AI for Software Engineering Group

WEEK 3 ASSIGNMENT

Part 1

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

TensorFlow is Machine Learning & deep learning library developed by Google.

TensorFlow makes it easy to create Machine Learning Models that can run in any environment. They are used to advance research and build AI powered applications. Examples of TensorFlow include; Google translate, AIR BNB, Healthcare

PyTorch is a python library designed for Machine Learning (ML) AND Deep Learning AI.

It helps developers train, build and test neural networks. PyTorch is used by developers to learn and do research. Examples of PyTorch include; ChatGPT, Tesla (self-driving), Facebook.

Primary Differences

Feature	TensorFlow	PyTorch
Programming Style and Syntax	Uses static computation graphs, requiring users to define the entire computation graph before running it.	Employs dynamic computation graphs (define-by-run), allowing users to build and modify the graph on the fly during execution.
Debugging and Development Experience	Debugging can be more challenging, especially in static graph mode, as errors may only appear during graph execution.	Easier to debug due to its dynamic nature. Standard Python debugging tools (like pdb) can be used directly.
Community and Ecosystem	Backed by Google, TensorFlow has a large ecosystem, including TensorFlow Lite (for mobile), TensorFlow Serving (for deployment), and TensorFlow	Supported by Meta (Facebook), PyTorch has gained significant traction in the research community. It is widely used for

	Extended (for production pipelines).	academic research and rapid prototyping.
Deployment and Production	Offers robust tools for deploying models in production environments, including TensorFlow Serving, TensorFlow Lite, and TensorFlow.js.	Historically focused on research, but recent tools like TorchServe and TorchScript have improved its production capabilities.
Performance and scalability	Optimized for large-scale deployments and distributed training. It supports deployment on various platforms and hardware accelerators.	Also supports distributed training and GPU acceleration, but TensorFlow has traditionally been preferred for very large-scale production systems.
Model Export and Interoperability	Models can be easily exported and deployed across different platforms.	TorchScript allows models to be serialized and run independently from Python, but interoperability is generally considered more mature in TensorFlow.

The choice between TensorFlow and PyTorch depends on project requirements, team expertise, and the intended use case (research vs. production). Below is a clear breakdown on the same.

Choose TensorFlow if:

- The project requires robust production deployment tools and scalability.
- You're building models ready for production
- You want access to TensorFlow’s ecosystem
- There is a need for cross-platform deployment (mobile, web, embedded).

Choose PyTorch if:

- The focus is on research, experimentation, or rapid prototyping.
- You prefer Pythonic syntax.
- You need flexibility for models that change during runtime

- Dynamic computation graphs and intuitive debugging are important.
- The project involves cutting-edge deep learning research or collaboration with the academic community.

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

- **Interactive Data Exploration and Visualization**

Jupyter Notebooks provide an interactive environment for exploring and visualizing datasets. Data scientists and AI practitioners can load data, perform preprocessing, and generate visualizations such as histograms, scatter plots, and heatmaps within the same document. It is also useful in running step-by-step code blocks to inspect data distributions, missing values, and correlations.

- **Prototyping and Experimentation with Machine Learning Models**

Jupyter Notebooks are widely used for prototyping machine learning models. Developers can write, test, and modify code in small, manageable cells, enabling quick experimentation with different algorithms, hyperparameters, and feature engineering techniques. Results, including metrics and visualizations, can be displayed inline, facilitating comparison and documentation of experiments. This makes Jupyter Notebooks ideal for collaborative research and sharing reproducible AI workflows.

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

Linguistic Awareness

- **spaCy**: Provides advanced linguistic features such as tokenization, part-of-speech tagging, lemmatization, named entity recognition, and dependency parsing. These features allow for a deeper understanding of language structure and meaning.
- **Basic Python String Operations**: Limited to simple manipulations like splitting, joining, replacing, or searching for substrings, without any understanding of grammar or context.

Tokenization

- **spaCy**: Accurately splits text into words, punctuation, and sentences, handling edge cases like contractions, abbreviations, and special characters.
- **Basic Python String Operations**: Relies on simple delimiters such as punctuation, which can lead to incorrect token boundaries and loss of information.

Named Entity Recognition (NER)

- **spaCy**: Automatically identifies and categorizes entities such as people, organizations, locations, dates, and more within text.
- **Basic Python String Operations**: Cannot recognize or classify entities without custom,

Part of Speech Tagging and Lemmatization

- **spaCy**: Assigns grammatical roles (nouns, verbs, adjectives, etc.) and reduces words to their base forms, enabling more accurate text analysis.
- **Basic Python String Operations**: Lacks the ability to analyze grammatical structure or normalize words.

Efficiency and Scalability

- **spaCy**: Optimized for speed and large-scale text processing, supporting multi-threading and efficient memory usage.
- **Basic Python String Operations**: Not optimized for large datasets or complex linguistic tasks.

Pre-trained Models and Language Support

- **spaCy**: Offers pre-trained models for multiple languages, enabling out-of-the-box support for various NLP tasks. SpaCy enhances NLP by adding intelligence, structure and meaning. It reads text, understands it, it structures sentence and speech. It also interprets language
- **Basic Python String Operations**: No built-in language models or support for advanced NLP tasks.

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 and Community support.

Feature	Scikit-learn	TensorFlow
Target Application	Designed for Classical ML algorithms (regression, classification, clustering)	Deep learning and neural-network based models including CNNs, RNNs and transformers. Ideal for complex tasks such as image recognition, natural language processing, and large-scale unstructured data analysis.
Ease of use	Beginner friendly, simple API	Steeper learning curve improved with keras
Community Support	Strong active open-source community with extensive documentation, tutorials and example	Large global community. Availability of tools for deployment, mobile and production environment

Part 3: Ethics and Optimization

<u>Amazon Reviews Biases</u>

These reviews include but are not limited to the below mentioned

- Positivity Bias

A lot of people only review products and/or services they are highly satisfied with. This can skew the results of the dataset

- Temporal Bias

This is a result of change in quality; there may be a decline or increase a quality which may be affected by previous reviews.

- Reviewer bias

Some reviewers are generous while others are hash with the reviews.

- Fake Bias

People, particularly influencers can be paid for reviews then give fake reviews, bots can also be used.

Potential bias in MNIST dataset

Overfitting to clean- The model learns to rely on the neatness and central positioning of digits

Poor generalization to real-world handwriting - It may fail when digits are off-center, rotated, messy, or written by children or elderly

Real world mismatch

Baggy Code

This original code which mixed two import styles: import tensorflow as tf and from tensorflow import keras, resulted in an underlining error on VS Code. This resulted in errors because confusion because keras was not explicitly recognized and also because TensorFlow's submodules was not properly indexed by the IDE. This code also didn't expose layers needed for building CNNs directly.

```
import tensorflow as tf
from tensorflow import keras

print(tf.__version__)
```

Debugged Code

This fixed the code and improved readability, IDE compatibility, and debugging clarity. It avoided ambiguous imports and ensured that all TensorFlow components are accessed through the tf namespace.

```
#imports
import tensorflow as tf
layers = tf.keras.layers
models = tf.keras.models
import matplotlib.pyplot as plt
```