Technologies for Reliable AI Test and Evaluation

Lei Hamilton, Garrett Botkin, Olivia Brown, Justin Goodwin, Michael Yee, Vincent Mancuso, and Sanjeev Mohindra
MIT Lincoln Laboratory
{lei.hamilton, garrett.botkin, olivia.brown, jgoodwin, myee, vincent.mancuso, smohindra}@ll.mit.edu

Abstract

Artificial Intelligence (AI) is revolutionizing many industries, while at the same time facing challenges to safe and reliable use such as vulnerability to adversarial attacks and data drift. Although many AI test and evaluation (T&E) tools exist, integrating them is difficult. Under a program funded by the Chief Digital and AI Office (CDAO), we are developing a library to simplify the AI T&E process by providing user- and developer-friendly interfaces for composing T&E workflows. We illustrate the effectiveness of this approach with an example that compares clean and perturbed accuracy of two models on a computer vision dataset.

Introduction

Artificial Intelligence (AI) is powering advances in important domains such as computer vision, natural language, and autonomy. However, even though deep neural networks—the main drivers of modern AI—often surpass human-level performance, they have been found to be vulnerable to both naturally arising challenges and malicious attacks. They can fail on inputs that differ sufficiently from the training data—for example, if objects are imaged under different lighting conditions or if inputs are corrupted by noise (Hendrycks and Dietterich 2019). They are also susceptible to small input modifications that drastically change system output despite being imperceptible to humans ("adversarial examples") (Szegedy et al. 2013). There is a critical need for comprehensive AI test and evaluation (T&E) tools that can easily and reliably assess AI models along multiple dimensions to support the development, evaluation, deployment, and monitoring of trusted AI systems that are robust to these and other challenges (Chief Digital and Artificial Intelligence Office 2022).

While the current AI ecosystem contains many individual T&E-related tools for tasks such as analyzing datasets, explaining models, augmenting data, generating adversarial attacks, etc., they are not designed to work together, which can make it difficult to compose T&E workflows that span multiple tools. Furthermore, the wide variety of deep learning frameworks and libraries (e.g., PyTorch, TensorFlow, Keras, JAX, Hugging Face, etc.) makes the evaluation of models developed with different tools even harder.

Listing 1: JATIC Core evaluate signature (simplified)

```
import jatic_toolbox.protocols as pr
from typing import Mapping, Any

def evaluate(
  model: pr.ImageClassifier,
  data: pr.VisionDataLoader,
  metric: Mapping[str, pr.Metric],
  augmentation: pr.Augmentation[
      [pr.SupportsImageClassification],
       pr.SupportsImageClassification
      ],
) -> dict[str, Any]:
    ...
```

Approach

To address these challenges, we propose JATIC Core, a library enabling capabilities for comprehensive AI model T&E, developed under the Joint AI Test Infrastructure Capability (JATIC) program funded by the Chief Digital and AI Office (CDAO). JATIC Core provides three key components: (1) Python protocols and types (for model prediction, datasets, data augmentations, and metrics) that are reusable and help specify strongly typed code to promote consistency and interoperability across different ML workflows; (2) interoperability utilities for using popular libraries (such as Hugging Face, Torchvision, and TorchMetrics) with the protocols, as well as high-level evaluation utilities that streamline the configuration and execution of core T&E workflows; and (3) testing utilities that support rigorous testing of the T&E tools themselves.

Python protocols are introduced in PEP 544 (Levkivskyi,

Listing 2: JATIC Core evaluation example

```
vit = load_model(
  provider="huggingface",
  model_name="aaraki/vit-base-patch16-224-
in21k-finetuned-cifar10",
  task="image-classification")

cifar10 = load_dataset(
  provider="torchvision",
  dataset_name="CIFAR10",
  task="image-classification",
  split="test")

accuracy = dict(
  accuracy=load_metric(
   provider="torchmetrics",
   metric_name="Accuracy",
   task="multiclass",
   num_classes=10))

output = evaluate(
  model=vit,
  data=cifar10,
  metric=accuracy)
```

Lehtosalo, and Langa 2017). Unlike classes, there is no need to explicitly inherit from a protocol. Any class that implements all attributes and methods defined in the protocol (with matching signatures) is seen as a subtype of that protocol (a concept known as "structural subtyping"). Use of protocols in JATIC Core helps make code more self-documenting, easier to understand, usable across different tasks, and more reliable by enabling both static and runtime type checking. Developers can also easily define interfaces that are specific to their needs while still being compatible with JATIC Core, allowing integration of new tools and extension to new workflows.

Building on these protocols, JATIC Core provides interoperability utilities for loading models, datasets, and metrics from providers such as Hugging Face and PyTorch. It also provides support for executing core T&E tasks such as measuring performance by simply specifying models, datasets, and metrics that conform to the JATIC protocols. Listing 1 illustrates the call signature of a high-level evaluation workflow for image classification that accepts an image classification model, a vision dataloader to load the dataset, a dictionary that maps metric names to metrics, and an augmentation that will be applied dynamically to inputs during inference. Note that the code has been simplified for purposes of illustration.

Finally, JATIC Core includes a testing utility called "pyright_analyze" that makes it easier to do static type checking on Python objects, docstrings, or files programmatically with Pyright—a full-featured, standards-based static type checker for Python (https://github.com/microsoft/pyright).

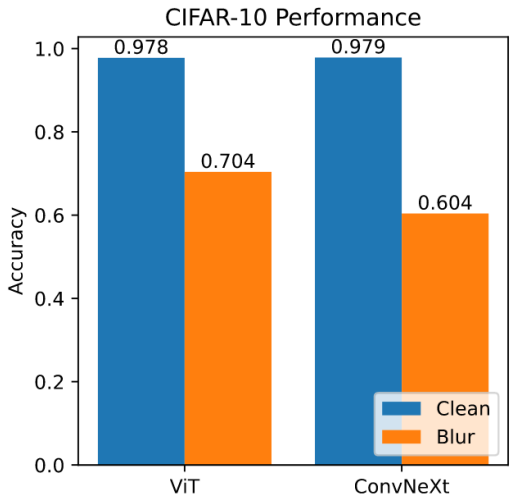

Figure 1: Results from evaluating accuracy of two models on clean and corrupted versions of CIFAR-10.

We show how to use these core capabilities and integrate with common providers with a demonstration that evaluates clean and perturbed accuracy for two image classification models: a ViT vision transformer model (Dosovitskiy et al. 2021) and a ConvNeXt (Liu et al. 2022) model. Listing 2 uses JATIC Core's interoperability utilities for configuring and loading models, datasets, and metrics from various supported providers. The listing ends with a call to evaluate, which will return the accuracy of the specified models on the CIFAR-10 test split. Figure 1 aggregates the results from four evaluate calls that vary the two models under test and the augmentation (none or blur) to provide the user insights into how the models perform under different conditions.

Conclusion

In summary, we present JATIC Core, an enabling library that provides common interfaces, utilities, and tooling to streamline the development of T&E capabilities, support evaluation use cases, and allow composition of different tools into custom workflows. Future directions include exploring the use of generics (Mendoza et al. 2020) to represent tensor shape information (e.g., supporting named batch, channel, height, and width dimensions) and open sourcing the library. Our goal is for JATIC Core to provide improved consistency, reliability, and interoperability across the AI T&E ecosystem.

Acknowledgements

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

This material is based upon work supported by the Under Secretary of Defense for Research and Engineering under Air Force Contract No. FA8702-15-D-0001. Any opinions, findings, conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the Under Secretary of Defense for Research and Engineering.

References

Chief Digital and Artificial Intelligence Office. 2022. JATIC T&E Software Tool Prototyping. https://go.ratio.exchange/opps/challenge.cfm?i=46C49763-B80B-4AF9-80AD-053F2B2095EF. Accessed: 2023-08-21.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; Uszkoreit, J.; and Houlsby, N. 2021. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. In 9th International Conference on Learning Representations, ICLR 2021.

Hendrycks, D.; and Dietterich, T. 2019. Benchmarking neural network robustness to common corruptions and perturbations. arXiv:1903.12261.

Levkivskyi, I.; Lehtosalo, J.; and Langa, Ł. 2017. PEP 544 – Protocols: Structural subtyping (static duck typing). https://peps.python.org/pep-0544/. Accessed: 2023-08-22.

Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T.; and Xie, S. 2022. A ConvNet for the 2020s. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 11966–11976.

Mendoza, M.; Rahtz, M.; Srinivasan, P. K.; and Siles, V. 2020. PEP 646 – Variadic Generics. https://peps.python.org/pep-0646/. Accessed: 2023-08-22.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. arXiv:1312.6199.