Jassey-Jabarr Fisayo

ITAI 3377: A.I. at the Edge & IIOT

Professor: Patricia McManus

February 4, 2025

ITAI 3377: Conceptual AI Model Development and Deployment

**Executive Summary:** This document outlines the complete process of developing and deploying a handwritten digit recognition AI model, from initial setup to final deployment on edge devices. My approach is to do a summary report and then detail technical step by step.

Part 1: Technical Implementation Overview - Environment Setup

- Python 3.9+ installation with TensorFlow compatibility

- TensorFlow/TensorFlow Lite configuration

- Jupyter Notebook integration

- Key verification steps for each component

**Model Development**

- MNIST dataset utilization (70,000 total images)

- Neural network architecture:

- Input: 28x28 grayscale images

- Hidden layers: Pattern recognition system

- Output: 10-digit classification

**Model Optimization**

- TensorFlow Lite conversion process

- Size reduction: 100MB → 25MB

- Performance optimization balancing size/speed/accuracy

- Deployment preparation for edge devices

**Deployment Strategy**

- Edge device implementation

- Input processing pipeline

- Real-time testing procedures

- Error handling and optimization

Reflective Journal

Challenges Faced

1. Technical Understanding

- Initially struggled with understanding neural network architecture

- Faced complexity in optimization choices

- Challenged by balancing performance vs resource usage

2. Conceptual Hurdles

- Grasping the relationship between model size and accuracy

- Understanding the importance of data preprocessing

- Visualizing the conversion process

Learning Outcomes

1. Technical Knowledge

- Understood complete AI development pipeline

- Learned about model optimization techniques

- Gained insights into edge deployment challenges

2. Practical Skills

- Environment setup procedures

- Model conversion strategies

- Deployment considerations

Real-World Applications

1. Immediate Applications

- Document digitization projects

- Mobile app development

- Edge device implementations

2. Future Opportunities

- Custom AI model development

- Mobile-first AI solutions

- Enterprise edge computing projects

Understanding & Growth

Through this assignment, I've gained a comprehensive understanding of the AI model deployment pipeline. The most valuable insight was learning how theoretical concepts translate into practical implementation decisions, especially in resource-constrained environments. This knowledge will be crucial for future projects involving edge AI deployment.

Thank you.

**Technical Report**

**The step by step activities of the report above**

1. Python Installation Process - Python 3.9+?

- I ensures compatibility with TensorFlow and modern AI libraries

Installation Steps:

Download Python:

- Visit python.org/downloads

- Select Python 3.9 or later version

- Choose appropriate installer for your OS (Windows/Mac/Linux)

Installation Configuration:

- Check "Add Python to PATH" ✓

- Select "Customize installation"

- Enable "pip" package installer

- Choose installation location (default recommended)

Verification:

- Open terminal/command prompt

- Type: `python --version`

- Expected output: Python 3.9.x

2. TensorFlow and TensorFlow Lite Setup

Installation Process: TensorFlow Installation: ``` pip install tensorflow==2.9.0 ```

3. TensorFlow Lite: python -c "import tensorflow as tf; print(tf.\_\_version\_\_)"

4. Jupyter Notebook Setup: Install Jupyter: pip install jupyter

Verification Tasks:

1. Create new Python 3 notebook

2. Test basic code:```python import tensorflow as tf print("Setup Complete!") ```

Building a Handwriting Recognition AI: a simple solution

Our Goal

We're building an AI that can read handwritten numbers - like reading a phone number someone wrote on a piece of paper. Think of it as teaching a computer to read handwriting, just like we learned in school!

The AI Brain (Neural Network) Imagine our AI as a brain with different layers:

1. Eyes (Input Layer)

- Looks at 28x28 pixel images (like a tiny grid)

- Each pixel is like a dot of gray

2. Pattern Recognition Layers

- First layer spots simple things (like lines and curves)

- Second layer combines these to recognize parts of numbers

- Final layer makes the actual guess: "This is a 7!"

Teaching Process

Step 1: Getting the Training Data

- We use MNIST - it's like a huge collection of handwritten numbers

- 60,000 images to learn from (like practicing with flashcards)

- 10,000 images to test how well it learned

Step 2: Preparing the Images

1. Clean the data

- Convert dark pixels to numbers between 0-1

- Like adjusting the brightness to make learning easier

2. Organize the data

- Split into "learning" and "testing" piles

- Just like how students practice with some questions and test with others

Step 3: Training

- Run through all images 10 times (epochs)

- AI makes a guess, we tell it if it's right or wrong

- It learns from mistakes and gets better

- Takes about 5 minutes to train

Expected Results

- Gets it right about 98% of the time

- Common mistakes: mixing up 4s and 9s (just like humans!)

**Convert to TensorFlow Lite**

Think of it like compressing a large video for your phone:

* Original AI model: Like a 4K movie (big, needs powerful computer)
* TFLite model: Like a compressed video (runs smoothly on phone)

**The Conversion Process**

**Step 1: Prepare the Model**

* Check our trained model works correctly
* Like testing a recipe before sharing it

**Step 2: Convert the Model**

Simple process:

1. Create converter
   * Tell it which model to convert
   * Like selecting which video to compress
2. Set Optimization Options
   * Choose how much to compress
   * Options:
     + DEFAULT: Good balance
     + OPTIMIZE\_FOR\_SIZE: Smaller file
     + OPTIMIZE\_FOR\_LATENCY: Faster running
3. Pick Optimization Level
   * Like choosing video quality:
     + Full size (most accurate)
     + Slightly compressed (good balance)
     + Highly compressed (fastest)

**Step 3: Save the Model**

1. Convert and save:
   * Save as '.tflite' file
   * Usually 3-4 times smaller than original
   * Ready for phones/small devices
2. Test the saved model:
   * Make sure it still recognizes numbers correctly
   * Might be slightly less accurate (95% instead of 98%)
   * But much faster and smaller!

**Results**

* Original model: ~100MB
* TFLite model: ~25MB
* Speed: 2-3 times faster

Putting the Model to Work: Deployment

Getting the Model Running -Setup Steps

1. Put model on device

- Copy .tflite file to phone/device

- Like installing a new app

2. Set up interpreter

- Load model file

- Allocate memory

- Like opening the app and getting it ready

3. Prepare for input

- Set up camera/image input

- Make sure images are 28x28 pixels

- Like making sure photos are the right size

Testing Our Model

1. Take a test photo

- Write number on paper

- Take picture

- Crop to just the number

2. Process image

- Convert to grayscale

- Resize to 28x28

- Normalize pixels (0-1)

3. Get prediction

- Feed image to model

- Get top guess

- Show confidence level (%)

Common Issues:

- Bad lighting → poor recognition

- Blurry image → wrong guess

- Wrong image size → no result