AMLA Documentation

Release 0.1

AMLA

Jun 28, 2018

Contents

AMLA is a framework for implementing and deploying AutoML algorithms for Neural Networks.

CHAPTER 1

Introduction

AMLA is a common framework to run different AutoML algorithms for neural networks without changing the underlying systems needed to configure, train and evaluate the generated networks. This has two benefits: * It ensures that different AutoML algorithms can be easily compared using the same set of hyperparameters and infrastructure, allowing for easy evaluation, comparison and ablation studies of AutoML algorithms. * It provides a easy way to deploy AutoML algorithms on multi-cloud infrastructure.

With a framework, we can manage the lifecycle of autoML easily. Without this, hyperparameters and architecture design are spread out, some embedded in the code, others in config files and other as command line parameters, making it hard to compare two algorithms or perform ablation studies.

Some design principles of AMLA: * The network generation process is decoupled from the training/evaluation process. * The network specification model is independent of the implementation of the training/evaluation/generation code and ML library (i.e. whether it uses TensorFlow/PyTorch etc.).

AMLA currently supports the NAC using EnvelopeNets AutoML algorithm, and we are actively adding newer algorithms to the framework. More information on AutoML algorithms for Neural Networks can be found here

CHAPTER 2

Architectural overview

In AMLA, an AutoML algorithm is run as a task and is specified through a configuration file. Sample configuration files may be found here and are described here

When run in single host mode (the default), the system consists of * Command Line Interface (CLI): An interface to add/start/stop tasks. * Scheduler: Starts and stops the AutoML tasks. * Generate/Train/Evaluate: The subtasks that comprise the AutoML task: network generation (via an AutoML algorithm), training and evaluation.

A more detailed description of the current architecture is available here

The current branch is limited to operation on a single host i.e. the CLI, scheduler, generation, training and evaluation all run on a single host. The scheduler may be run as a service or a library, while the generate/train and evaluate subtasks are run as processes. A distributed system that allows concurrent execution of multiple training/evaluation tasks and distributed training on a pod of machines is under development.

CHAPTER 3

Contributing

At this point, AMLA is in its early stages. There are several areas in which development is yet to start or that are under development. If you would like to contribute to AMLA's development, please send in pull requests, feature requests or submit proposals. Here is how to contribute.

Here are some areas that we need help with: * **'New AutoML algorithms <>'__** Add support for new AutoML algorithms such as NAS, ENAS, AmoebaNet * **'Machine learning frameworks <>'__** Add support for more machine learning frameworks such as PyTorch etc. * Standard model format Improve the model specification (add hyper parameters), support for ONNX, other standard model specification formats. * Deployers: Add support for Kubeflow as a deployer * Front end: Contribute to ongoing development using vue.js * **'Test scripts <>'__**: Regression scripts, please! * Documentation: Documentation, please!

Proposals in progress are here ## Installation

3.1 Prerequisites:

Current AMLA supports Tensorflow as the default machine learning library. To install Tensorflow, follow the instructions here: - https://www.tensorflow.org/install/install_linux#InstallingVirtualenv to install TensorFlow in a virtualenv for GPU/CPU. - Alternatively, use an AWS DeepLearning AMI on an AWS GPU instance: http://aws.amazon.com/blogs/machine-learning/get-started-with-deep-learning-using-the-aws-deep-learning-ami/

3.2 Install

```
git clone https://github.com/ciscoai/amla
```

3.3 Run the CLI

```
cd amla/amla
python amla.py
```

3.4 Add/start a task

```
Run an AutoML algorithm (NAC) to generate/train/evaluate

#amla add_task configs/config.nac.construction.json
Added task: {'taskid': 0, 'state': 'init', 'config': 'configs/config.nac.construction.
↪json'} to schedule.
#amla start_task 0

Start a single train/evaluate run using a network defined in the config file

#amla add_task configs/config.run.json
Added task: {'taskid': 1, 'state': 'init', 'config': 'configs/config.run.json'} to␣
↪schedule.
#amla start_task <taskid>

Run the test construction algorithm (few iterations, few training steps)

#amla add_task configs/config.nac.construction.test.json
Added task: {'taskid': 2, 'state': 'init', 'config': 'configs/config.nac.construction.
↪test.json'} to schedule.
#amla start_task <taskid>

Run the test training/evaluation task

#amla add_task configs/config.run.test.json
Added task: {'taskid': 3, 'state': 'init', 'config': 'configs/config.run.test.json'}␣
↪to schedule.
#amla start_task <taskid>
```

3.5 Analyze

```
tensorboard --logdir=amla/results/<arch name>/results/
```

CHAPTER 4

Authors

- Utham Kamath pukamath@cisco.com
- Abhishek Singh abhishs8@cisco.com
- Debo Dutta dedutta@cisco.com

If you use AMLA for your research, please cite this paper

```
@INPROCEEDINGS{kamath18,
  AUTHOR = {P. Kamath and A. Singh and D. Dutta},
  TITLE = {{AMLA: An AutoML frAmework for Neural Network Design}}
  BOOKTITLE = {AutoML Workshop at ICML 2018},
  CITY = {Stockholm},
  MONTH = {July},
  YEAR = {2018},
  PAGES = {},
  URL = {}
}
```