AI Tools and Frameworks – Week 3 Assignment

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

Group 43 AI project comprisedTensorFlow-based convolutional neural network (CNN) for classifying handwritten digits (MNIST dataset) and a spaCy-based model for named entity recognition (NER) and sentiment analysis on Amazon Product Reviews. Tensor Flow and PyTorch differ mainly in how they build and execute computation graphs. For example, Tensor Flow initially used static graphs executed through sessions, but currently provides eager execution like PyTorch. On the other hand, PyTorch uses dynamic computation graphs by default. PyTorch computational elements are more intuitive for developers.

Tensor Flow CNN Model Accuracy Graph

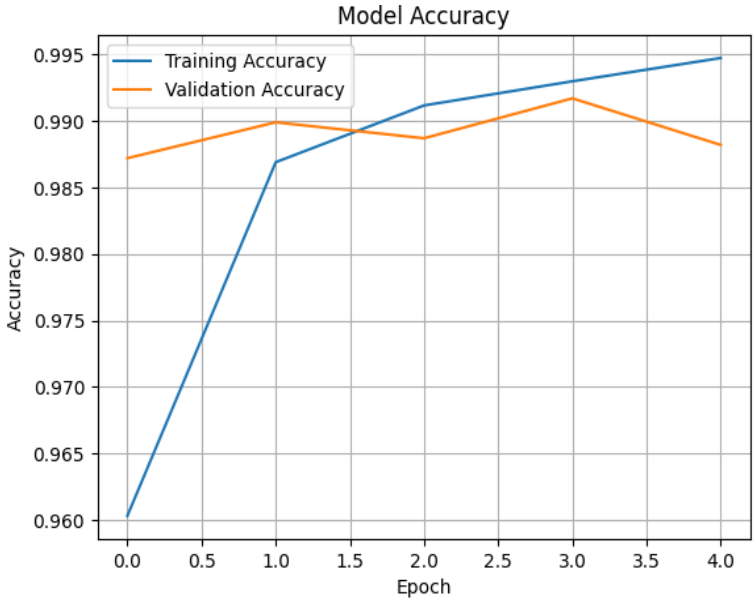


Figure 1: Accuracy graph for the tensor flow CNN model over 5 epochs

Figure 1 shows a visualization of training and validation. The X-axis (epoch) shows the training cycles throughout the training, while the Y-axis shows the proportion of classified images. It ranges between 0.96 and 0.995. The blue line shows that the training model is learning well as it begins at 0.96 and increases steadily, almost reaching 99.5% at epoch 5. The orange line, which starts at 0.987 is quite higher in the first epoch, peaks at 3, and drops epoch 4. The

validation scale suggests slight overfitting after epoch 3. This suggests that the model continues to learn based on the training data, but performance with unseen data slightly drops. Generally, the CNN model is performing well.

When to choose:

Due to its Pythonic syntax and flexibility, PyTorch should be chosen for research and rapid prototyping. Choosing TensorFlow is useful in scalable deployment, particularly in production environments such as using TensorFlow Serving or Lite.

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

- Experimentation & Prototyping – Developers can test small chunks of code, view output immediately, and explain code logic with markdown cells.
- Data Analysis & Visualization – Useful for cleaning data, visualizing it with libraries like Matplotlib/Seaborn, and understanding trends before training models.

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

spaCy offers advanced NLP features that outperform basic string functions:

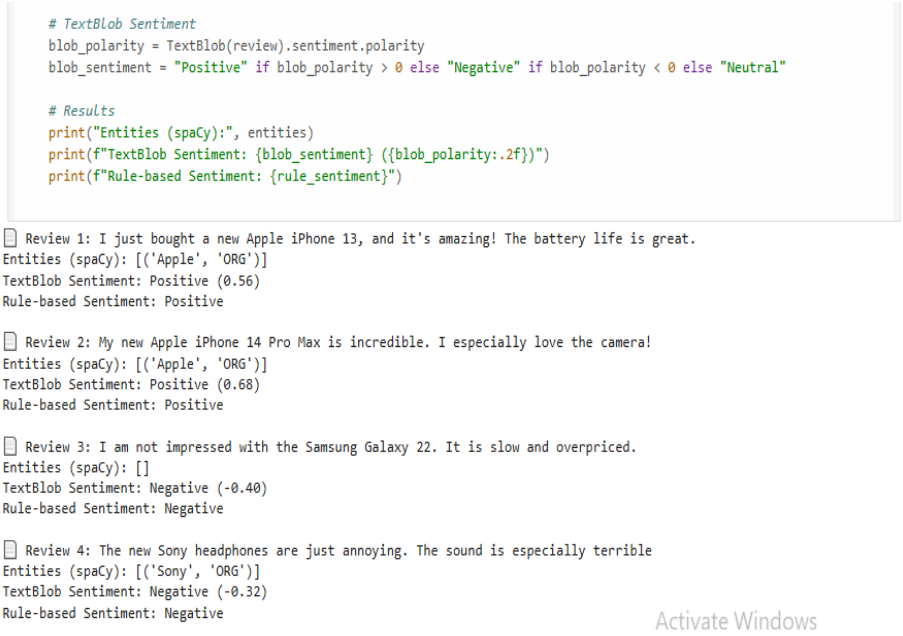
- Efficient Tokenization – spaCy understands punctuation, contractions, and special word cases.
- Entity Recognition & POS Tagging – It identifies parts of speech, names, and entities like organizations or locations, which basic Python cannot do.

Sentiment Analysis via SpaCy

Figure 2 below shows sentiment analysis based on rule-based method via keyword matching and TextBlob. TextBlob employs a pre-trained model to compute polarity scores from range (-1), showing very negative and (+1) showing very positive. The sentiments are labeled as either positive or negative depending on polarity value. Key advantages of this model include detection of subtle sentiments and consideration of context, and syntax. However, there are risks of overgeneralization and may miss domain specific aspects.

When it comes to rule-based approach, the model checks specific keywords, which are predefined as positive or negative and matches the sentiments accordingly. The rule-based

approach is fast, transparent, and customizable based on domains. Still, context is a key consideration when it comes to sentiment analysis.



```
    # TextBlob Sentiment
    blob_polarity = TextBlob(review).sentiment.polarity
    blob_sentiment = "Positive" if blob_polarity > 0 else "Negative" if blob_polarity < 0 else "Neutral"

    # Results
    print("Entities (spaCy):", entities)
    print(f"TextBlob Sentiment: {blob_sentiment} ({blob_polarity:.2f})")
    print(f"Rule-based Sentiment: {rule_sentiment}")
```

```
Review 1: I just bought a new Apple iPhone 13, and it's amazing! The battery life is great.
Entities (spaCy): [('Apple', 'ORG')]
TextBlob Sentiment: Positive (0.56)
Rule-based Sentiment: Positive

Review 2: My new Apple iPhone 14 Pro Max is incredible. I especially love the camera!
Entities (spaCy): [('Apple', 'ORG')]
TextBlob Sentiment: Positive (0.68)
Rule-based Sentiment: Positive

Review 3: I am not impressed with the Samsung Galaxy 22. It is slow and overpriced.
Entities (spaCy): []
TextBlob Sentiment: Negative (-0.40)
Rule-based Sentiment: Negative

Review 4: The new Sony headphones are just annoying. The sound is especially terrible
Entities (spaCy): [('Sony', 'ORG')]
TextBlob Sentiment: Negative (-0.32)
Rule-based Sentiment: Negative
```


Figure 2: Sentiment analysis via spacy model

Part 2: Comparative Analysis

Scikit-learn and TensorFlow Comparison		
Feature	**Scikit-learn**	**TensorFlow**
Target Applications	Classical Machine Learning, such as SVM	Deep Learning, such as CNN
Ease of Use	Simple API that is beginner-friendly	Customizable, but the learning curve is steeper
Community Support	Strong for traditional ML, mature documentation	Wide support, active development by Google

Ethical Reflection: Mitigating Biases in Our AI Models

Our group's AI project comprises a TensorFlow-based convolutional neural network (CNN) for classifying handwritten digits (MNIST dataset) and a spaCy-based model for named entity recognition (NER) and sentiment analysis on Amazon Product Reviews. Both models present ethical challenges related to bias and privacy, which we address using TensorFlow Fairness Indicators and spaCy's rule-based systems to promote fair and responsible AI deployment.For the MNIST model, achieving over 95% accuracy, a primary concern is bias in the training data. The MNIST dataset, sourced mainly from American contributors, may underrepresent diverse handwriting styles, such as those from non-Western cultures (e.g., unique "7" shapes in certain scripts) or individuals with motor impairments. Our code's visualization of predictions on five test images may reveal misclassifications (e.g., "7" mistaken for "1"), indicating potential bias. Such errors could disadvantage users in applications like automated banking form processing, undermining inclusivity. To mitigate this, propose using TensorFlow Fairness Indicators to evaluate model performance across hypothetical subgroups (e.g., handwriting styles by cultural or demographic factors). By analyzing fairness metrics like false positive rates, we can identify disparities and retrain the model with augmented data, such as diverse numeral samples. However, acquiring such data is resource-intensive, necessitating future efforts to enhance dataset diversity.

For the Amazon Reviews model, biases in NER and sentiment analysis are significant. Our spaCy en_core_web_sm model identifies entities like "Apple" and "iPhone 13" but may miss niche or non-English brand names (e.g., "Xiaomi"), due to its English-centric training, potentially marginalizing diverse products. TextBlob's polarity-based sentiment analysis oversimplifies nuanced feedback (e.g., missing mixed sentiments like "good but overpriced"), skewing insights toward English-speaking reviewers. Additionally, NER risks extracting sensitive information (e.g., names in reviews), raising privacy concerns. We address these spaCy's rule-based systems. Custom Matcher rules can enhance NER by recognizing niche

brands, while rules to mask sensitive entities (e.g., PERSON) protect privacy. Refined sentiment patterns improve accuracy, though extensive testing is needed to cover diverse reviews.

In conclusion, addressing biases in our models enhances their fairness and trustworthiness. TensorFlow Fairness Indicators and spaCy's rule- based systems offer practical solutions, but ongoing evaluation is critical to ensure equitable and privacy-respecting outcomes in real-world applications.