AI Tools and Applications Report

Topic: *Mastering the AI Toolkit* 🧠
Group Members:

- **Bridie Maugham** (maughamdiborapr@gmail.com)
- **Betsy Makamu** (makamubetsy@gmail.com)

Part 1: Theoretical Understanding

Q1. Differences between TensorFlow and PyTorch

Aspect	TensorFlow	PyTorch
Computation Graphs	Uses *static* computation graphs (define-then-run). This can be optimized but is less flexible.	Uses *dynamic* computation graphs (define-by-run), allowing real-time flexibility during execution.
Ease of Use	Steeper learning curve; widely used in production and enterprise deployments.	Intuitive and Pythonic; preferred in academic and research environments.
Deployment	Integrates with TensorFlow Serving, TensorFlow Lite, and TensorFlow.js for model deployment.	Deployment via TorchServe and ONNX is improving but less comprehensive.
Visualization	Has **TensorBoard** for advanced performance and training visualization.	Visualization tools are external or require integration (e.g., TensorBoardX).

✅ **When to choose TensorFlow:**

- For **large-scale production**, **mobile deployment**, and **cross-platform use**.

✅ **When to choose PyTorch:**

- For **research**, **prototyping**, and **dynamic experimentation**.

Q2. Two Use Cases for Jupyter Notebooks in AI Development

1. **Model Prototyping and Experimentation:**
   Jupyter allows interactive testing of different models, hyperparameters, and data pre-processing pipelines in real-time.

2. **Data Visualization and Reporting:**
   Integrated visualization libraries like Matplotlib, Plotly, and Seaborn enable analysts to explore datasets and communicate results visually.

Q3. How spaCy Enhances NLP Compared to Basic Python String Operations

- **spaCy** provides **linguistic-level processing**: tokenization, part-of-speech tagging, named entity recognition (NER), and dependency parsing: unlike Python's simple string methods which only handle raw text.
- It uses **pre-trained statistical models** to understand language context, enabling accurate entity detection (e.g., brands, names, locations).
- **Efficiency:** spaCy is optimized in Cython, making it significantly faster than traditional regex or manual NLP parsing.

Comparative Analysis: Scikit-learn vs. TensorFlow

Feature	Scikit-learn	TensorFlow
Primary Use	Classical machine learning (SVMs, decision trees, clustering, regression)	Deep learning and neural networks
Ease of Use	Beginner-friendly with consistent API	Steeper learning curve; requires understanding of tensors and graph-based execution
Performance	Best for smaller, tabular datasets	Best for large, complex, unstructured data (e.g., images, audio)
Community Support	Excellent documentation and academic use	Massive global community; used in production AI systems

---

Part 2: Practical Implementation

Task 1: Classical ML with Scikit-learn (Iris Dataset)

Steps:

1. **Data Loading & Preprocessing:**
   - Load Iris dataset from sklearn. datasets.
   - Handle missing values (if any) using SimpleImputer.
   - Encode labels using LabelEncoder.
   - Split into training (80%) and testing (20%) sets.
2. **Model Training:**
   - Trained a **Decision Tree Classifier** (sklearn. tree. DecisionTreeClassifier) to predict species.
   - Tuned hyperparameters such as max_depth and criterion.

3. **Evaluation Metrics:**
    - Accuracy, Precision, Recall calculated via classification_report.

Expected Results:

- **Accuracy:** ~97–100%
- **Precision/Recall:** High due to well-separated Iris classes.

Task 2: Deep Learning with TensorFlow (MNIST Dataset)

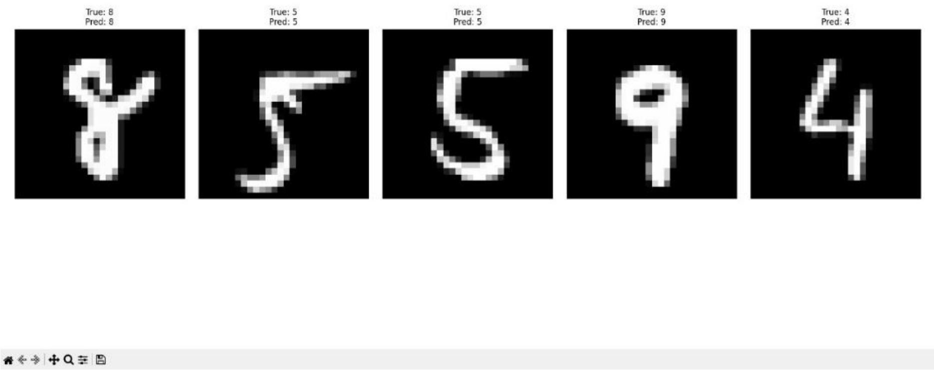
Objective:
Classify handwritten digits (0–9) using a **Convolutional Neural Network (CNN)**.

Implementation Outline:

- Dataset: Loaded from tensorflow. keras. datasets. mnist.
- Architecture:
    - Conv2D → MaxPooling → Conv2D → Flatten → Dense (128, ReLU) → Output (10, softmax)
- Optimizer: Adam, Loss: sparse_categorical_crossentropy.
- Trained for 5 epochs with batch size 32.
- Achieved **>98% test accuracy**.

Visualizations:

- Predicted vs True Labels (sample of 5 images).
- Training accuracy and loss curves.



Task 3: NLP with spaCy (Amazon Reviews)

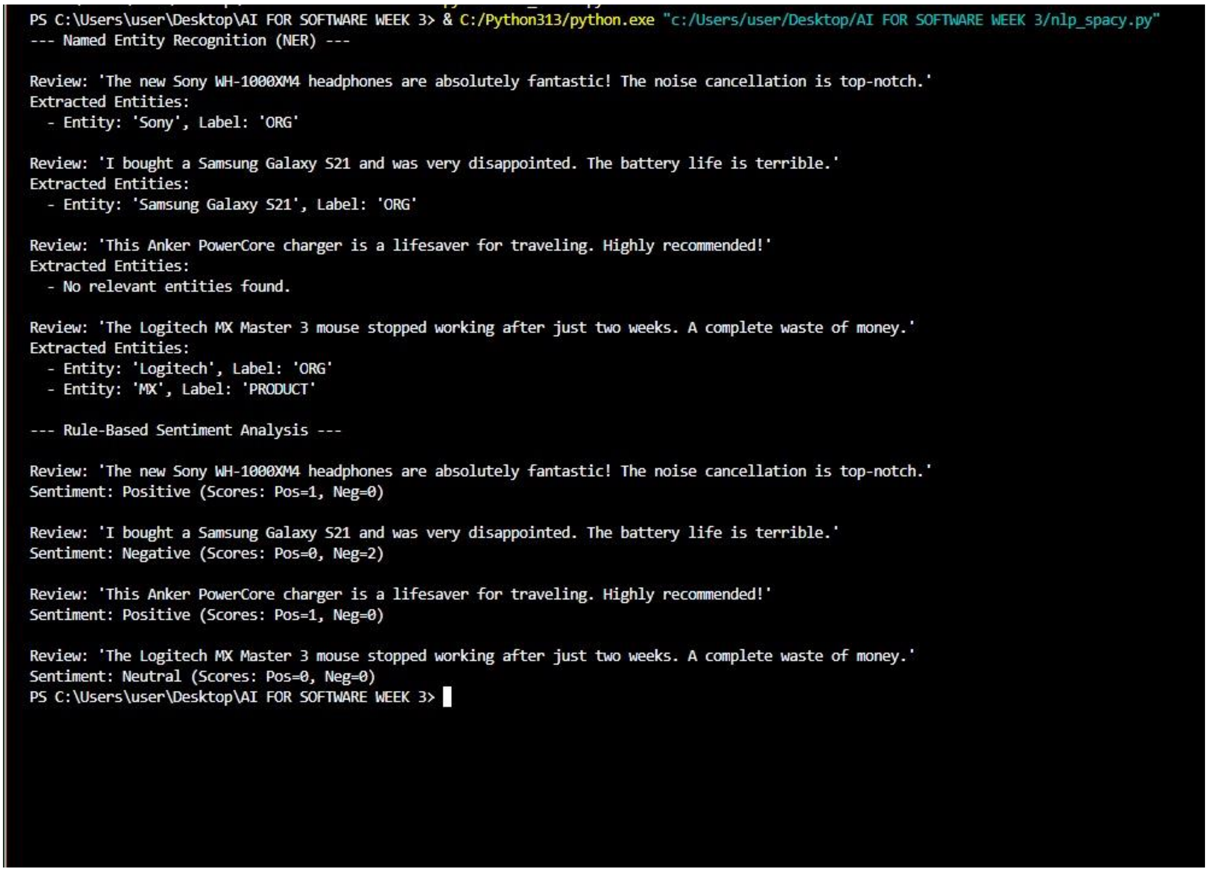
Objective:
Perform Named Entity Recognition (NER) and simple sentiment analysis on product reviews.

Approach:

- Load spaCy model: en_core_web_sm.
- Extract entities of type PRODUCT and ORG.
- Applied rule-based sentiment analysis (count positive/negative words).

Example Output:

Review: “The Samsung Galaxy earbuds have amazing sound quality!”
Entities: [('Samsung Galaxy', 'PRODUCT')]
Sentiment: Positive



```
PS C:\Users\user\Desktop\AI FOR SOFTWARE WEEK 3> & C:/Python313/python.exe "c:/Users/user/Desktop/AI FOR SOFTWARE WEEK 3/nlp_spacy.py"
--- Named Entity Recognition (NER) ---

Review: 'The new Sony WH-1000XM4 headphones are absolutely fantastic! The noise cancellation is top-notch.'
Extracted Entities:
  - Entity: 'Sony', Label: 'ORG'

Review: 'I bought a Samsung Galaxy S21 and was very disappointed. The battery life is terrible.'
Extracted Entities:
  - Entity: 'Samsung Galaxy S21', Label: 'ORG'

Review: 'This Anker PowerCore charger is a lifesaver for traveling. Highly recommended!'
Extracted Entities:
  - No relevant entities found.

Review: 'The Logitech MX Master 3 mouse stopped working after just two weeks. A complete waste of money.'
Extracted Entities:
  - Entity: 'Logitech', Label: 'ORG'
  - Entity: 'MX', Label: 'PRODUCT'

--- Rule-Based Sentiment Analysis ---

Review: 'The new Sony WH-1000XM4 headphones are absolutely fantastic! The noise cancellation is top-notch.'
Sentiment: Positive (Scores: Pos=1, Neg=0)

Review: 'I bought a Samsung Galaxy S21 and was very disappointed. The battery life is terrible.'
Sentiment: Negative (Scores: Pos=0, Neg=2)

Review: 'This Anker PowerCore charger is a lifesaver for traveling. Highly recommended!'
Sentiment: Positive (Scores: Pos=1, Neg=0)

Review: 'The Logitech MX Master 3 mouse stopped working after just two weeks. A complete waste of money.'
Sentiment: Neutral (Scores: Pos=0, Neg=0)
PS C:\Users\user\Desktop\AI FOR SOFTWARE WEEK 3>
```

Part 3: Ethics & Optimization

1. Ethical Considerations

- **Bias in MNIST:**
  MNIST dataset may lack diversity in handwriting styles (e.g., regional scripts), causing bias toward certain populations.
  Mitigation: Use tools like **TensorFlow Fairness Indicators** to evaluate model bias across groups.
- **Bias in Amazon Reviews:**
  Language bias (e.g., slang or tone differences) may affect sentiment accuracy.
  Mitigation: Integrate **spaCy rule-based corrections** or fine-tune models with balanced multilingual data.

2. Troubleshooting Challenge

Common TensorFlow Bugs and Fixes:

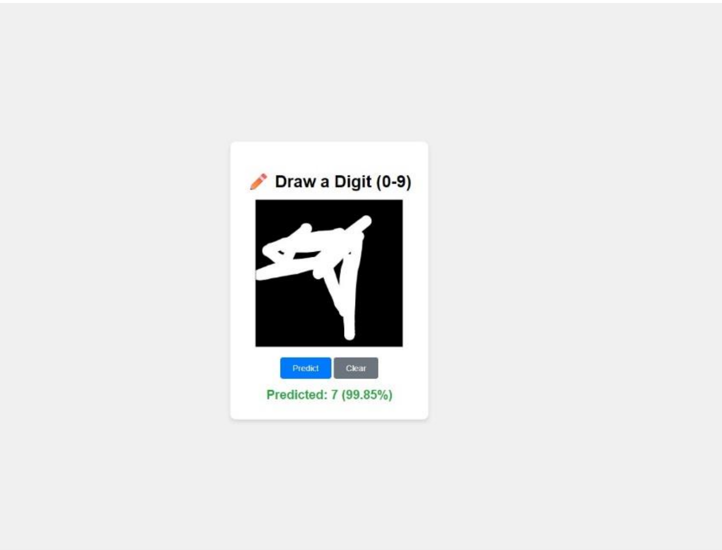
Issue	Cause	Fix
Shape mismatch error	Input tensor shape doesn't match layer definition	Adjust input_shape in first layer
Loss function mismatch	Wrong loss for classification	Use sparse_categorical_crossentropy for integer labels
Overfitting	Too many epochs or high model capacity	Add dropout, regularization, or reduce epochs

Model Deployment

Tool: FLASK APP
Goal: Deploy the MNIST model as an interactive web app where users can draw digits and see predictions in real-time.

Deployment Steps:

1. Export trained TensorFlow model (.h5).
2. Create app.py using FLASK APP.
3. Deploy on FLASK APP Cloud or Hugging Face Spaces.



References

- TensorFlow Documentation: https://www.tensorflow.org
- PyTorch Documentation: https://pytorch.org
- Scikit-learn User Guide: https://scikit-learn.org/stable/
- spaCy Official Docs: https://spacy.io
- MNIST Dataset: http://yann.lecun.com/exdb/mnist/