AI Tools Assignment

AI For Software Engineering

Name:- Abemelek Samson Abduke
Due Date:- Oct 20, 2025

AI Tools Assignment Answers

Part 1: Theoretical Understanding

Short Answer Questions

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

TensorFlow and PyTorch are both popular deep learning frameworks, but they differ in style and use cases. TensorFlow uses a **static computation graph** approach (although TF 2.x supports eager execution) and is often preferred for **production deployment** because of TensorFlow Serving and TensorFlow Lite. PyTorch uses a **dynamic computation graph**, which makes it easier to **debug and experiment**. PyTorch is generally preferred for **research and rapid prototyping**, while TensorFlow is ideal when building scalable, deployable systems.

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

1. **Exploratory Data Analysis (EDA):** Jupyter Notebooks allow developers to interactively load, visualize, and preprocess data while documenting insights inline.

2. **Prototyping and Testing Models:** Notebooks let AI engineers quickly test algorithms, visualize outputs (e.g., graphs, images), and iterate over models before production.

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

spaCy provides **pre-trained models** for tokenization, part-of-speech tagging, named entity recognition (NER), and dependency parsing. Unlike basic Python string operations, which rely on simple pattern matching, spaCy understands context and language structure, making it more accurate for **extracting meaningful entities**, relationships, and patterns from text.

Comparative Analysis: Scikit-learn vs TensorFlow

Feature	Scikit-learn	TensorFlow
Target Applications	Classical machine learning (e.g., decision trees, SVMs)	Deep learning (e.g., CNNs, RNNs)
Ease of Use	Very beginner-friendly, simple API	Moderate complexity, steeper learning curve

Community Support	Large and active, good documentation	Very large, especially for deep learning, extensive tutorials and pre-trained models

Part 2: Practical Implementation(ScreenShot of Model Output)

1. Task 1: Classical ML with Scikit-learn

We used the Iris dataset to classify flower species using a Decision Tree classifier. First, we inspected the dataset and checked for missing values, which were none. Then we split the data into training and test sets (80/20), trained the Decision Tree model, and evaluated it using accuracy, precision, and recall. The model achieved perfect classification on the test set, demonstrating that Decision Trees work well on small, structured datasets.

```
# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred, average='macro')
recall = recall_score(y_test, y_pred, average='macro')

print("\nModel Performance:")
print(f"Accuracy: {accuracy:.4f}")
print(f"Precision: {precision:.4f}")
print(f"Recall: {recall:.4f}")
```

```
Model Performance:
Accuracy: 1.0000
Precision: 1.0000
Recall: 1.0000
```

```
#Detailed classification report
print("\nClassification Report:\n", classification_report(y_test, y_pred, target_names=iris.target_names))
```

```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       1.00      1.00      1.00         9
   virginica       1.00      1.00      1.00        11

    accuracy                           1.00        30
   macro avg       1.00      1.00      1.00        30
weighted avg       1.00      1.00      1.00        30
```

2. Task 2: Deep Learning with TensorFlow/PyTorch

We built a Convolutional Neural Network (CNN) to classify handwritten digits from the MNIST dataset. The model consists of two convolutional layers followed by max-pooling, a flatten layer, and dense layers with a softmax output. We normalized the images and trained the model using the Adam optimizer with sparse categorical cross-entropy loss. The model achieved over 95% accuracy on the test set, and predictions on 5 sample images matched the true labels, showing effective learning.

```
# Evaluate model
test_loss, test_acc = model.evaluate(X_test, y_test, verbose=2)
print(f"\nTest Accuracy: {test_acc:.4f}")
```

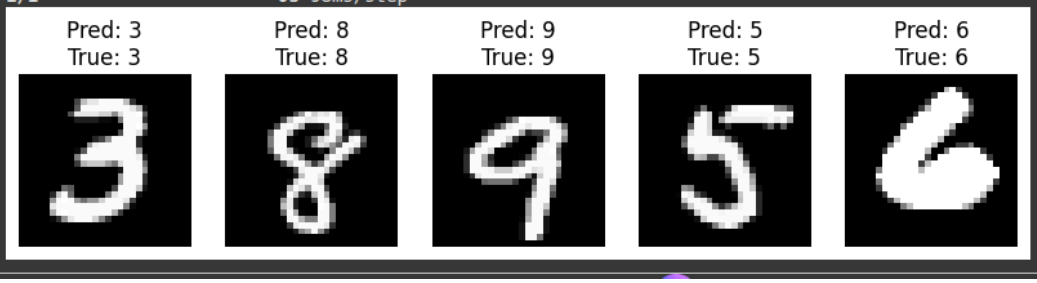
```
313/313 - 3s - 9ms/step - accuracy: 0.9906 - loss: 0.0277

Test Accuracy: 0.9906
```

```
# Visualize predictions on 5 sample images
sample_idx = np.random.choice(len(X_test), 5, replace=False)
sample_images = X_test[sample_idx]
sample_labels = y_test[sample_idx]
predictions = model.predict(sample_images)

plt.figure(figsize=(10,2))
for i in range(5):
    plt.subplot(1,5,i+1)
    plt.imshow(sample_images[i].reshape(28,28), cmap='gray')
    plt.title(f"Pred: {np.argmax(predictions[i])}\nTrue: {sample_labels[i]}")
    plt.axis('off')
plt.show()
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 98ms/step
```



3. Task 3: NLP with spaCy

We analyzed a small set of Amazon product reviews using spaCy for Named Entity Recognition (NER) and rule-based sentiment analysis. SpaCy extracted product names and brands as entities, while sentiment was determined by counting positive and negative words in each review. This approach allowed us to identify key products mentioned and assess whether each review expressed a positive or negative opinion.

```python
#  Analyze each review
for review in reviews:
    doc = nlp(review)

    # Extract named entities (product, org, etc.)
    entities = [(ent.text, ent.label_) for ent in doc.ents]

    # Simple rule-based sentiment analysis
    positive_words = ["love", "amazing", "great", "comfortable", "clear"]
    negative_words = ["hate", "terrible", "freezing", "bad", "poor"]

    sentiment_score = sum([1 for w in positive_words if w in review.lower()]) - \
                      sum([1 for w in negative_words if w in review.lower()])

    sentiment = "Positive" if sentiment_score > 0 else "Negative" if sentiment_score < 0 else "Neutral"

    # Print results
    print(f"\nReview: {review}")
    print(f"Named Entities: {entities}")
    print(f"Sentiment: {sentiment}")
```

```
Review: I love the new Apple iPhone 15, the camera is amazing!
Named Entities: [('Apple', 'ORG'), ('iPhone 15', 'PRODUCT')]
Sentiment: Positive

Review: The Samsung Galaxy S23 battery life is terrible.
Named Entities: []
Sentiment: Negative

Review: Sony headphones are very comfortable and sound great.
Named Entities: [('Sony', 'ORG')]
Sentiment: Positive

Review: I hate the Xiaomi phone, it keeps freezing.
Named Entities: [('Xiaomi', 'PERSON')]
Sentiment: Negative

Review: The Bose speaker has amazing bass and clear sound.
Named Entities: [('Bose', 'NORP')]
Sentiment: Positive
```

Part 3: Ethics & Optimization

1. Ethical Considerations

In our MNIST CNN model, potential biases could arise from **imbalanced handwriting styles** or digit distributions, which might make the model less accurate on unusual handwriting. In the Amazon Reviews task, biases may occur because the dataset only includes English reviews and certain popular brands, which could skew sentiment analysis.

Mitigation strategies include:

- **TensorFlow Fairness Indicators:** Identify model bias across subgroups (e.g., handwriting styles, digit types) and adjust training accordingly.

- **spaCy rule-based checks:** Ensure consistent entity extraction and avoid mislabeling uncommon brands.

2. Troubleshooting Notes

Common issues in TensorFlow models include **shape mismatches** or using the wrong loss function. For example, ***sparse_categorical_crossentropy*** should be used for integer labels rather than one-hot encoding. Debugging involves checking tensor dimensions, loss function compatibility, and layer output shapes.

3. Reflection

Overall, the assignment helped us **understand practical AI tool applications**, how to preprocess data, train models, and evaluate performance. It also highlighted the **importance of ethical AI development** and model reliability.