Florentin Degbo

ITAI 3377

Professor Patricia McManus

**L02- Deploying an AI Model on a Simulated Edge Device 2 Options**

**Option A - Conceptual**

**Part 1: Conceptual Setup of Development Environment**

**Python Installation**

*Steps to Install Python:*

1. Download Python:
   * Visit the official Python website: [python.org](https://www.python.org/).
   * Navigate to the Downloads section and choose the appropriate version for your operating system.
2. Run the Installer:
   * Open the downloaded installer.
   * Ensure to check the box that says, "Add Python to PATH" before clicking "Install Now."
3. Verify Installation:
   * Open a command prompt or terminal.
   * Type python --version to verify the installation.

**TensorFlow and TensorFlow Lite Installation**

Steps to Install TensorFlow and TensorFlow Lite:

1. Open Command Prompt or Terminal:
   * Ensure you have Python and pip installed.
2. Install TensorFlow:
   * Run the command: pip install tensorflow
3. Install TensorFlow Lite:
   * TensorFlow Lite is included with TensorFlow, so no additional installation is needed.

**Jupyter Notebook Installation**

Steps to Install Jupyter Notebook:

1. Open Command Prompt or Terminal:
   * Ensure you have Python and pip installed.
2. Install Jupyter Notebook:
   * Run the command: pip install notebook
3. Launch Jupyter Notebook:
   * Run the command: jupyter notebook
   * This will open Jupyter Notebook in your default web browser.

**Part 2: Conceptual Creation and Training of an AI Model**

**Neural Network Model Design**

Model Architecture:

* Input Shape: (28, 28, 1) for the MNIST dataset.
* Layers:
  + Convolutional Layer: 32 filters, kernel size (3, 3), activation function ReLU.
  + MaxPooling Layer: Pool size (2, 2).
  + Flatten Layer.
  + Dense Layer: 128 units, activation function ReLU.
  + Output Layer: 10 units (for 10 classes), activation function Softmax.

**Data Loading and Preprocessing**

Steps to Load and Preprocess MNIST Dataset:

1. Load Dataset:
   * Use tf.keras.datasets.mnist.load\_data() to load the dataset.
2. Preprocess Data:
   * Normalize the pixel values to the range [0, 1] by dividing by 255.
   * Reshape the data to include the channel dimension: (28, 28, 1).

**Model Training Process**

Steps to Compile and Train the Model:

1. Compile the Model:
   * Use model.compile(optimizer='adam', loss='sparse\_categorical\_crossentropy', metrics=['accuracy']).
2. Train the Model:
   * Use model.fit(train\_images, train\_labels, epochs=5, batch\_size=64, validation\_split=0.2).

**Part 3: Conceptual Conversion and Saving of the Model**

**Model Conversion to TensorFlow Lite**

Steps to Convert the Model:

1. Save the Keras Model:
   * Use model.save('mnist\_model.h5').
2. Convert to TensorFlow Lite:
   * Use the TensorFlow Lite converter:
   * import tensorflow as tf
   * converter = tf.lite.TFLiteConverter.from\_keras\_model\_file('mnist\_model.h5')

tflite\_model = converter.convert()

1. Save the TensorFlow Lite Model:
   * Use with open('mnist\_model.tflite', 'wb') as f: f.write(tflite\_model).

**Part 4: Conceptual Deployment of the Model**

**Deployment Process**

Steps to Deploy the Model:

1. Load the TensorFlow Lite Model:
   * Use the TensorFlow Lite interpreter:
   * interpreter = tf.lite.Interpreter(model\_path='mnist\_model.tflite')

interpreter.allocate\_tensors()

1. Set Up Input and Output Tensors:
   * Get input and output tensor details using interpreter.get\_input\_details() and interpreter.get\_output\_details().
2. Run Inference:
   * Prepare the input data and invoke the interpreter:
   * interpreter.set\_tensor(input\_details[0]['index'], input\_data)
   * interpreter.invoke()

output\_data = interpreter.get\_tensor(output\_details[0]['index'])

**Model Testing on Sample Input**

Steps to Test the Model:

1. Prepare Sample Input:
   * Load and preprocess a sample image from the MNIST dataset.
2. Run Inference on Sample Input:
   * Follow the steps outlined in the deployment process to run inference on the sample input.
3. Interpret Results:
   * Analyze the output data to determine the predicted class label.

**Reflective Journal**

**Python Installation**

Researching the Python installation process was quite easy for me since I had already gone through it before. Documenting the steps was a breeze, and I did not encounter any difficulties in this part. It was a good starting point that gave me the confidence to tackle the more complex tasks ahead.

**Jupyter Notebook Installation**

Similarly, researching the Jupyter Notebook installation was straightforward because I already had it installed on my system. This familiarity allowed me to focus more on understanding the best practices for using Jupyter Notebook effectively. Exploring the interface and learning how to run Python code within the notebook environment was enjoyable and reinforced my existing knowledge.

**TensorFlow and TensorFlow Lite Installation**

Researching the installation of TensorFlow and TensorFlow Lite was more challenging. While I did not face any significant hurdles, understanding the specific commands and steps needed for their installation required careful mindfulness. Documenting these steps was a meticulous process, and although I grasped the basics, I feel that doing the installation firsthand might help solidify my understanding.

**Neural Network Model Design**

Conceptually designing a neural network model for the MNIST dataset using Keras was a significant part of the assignment. I had to dive deep into neural network concepts and Keras documentation to understand the architecture, including the input shape, number of layers, types of layers, activation functions, and output. Documenting each component of the model architecture was a detailed process, but it was rewarding to see it all come together. However, I need more practical experience to fully understand the nuances of neural network design.

**Data Loading and Preprocessing**

Researching the process of loading and preprocessing the MNIST dataset was an enlightening experience. I learned how to load the dataset using TensorFlow and understood the importance of preprocessing steps, such as normalizing the data. Documenting these steps required a thorough understanding of data preprocessing techniques and their impact on model performance. This part was quite technical, and while I found it fascinating, I believe that firsthand practice will help me grasp the concepts more deeply.

**Model Training Process**

Researching the model training process involved understanding various components such as optimizer, loss function, and metrics. I had to investigate the appropriate settings for these components and document the steps involved in compiling and training the model. The process of understanding how to train the model and interpreting the results was initially daunting, but it became clearer as I progressed through the research. This part was particularly satisfying, as it gave me a sense of how the model improves over time. However, I think that implementing these steps in a practical setting will provide a clearer understanding.

**Model Conversion to TensorFlow Lite**

Researching the conversion of a trained model to TensorFlow Lite format was a new concept for me. I had to understand the purpose of model conversion and the steps involved in using the TensorFlow Lite converter. Documenting these steps requires a thorough understanding of the conversion process and its importance for deploying models on edge devices. This part was eye-opening, as it showed me the practical applications of model conversion in real-world scenarios. However, I feel that firsthand experience with the conversion process will be beneficial.

**Model Deployment**

Researching the deployment process of the model on a simulated edge device was the most challenging part. I had to learn how to write a script to load and run the TensorFlow Lite model, set up the TensorFlow Lite interpreter, and allocate tensors. Understanding how to evaluate the model on sample input and interpreting the results required a good understanding of the entire workflow. This part was quite complex, but it was exciting to see how the model could be deployed and evaluated in a real-world setting. I believe that practical implementation will help me better understand the deployment process.

**Learning Outcomes**

Through this assignment, I gained a comprehensive understanding of the end-to-end process of developing, training, converting, and deploying an AI model, even though it was purely research-based. I learned the importance of each step, from installing the necessary tools to design and training a neural network model. The assignment also highlighted the significance of model conversion and deployment, which are crucial for real-world applications.

**Application of Knowledge**

The knowledge gained from this conceptual assignment can be applied in various real-world scenarios. For instance, understanding how to install and use TensorFlow and TensorFlow Lite can be beneficial for developing AI models for mobile and edge devices. The skills acquired in designing and training neural networks can be applied to a wide range of problems, from image classification to natural language processing.

In the subsequent practical deployment assignment, I will be able to apply the conceptual knowledge to a real-world project. This will involve setting up the development environment, designing and training a model, converting it to TensorFlow Lite, and deploying it on an edge device. The reflective journal has helped me identify areas where I need to improve and reinforce my understanding of the entire process.

**References**

Python Software Foundation. (n.d.). Python Downloads. Retrieved from [https://www.python.org/downloads/](<https://www.python.org/downloads/>)

TensorFlow Authors. (n.d.). TensorFlow Installation. Retrieved from [https://www.tensorflow.org/install](<https://www.tensorflow.org/install>)

TensorFlow Authors. (n.d.). TensorFlow Lite. Retrieved from [https://www.tensorflow.org/lite](<https://www.tensorflow.org/lite>)

Project Jupyter. (n.d.). Installing Jupyter. Retrieved from [https://jupyter.org/install](<https://jupyter.org/install>)

Keras Team. (n.d.). Keras Documentation. Retrieved from [https://keras.io/](<https://keras.io/>)

LeCun, Y., Cortes, C., & Burges, C. J. C. (n.d.). The MNIST Database of Handwritten Digits. Retrieved from [http://yann.lecun.com/exdb/mnist/](<http://yann.lecun.com/exdb/mnist/>)

TensorFlow Authors. (n.d.). TensorFlow Lite Converter. Retrieved from [https://www.tensorflow.org/lite/convert](<https://www.tensorflow.org/lite/convert>)

TensorFlow Authors. (n.d.). Deploying TensorFlow Lite Models. Retrieved from [https://www.tensorflow.org/lite/guide/android](<https://www.tensorflow.org/lite/guide/android>)