

AI Tools and Frameworks – Assignment Report

1. 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 powerful deep learning frameworks, but they differ mainly in style and usability. TensorFlow is more production-oriented, great for large-scale deployment, and integrates smoothly with TensorFlow Serving and TensorFlow Lite. PyTorch, on the other hand, feels more 'Pythonic' and intuitive for research — its dynamic computation graph makes experimentation faster and debugging easier. I'd pick PyTorch for quick prototyping or research projects, and TensorFlow for enterprise-level apps where scalability and deployment really matter.

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

First, Jupyter Notebooks are perfect for exploratory data analysis — they let you visualize, test, and tweak data pipelines interactively. Second, they're great for model prototyping and documentation, since you can mix code, markdown notes, and results in one place, which makes collaboration and reproducibility much easier.

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

spaCy brings industrial-strength NLP tools that go way beyond simple string manipulation. Instead of manually splitting and counting words, spaCy provides tokenization, part-of-speech tagging, named entity recognition, and dependency parsing — all optimized for performance. It's basically a full linguistic engine under the hood, not just a set of string tricks.

2. Comparative Analysis: Scikit-learn vs. TensorFlow

Target Applications:

Scikit-learn focuses on classical machine learning — algorithms like regression, classification, clustering, and dimensionality reduction. TensorFlow targets deep learning — building and training neural networks for things like image recognition, NLP, and reinforcement learning.

Ease of Use:

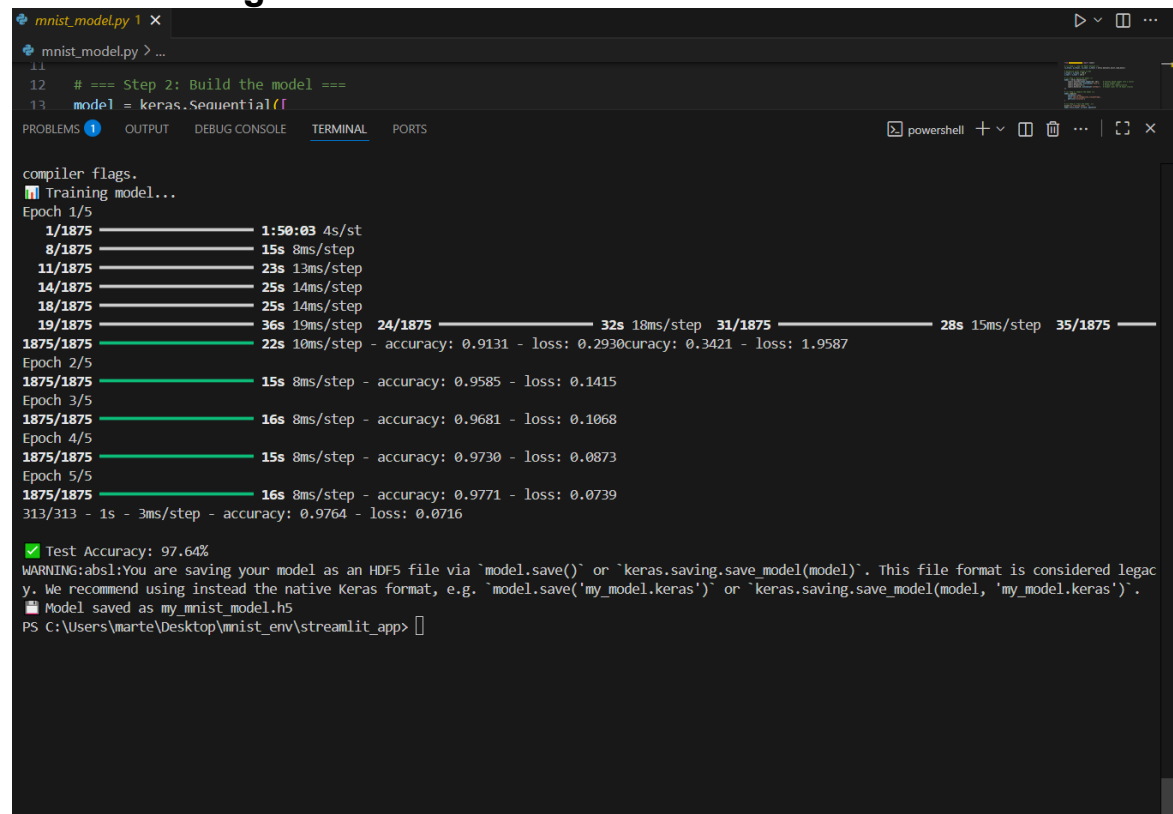
Scikit-learn is simpler and beginner-friendly, with a very consistent API (fit, predict, score). TensorFlow, while more powerful, can feel heavier for newcomers due to its complexity and broader scope.

Community Support:

Both have strong communities. Scikit-learn shines in the academic and data science space, while TensorFlow has massive industry support, tutorials, and tools for scaling ML pipelines.

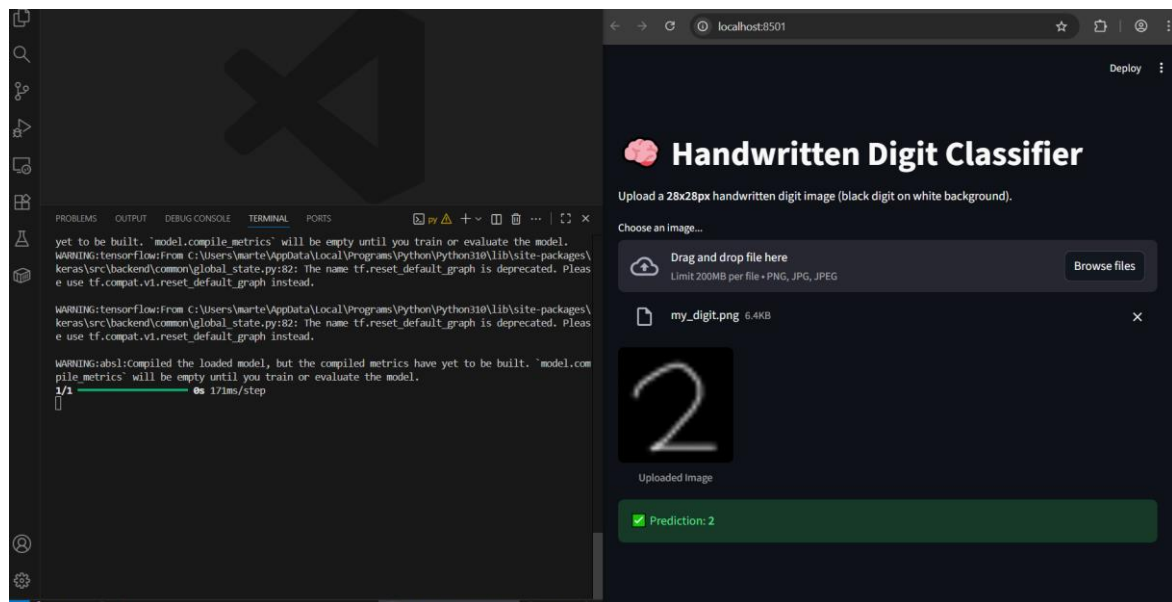
3. Screenshots of model outputs

Model training stats

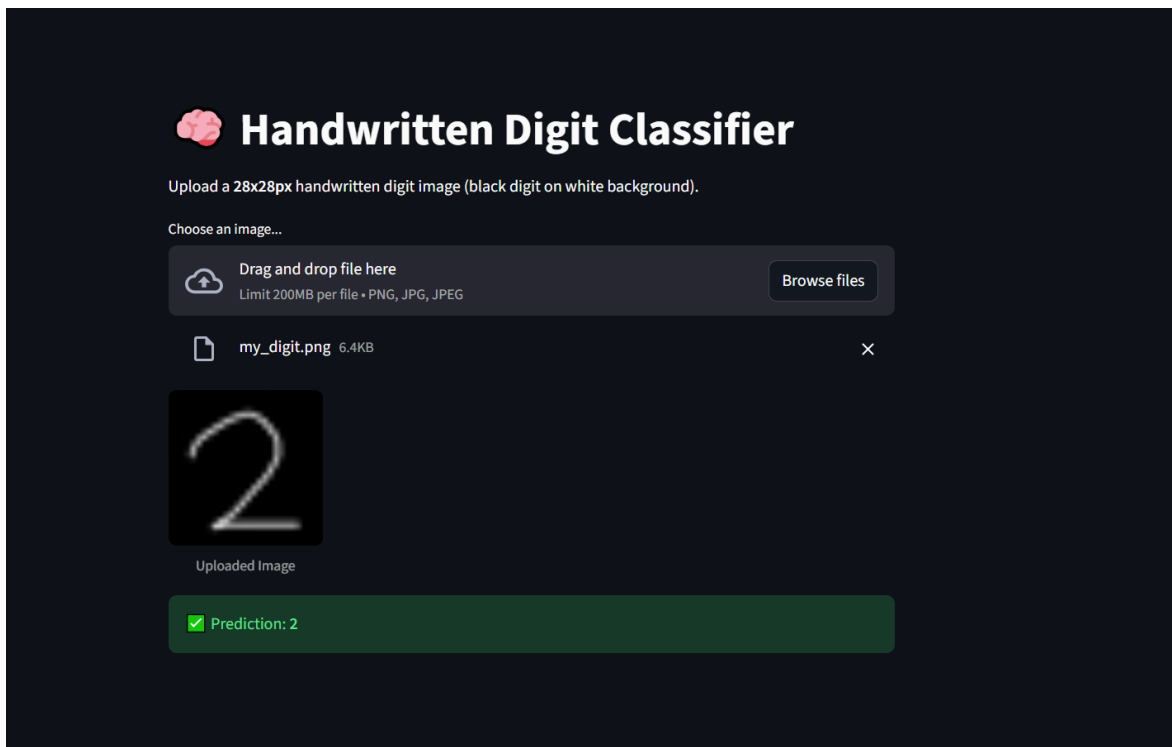


```
mnist_model.py > ...  
11  
12 # === Step 2: Build the model ===  
13 model = keras.Sequential([  
    compiler flags.  
    Training model...  
Epoch 1/5  
1/1875 ----- 1:50:03 4s/st  
8/1875 ----- 15s 8ms/step  
11/1875 ----- 23s 13ms/step  
14/1875 ----- 25s 14ms/step  
18/1875 ----- 25s 14ms/step  
19/1875 ----- 36s 19ms/step 24/1875 ----- 32s 18ms/step 31/1875 ----- 28s 15ms/step 35/1875 -----  
1875/1875 ----- 22s 10ms/step - accuracy: 0.9131 - loss: 0.2930 accuracy: 0.3421 - loss: 1.9587  
Epoch 2/5  
1875/1875 ----- 15s 8ms/step - accuracy: 0.9585 - loss: 0.1415  
Epoch 3/5  
1875/1875 ----- 16s 8ms/step - accuracy: 0.9681 - loss: 0.1068  
Epoch 4/5  
1875/1875 ----- 15s 8ms/step - accuracy: 0.9730 - loss: 0.0873  
Epoch 5/5  
1875/1875 ----- 16s 8ms/step - accuracy: 0.9771 - loss: 0.0739  
313/313 - 1s - 3ms/step - accuracy: 0.9764 - loss: 0.0716  
  
✓ Test Accuracy: 97.64%  
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.save_model(model)`. This file format is considered legac  
y. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.save_model(model, 'my_model.keras')`.  
Model saved as my_mnist_model.h5  
PS C:\Users\marie\Desktop\mnist_env\streamlit_app> |
```

Model testing and integration



Model deployment



[View Live Model](#)

4. Ethical reflection

Even though the MNIST dataset seems neutral, it can still have bias. Most of the handwriting samples come from similar groups of people, so the model might not perform as well on digits written by others with different writing styles. That means it could struggle in real-world cases where handwriting varies a lot.

Also, the CNN model works well, but it's not easy to understand why it makes certain predictions. Tools like TensorFlow Fairness Indicators can help check for unfair patterns or performance gaps and guide improvements.

Building fair models isn't just about accuracy. It's about making sure the system treats all users fairly and works well for everyone.