Adebanke Damola-Fashola

ITAI 3377- AI at the Edge and IIOT Environments

Professor Patricia McManus

February 04, 2025.

Reflection on Converting and Deploying AI Models using TensorFlow Lite

Challenges I Faced During the Lab:

While working through this lab, I encountered several challenges that tested my understanding of Keras, TensorFlow Lite, and model deployment. These obstacles provided valuable learning experiences and helped me refine my problem-solving approach.

- **Keras Sequential Model Warning**: Initially, I defined my Sequential model by specifying the input_shape directly in the Flatten layer. This triggered a warning suggesting that input_shape should only be passed in the Input() layer. I resolved this by adding tf.keras.Input(shape=(28, 28)) as the first layer in the model.
- **Deprecation Warning for Model Saving Format**: After saving my model in the .h5 format, Keras displayed a warning that this format is now considered legacy and recommended switching to the .keras format. To follow best practices, I updated my save command to model.save('mnist_model.keras').
- **Optimizer Variable Mismatch When Loading Model**: When reloading my saved model, I encountered a warning stating that the saved optimizer had a different number of variables than expected (e.g., 10 variables vs. 6 variables). To fix this, I saved the model without the optimizer (include_optimizer=False) and then recompiled it after loading.

- **Tensor Dimension Mismatch in TensorFlow Lite (TFLite) Inference**: During TFLite inference, I faced a ValueError: Dimension mismatch when setting the input tensor. The model expected a different number of dimensions than the test image I provided. To resolve this, I used np.expand_dims() to reshape the test image to match the expected input format.
- **Persistent Shape Mismatch in TFLite Input**: Even after reshaping, I encountered another ValueError: Dimension mismatch, indicating that the input shape was still incorrect. To debug the issue, I printed input_details[0][‘shape’] to check the expected dimensions and adjusted my reshaping approach accordingly.
- **Mismatch Between Expected Batch Size and Provided Input**: My model expected an input shape of [32, 28, 28], i.e. it required a batch of 32 images. However, I passed a single image, leading to a mismatch error. To resolve this, I expanded the test image using np.tile() to replicate it 32 times, ensuring it conformed to the expected batch size.

Lessons I Learned:

I gained valuable insights into the best practices for building and deploying Keras models.

- **Defining Keras Models Properly**: I realized the importance of explicitly specifying input shapes to avoid unexpected errors and ensure smooth model training and inference.
- **Keeping Up with Keras Updates**: The transition from .h5 to .keras format highlighted the need to stay informed about evolving frameworks to prevent compatibility issues.
- **Ensuring Optimizer Compatibility**: I learned that recompiling models after modifications is essential to maintaining optimizer functionality and preventing performance issues.

- **Verifying Input Shapes in TFLite**: I encountered challenges related to input dimension mismatches when working with TensorFlow Lite, reinforcing the importance of validating expected input shapes before deployment.
- **Enhancing Deployment Flexibility**: I discovered that allowing dynamic batch sizes during model conversion can simplify real-world applications and make models more adaptable to different use cases.

How TensorFlow Lite applies to real-world AI deployments

TensorFlow Lite (TFLite) is a streamlined version of TensorFlow designed to facilitate the deployment of machine learning models on resource-constrained devices such as smartphones, embedded systems, and IoT devices. Its lightweight nature and optimization capabilities make it ideal for real-world AI applications where low latency, offline processing, and efficient resource utilization are critical.

Real-World Applications of TensorFlow Lite

- **Healthcare – Medical Imaging**: GE Healthcare uses TensorFlow Lite for faster, more accurate MRI scans through on-device processing.
- **Social Media & Content Platforms – Image Classification & Object Detection**: Airbnb improves listing accuracy by using TensorFlow Lite for large-scale image recognition.
- **E-commerce – Personalized Recommendations**: Carousell, an online marketplace, enhances image and text-based search, helping buyers find relevant listings and simplifying the seller experience.
- **Augmented Reality – Virtual Makeup Try-On**: ModiFace enables customers to test makeup shades in real-time via their device cameras.

- **Edge Computing – Real-Time Computer Vision**: TensorFlow Lite supports instant object detection and image recognition on edge devices, reducing latency and enabling offline AI processing.

TensorFlow Lite enhances AI deployment across industries by improving efficiency, reducing reliance on cloud computing, and enabling real-time, on-device machine learning.

References

https://viso.ai/edge-ai/tensorflow-lite/

https://www.analyticsvidhya.com/blog/2024/12/tensorflow-lite-vs-pytorch-mobile/

https://developers.googleblog.com/en/tensorflow-lite-is-now-litert/

https://www.tensorflow.org/about/case-studies