ITAI 3377

Group 3: Hoang Dinh, Jeran Kilpack

**LAB REPORT: Converting and Deploying AI Models Using Tensorflow Lite**

**1. INTRODUCTION**

This lab focuses on training a simple neural network model using TensorFlow and then converting it to TensorFlow Lite (TFLite) format for optimized deployment. The steps include:

* Setting up the Python environment
* Training a neural network on the MNIST dataset
* Converting the trained model to TensorFlow Lite format
* Loading and testing the converted model using the TensorFlow Lite Interpreter

This report will explain every step in the following code, ensuring a clear understanding of the implementation process.

**2. SETTING UP THE DEVELOPMENT ENVIRONMENT**

**Step 1: Verify Python and TensorFlow Installation**

Before training the model, it's important to ensure that Python and TensorFlow are installed correctly.

**Code Snippet:**

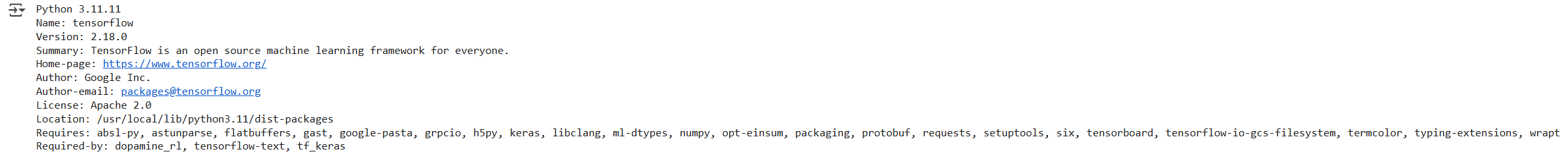
A close up of black text

Description automatically generated

**Expected Output:**

This will display the installed Python version and details about the installed TensorFlow package, including its version and dependencies.

**Actual Output:**



**3. CREATING AND TRAINING AN AI MODEL**

**Step 2: Load the MNIST Dataset and Visualize Sample Images**

The MNIST dataset is a collection of 28x28 pixel grayscale images of handwritten digits (0-9). It is a standard dataset for testing classification models.

**Code Snippet:**

A screen shot of a computer code

Description automatically generated

**Explanation:**

* load\_data() loads the MNIST dataset and splits it into training and testing sets.
* Normalization helps improve model performance by ensuring consistent input scaling.

To better understand the dataset, we display a few sample images from the training set.

**Code Snippet:**

A screenshot of a computer code

Description automatically generated

**Explanation:**

* plt.figure(figsize=(5,5)) initializes a figure of size 5x5 inches.
* plt.subplot(3,3,i+1) arranges the images in a 3x3 grid.
* plt.imshow(x\_train[i], cmap="gray") displays each image in grayscale.
* plt.axis('off') hides axis labels to focus on the images.
* plt.show() renders the visualization.

**Expected Output:**

A 3x3 grid of grayscale images from the MNIST dataset displayed without axis labels.

**Actual Output:**

**A number in a square

Description automatically generated with medium confidence**

**Step 3: Define the Neural Network Model**

A simple feedforward neural network is used for classification.

**Code Snippet:**

A screen shot of a computer code

Description automatically generated

**Explanation:**

* The Flatten layer transforms the 28x28 input into a 1D array.
* A dense (fully connected) layer with 128 neurons and ReLU activation processes the input.
* The output layer has 10 neurons (for 10 classes) using the softmax activation function.
* The model uses the Adam optimizer and sparse categorical cross-entropy loss function.

**Step 4: Train and Save the Model**

The model is trained for five epochs on the MNIST dataset.

**Code Snippet:**

A white background with black text

Description automatically generated

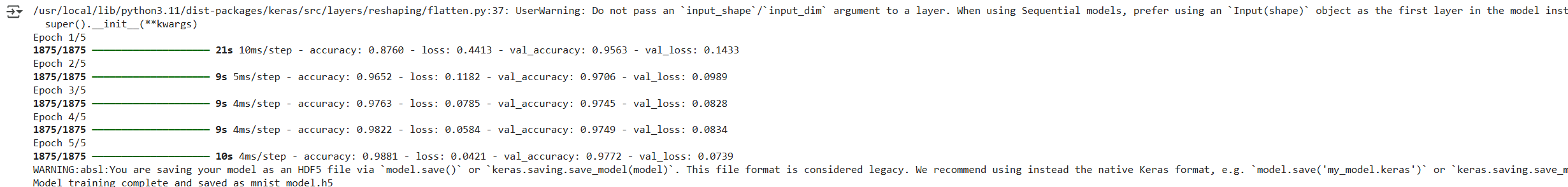
**Explanation:**

* epochs=5: The model iterates over the dataset five times.
* validation\_data=(x\_test, y\_test): Evaluates model performance on the test set.
* model.save("mnist\_model.h5"): Saves the trained model in HDF5 format for future use.

**Expected Output:**

* Training loss and accuracy per epoch.
* Validation loss and accuracy after each epoch.
* Confirmation message: "Model training complete and saved as mnist\_model.h5".

**Actual Output:**

****

**4. CONVERTING THE MODEL TO TENSORFLOW LITE FORMAT**

**Step 5: Convert the Trained Model to TFLite**

To deploy on lightweight environments, we convert the model into the TensorFlow Lite format.

**Code Snippet:**

A screen shot of a computer program

Description automatically generated

**Explanation:**

* TFLiteConverter.from\_keras\_model(model): Converts the Keras model to TFLite format.
* convert(): Performs the conversion process.
* The converted model is saved as mnist\_model.tflite.

**Expected Output:**

* The model successfully converted to TensorFlow Lite format and saved as mnist\_model.tflite.

**Actual Output:**

**A white background with black text

Description automatically generated**

**5. LOADING AND TESTING THE TFLITE MODEL**

**Step 6: Load the TFLite Model and Perform Inference**

To verify the model works correctly in its lightweight form, we use the TensorFlow Lite Interpreter.

**Code Snippet:**

A screen shot of a computer program

Description automatically generated

**Explanation:**

* Interpreter(model\_path): Loads the TFLite model.
* allocate\_tensors(): Prepares the model for inference.
* get\_input\_details() and get\_output\_details(): Retrieve model input/output specifications.

**Expected Output:**

* Display of input and output tensor details for the converted TFLite model.

**Actual Output:**

****

**Step 8: Perform Inference with a Test Image**

To verify the model’s predictions, we select a test image and run inference using the TensorFlow Lite Interpreter.

**Code Snippet:**

**A screenshot of a computer program

Description automatically generated**

**Explanation:**

* test\_image = x\_test[0].astype(np.float32): Converts the test image to a float32 format, matching model input.
* np.expand\_dims(test\_image, axis=0): Adds an extra dimension to match the input shape.
* set\_tensor(input\_details[0]['index'], test\_image): Assigns the test image as input to the model.
* invoke(): Runs inference on the input image.
* get\_tensor(output\_details[0]['index']): Retrieves the model’s output.
* np.argmax(output\_data): Extracts the predicted class from the output data.
* The image is displayed along with the predicted and actual label.

**Expected Output:**

* The test image is displayed along with the predicted and actual label.

**Actual Output:**

**A screen shot of a number

Description automatically generated**

**6. CONCLUSION**

This lab covered:

* Training a neural network using TensorFlow
* Converting it to TensorFlow Lite format
* Loading and testing the converted model

This process is essential for deploying AI models in resource-constrained environments such as mobile devices and embedded systems.