prototyping
    - Research and experimentation
    - Prefer Pythonic, flexible code

- Q2: Use Cases for Jupyter Notebooks in AI Development

1. **Interactive Experimentation**: Ideal for testing models, visualizing data, and tweaking hyperparameters in real-time.
2. **Documentation and Tutorials**: Combines code, output, and markdown to create readable, shareable learning materials or project walkthroughs.

- Q3: spaCy vs Basic Python String Operations in NLP

  spaCy enhances NLP tasks by offering:

  - **Tokenization, POS tagging, and Named Entity Recognition (NER)** out of the box, using trained statistical models.
  - **Language-specific pipelines** that understand context, grammar, and semantics—far beyond simple `.split()` or regex.

2. Comparative Analysis

a) Target Application
   i. Scikit- Learn
      - Best for classical ML (e.g., regression, classification, clustering)
      - Ideal for structured/tabular data.
      - Not designed for deep learning
   ii. Tensorflow:
      - Built for deep learning (e.g., CNNs, RNNs, transformers).
      - Suited for unstructured data (images, text, audio)
      - Scales well for large datasets and production.

b) Ease of Use for Beginners
   i. Scikit-Learn:
      - Simple, consistent API (fit, predict, transform).
      - Minimal setup, fast to prototype.
      - Great for learning ML fundamentals.
   ii. Tensorflow:
      - Steeper learning curve (especially low-level API).
      - Easier with Keras (high-level wrapper).
      - Requires understanding of tensors and computational graphs.

c) Community Support.
   iii. Scikit-Learn:
      - Strong academic and data science community).
      - Extensive documentation and tutorials.
      - Stable and mature library.
   iv. Tensorflow:
      - Massive global community, backed by Google.
      - Rich ecosystem (TensorBoard, TFLite, TF Hub)
      - Frequent updates, large number of learning resources.

Part 2: Practical Implementation (50%)

Handwritten Model

```
# Model Evaluation
model.evaluate(X_test, y_test)
[65]

313/313 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.9785 - loss: 0.0734
[0.07338853925466537, 0.9785000085830688]
```

Iris Model

```
# 📊 Step 5: Evaluate the Model
[26]

# Evaluate using accuracy, precision, and recall
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision (macro):", precision_score(y_test, y_pred, average='macro'))
print("Recall (macro):", recall_score(y_test, y_pred, average='macro'))
[27]

Accuracy: 1.0
Precision (macro): 1.0
Recall (macro): 1.0
```

NER results

```
139209  I just discovered Coconut Oil this year.  Coco...

                                                 Entities Sentiment
18828                                                []  Positive
363857                                               []   Neutral
342609                                 [(Native Forest, ORG)]   Neutral
62213                                                []   Neutral
467133                                               []   Neutral
518641  [(Delivery, ORG), (Connoisseur, ORG), (USDA, O...  Positive
539725                                               []  Positive
```

Part 3: Ethics & Optimization

1. Ethical Considerations

a) Potential Biases

MNIST Model

• Class imbalance: Some digits (e.g., ‘1’, ‘7’) may appear more frequently, leading to skewed predictions.

• Handwriting style bias: Model may perform poorly on digits written in styles not well represented in the training set (e.g., regional or age-based variations).

Amazon Reviews Model

• Sentiment bias: Overrepresentation of positive or negative reviews can skew predictions.

• Demographic bias: Language patterns from certain regions or dialects may be misclassified.

• Product category bias: Sentiment may vary by category (e.g., electronics vs. books), but model may not account for this.

b) Mitigation Tools

TensorFlow Fairness Indicators

• Analyze model performance across slices (e.g., gender, age, product category).

• Identify disparities in accuracy, precision, recall across subgroups.

• Visualize fairness metrics to guide retraining or data augmentation.

spaCy’s Rule-Based Systems

• Customize token patterns to reduce bias in entity recognition or sentiment tagging.

• Apply consistent rules across dialects or product categories to normalize interpretation.

• Combine with statistical models to override biased predictions in edge cases

2. Troubleshoot Challenge

Sample Buggy Code

```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(128, input_shape=(28, 28)),
    tf.keras.layers.Dense(10, activation='softmax')
])


model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
```

Issues:

- Input shape mismatch: input_shape=(28, 28) expects 2D input, but Dense layers need 1D.
- Incorrect loss function: binary_crossentropy is wrong for multi-class classification

Fixed Code:

```
model = tf.keras.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),  # Flatten 2D input
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(10, activation='softmax')  # 10 classes
])

model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

Explanation

Flatten() converts 28x28 input into 784-dim vector

'sparse_categorical_crossentropy is correct when labels are integers (0–9).