Aim

Release 2.7.4

Gev Sogomonian, Gor Arakelyan et al.

Oct 11, 2021

DOCUMENTATION

Aim is an open-source comparison tool for AI experiments. With more resources and complex models more experiments are ran than ever. Use Aim to deeply inspect thousands of hyperparameter-sensitive training runs at once.

CHAPTER

ONE

GETTING STARTED IN 3 STEPS

Follow the steps below to get started with Aim.

1. Install Aim on your training environment

Prerequisite: You need to have python3 and pip3 installed in your environment before installing Aim

```
$ pip install aim
```

2. Integrate Aim with your code

- Integrate your Python script

```
import aim

# Save inputs, hparams or any other `key: value` pairs
aim.set_params(hyperparam_dict, name='hparams') # Passing name argument is optional

# ...
for step in range(10):
    # Log metrics to visualize performance
    aim.track(metric_value, name='metric_name', epoch=epoch_number)
# ...
```

See documentation *here*.

- Integrate PyTorch Lightning

```
from aim.pytorch_lightning import AimLogger

# ...
trainer = pl.Trainer(logger=AimLogger(experiment='experiment_name'))
# ...
```

See documentation *here*.

- Integrate Hugging Face

```
from aim.hugging_face import AimCallback

# ...
aim_callback = AimCallback(repo='/path/to/logs/dir', experiment='mnli')
trainer = Trainer(
    model=model,
```

(continues on next page)

(continued from previous page)

```
    args=training_args,
    train_dataset=train_dataset if training_args.do_train else None,
    eval_dataset=eval_dataset if training_args.do_eval else None,
    callbacks=[aim_callback],
    # ...
)
# ...
```

See documentation *here*.

- Integrate Keras & tf.keras

```
import aim

# ...
model.fit(x_train, y_train, epochs=epochs, callbacks=[
    aim.keras.AimCallback(repo='/path/to/logs/dir', experiment='experiment_name')

    # Use aim.tensorflow.AimCallback in case of tf.keras
    aim.tensorflow.AimCallback(repo='/path/to/logs/dir', experiment='experiment_name')
])
# ...
```

See documentation *here*.

- Integrate XGBoost

```
from aim.xgboost import AimCallback

# ...
aim_callback = AimCallback(repo='/path/to/logs/dir', experiment='experiment_name')
bst = xgb.train(param, xg_train, num_round, watchlist, callbacks=[aim_callback])
# ...
```

See documentation *here*.

3. Run the training as usual and start Aim UI

```
$ aim up
```

CHAPTER

TWO

OVERVIEW

Aim helps you to compare 1000s of training runs at once through its framework-agnostic python SDK and performant UI.

While using Aim SDK you create a Session object. It handles the tracking of metrics and parameters.

When the training code is instrumented with Aim SDK's *Python Library* and ran, Aim creates the `.aim` repository in your specified path and stores the data. Otherwise the data is created and stored in working directory.

Additionally, Aim SDK also gives you flexibility to:

- use multiple sessions in one training script to store multiple runs at once. When not initialized explicitly, Aim creates a default session.
- use experiments to group related runs together. An experiment named `default` is created otherwise.
- use integrations to automate tracking

CHAPTER

THREE

DEMOCRATIZING AI DEV TOOLS

3.1 The mission...

Aim's mission is to democratize AI dev tools. We believe that the best AI tools need to be:

- open-source, open-data-format, community-driven, extensible
- have great UI/UX, CLI and other interfaces for automation
- performant both on UI and data

3.2 Our motivation...

Existing open-source tools (TensorBoard, MLFlow) are super-inspiring.

However we see lots of improvements to be made. Especially around issues like:

- ability to handle 1000s of large-scale experiments
- actionable, beautiful and performant visualizations
- extensibility - how easy are the apis for extension/democratization?

These problems are a huge motivation.

We are inspired to build beautiful, scalable AI dev tools with great APIs. That's what unites the Aim community.

Join us, help us build the future of AI tooling!

CHAPTER

FOUR

SDK SPECIFICATIONS

4.1 Session

Session is the main object that tracks and stores the ML training metadata (metrics and hyperparams).

Use Session arguments to define:

- custom path for `.aim` directory
- experiment name: each session is associated with an experiment
- run name/message

Use Session methods to specify:

- the metrics of your training run(s)
- the hyperparameters of your training run(s)

Class aim.**Session_**()

Arguments

- **repo** - Full path to parent directory of Aim repo - the `.aim` directory. By default current working directory.
- **experiment** - A name of the experiment. By default `default`. Use experiments to group related runs together.
- **flush_frequency** - The frequency per step to flush intermediate aggregated values of metrics to disk. By default per `128` step.
- **block_termination** - If set to `True` process will wait until all the tasks are completed, otherwise pending tasks will be killed at process exit. By default `True`.
- **run** - A name of the run. If run name is not specified, universally unique identifier will be generated.
- **system_tracking_interval** - System resource usage tracking interval in seconds. By default 10 seconds. In order to disable system tracking set `system_tracking_interval=0`.

Methods

- `track()`- Tracks the training run metrics associated with the Session
- `set_params()` - Sets the params of the training run associated with the Session
- `flush()` - Flushes intermediate aggregated metrics to disk. This method is called at a given frequency and at the end of the run automatically.
- `close()` - Closes the session. If not invoked, the session will be automatically closed when the training is done.

Returns

Session object to attribute recorded training run to.

Example

- Here are a few examples of how to use the `aim.Session` in code.

The Default Session

When no session is explicitely initialized, a default Session object is created by Aim.

When `aim.track` or `aim.set_params` are invoked, underneath the default session object's `track` and `set_param` are called.

4.1.1 track

Session.**track**_(value, name='metric_name' [, epoch=epoch] [, **context_args])

Parameters

- **value** - the metric value of type `int/float` to track/log
- **name** - the name of the metric of type `str` to track/log (preferred divider: `snake_case`)
- **epoch** - an optional value of the epoch being tracked
- **context_args** - any set of other parameters passed would be considered as key-value context for metrics

Example

```
session_inst = aim.Session()

session_inst.track(0.01, name='loss', epoch=43, subset='train', dataset='train_1')
session_inst.track(0.003, name='loss', epoch=43, subset='val', dataset='val_1')
```

Once tracked this way, the following search expressions are enabled:

```
loss if context.subset in (train, val) # Retrieve all losses in both train and val phase
loss if context.subset == train and context.dataset in (train_1) # Retrieve all losses␣
↪in train phase with given datasets
```

Please note that any key-value could be used to track this way and enhance the context of metrics and enable even more detailed search.

Search by context example here:

4.1.2 set_params

Session.**set_params**_(dict_value, name)

Parameters

- **dict_value** - Any dictionary relevant to the training
- **name** - A name for dictionaries

Example

```
session_inst = aim.Session()

 # really any dictionary can go here
hyperparam_dict = {
```

(continues on next page)

(continued from previous page)

```
  'learning_rate': 0.0001,
  'batch_siz': 32,
}
session_inst.set_params(hyperparam_dict, name='params')
```

The following params can be used later to perform the following search experssions:

```
loss if params.learning_rate < 0.01 # All the runs where learning rate is less than 0.01
loss if params.learning_rate == 0.0001 and params.batch_size == 32 # all the runs where␣
↪learning rate is 0.0001 and batch_size is 32
```

Note: If the `set_params` is called several times with the same name all the dictionaries will add up in one place on the UI.

4.1.3 flush

Session.**flush**_()

Aim calculates intermediate values of metrics for aggregation during tracking. This method is called at a given frequency(see *Session*) and at the end of the run automatically. Use this command to flush those values to disk manually.

4.1.4 Instrumentation

Use Python Library to instrument your training code to record the experiments.

The instrumentation only takes 2 lines:

```
# Import aim
import aim

# Initialize a new session
session_inst = Session()
```

Afterwards, simply use the two following functions to track metrics and any params respectively.

```
session_inst.set_params(hyperparam_dict, name='dict_name')

for iter, sample in enumerate(train_loader):
  session_inst.track(metric_val, name='metric_name', epoch=current_epoch)
```

4.2 Integrations

We have integrated Aim to Tensorflow, Keras and Pytorch Lightning to enable automatic tracking. It allows you to track metrics without the need for explicit track statements.

4.2.1 TensorFlow and Keras

Pass an instance of `aim.tensorflow.AimCallback` to the trainer callbacks list.

****Note:**** Logging for pure `keras` is handled by `aim.keras.AimCallback`

Parameters

- **repo** - Full path to parent directory of Aim repo - the `.aim` directory (optional)
- **experiment** - A name of the experiment (optional)

Example

```
import aim

# ...
model.fit(x_train, y_train, epochs=epochs, callbacks=[
    aim.tensorflow.AimCallback(repo='/path/to/logs/dir', experiment='experiment_name')

    # Use aim.keras.AimCallback in case of pure keras
    aim.keras.AimCallback(repo='/path/to/logs/dir', experiment='experiment_name')
])
# ...
```

TensorFlow v1 full example here TensorFlow v2 full example here Keras full example here

4.2.2 PyTorch Lightning

Pass `aim.pytorch_lightning.AimLogger` instance as a logger to the `pl.Trainer` to log metrics and parameters automatically.

Parameters

- **repo** - Full path to parent directory of Aim repo - the `.aim` directory (optional)
- **experiment** - A name of the experiment (optional)
- **train_metric_prefix** - The prefix of metrics names collected in the training loop. By default `train_` (optional)
- **test_metric_prefix** - The prefix of metrics names collected in the test loop. By default `test_` (optional)
- **val_metric_prefix** - The prefix of metrics names collected in the validation loop. By default `val_` (optional)
- **flush_frequency** - The frequency per step to flush intermediate aggregated values of metrics to disk. By default per `128` step. (optional)
- **system_tracking_interval** - System resource usage tracking interval in seconds. By default 10 seconds. In order to disable system tracking set `system_tracking_interval=0`. (optional)

Example

```
from aim.pytorch_lightning import AimLogger

...
# Initialize Aim PL logger instance
aim_logger = AimLogger(experiment='pt_lightning_exp')

# Log parameters (optional)
aim_logger.log_hyperparams({
```

(continues on next page)

(continued from previous page)

```
  "max_epochs": 10,
})

trainer = pl.Trainer(logger=aim_logger)
trainer.fit(model, train_loader, val_loader)
...
```

Full example here

4.2.3 Hugging Face

Pass `aim.hugging_face.AimCallback` instance as a callback to the `transformers.Trainer` to log metrics and parameters automatically.

Parameters

- **repo** - Full path to parent directory of Aim repo - the `.aim` directory (optional)
- **experiment** - A name of the experiment (optional)
- **system_tracking_interval** - System resource usage tracking interval in seconds. By default 10 seconds. In order to disable system tracking set `system_tracking_interval=0`. (optional)

Example

```
from aim.hugging_face import AimCallback

# ...
# Initialize Aim callback instance
aim_callback = AimCallback(repo='/path/to/logs/dir', experiment='mnli')

# Initialize trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset if training_args.do_train else None,
    eval_dataset=eval_dataset if training_args.do_eval else None,
    compute_metrics=compute_metrics,
    tokenizer=tokenizer,
    data_collator=data_collator,
    callbacks=[aim_callback]
)
# ...
```

Full example here

4.2.4 XGBoost

Pass `aim.xgboost.AimCallback` instance as a callback to the `xgboost.train` to log metrics automatically.

Parameters

- **repo** - Full path to parent directory of Aim repo - the `.aim` directory (optional)
- **experiment** - A name of the experiment (optional)
- **system_tracking_interval** - System resource usage tracking interval in seconds. By default 10 seconds. In order to disable system tracking set `system_tracking_interval=0`. (optional)
- **flush_frequency** - The frequency per step to flush intermediate aggregated values of metrics to disk. By default per `128` step. (optional)

Example

```
from aim.xgboost import AimCallback

# ...
# Initialize Aim callback instance
aim_callback = AimCallback(repo='/path/to/logs/dir', experiment='experiment_name')

# Initialize trainer
bst = xgb.train(param, xg_train, num_round, watchlist, callbacks=[
  aim_callback,
])
# ...
```

Full example here

CHAPTER

FIVE

COMMAND LINE INTERFACE

Aim CLI offers a simple interface to easily organize and record your experiments. Paired with the *Python Library*, Aim is a powerful utility to record, search and compare AI experiments. Here are the set of commands supported:

Command	Description
`init`	Initialize the `aim` repository.
`version`	Displays the version of aim cli currently installed.
`experiment`	Creates a new experiment to group similar training runs into.
`up`	Runs Aim web UI for the given repo

****This step is optional.**** Initialize the aim repo to record the experiments.

```
$ aim init
```

Creates `.aim` directory to save the recorded experiments to. Running `aim init` in an existing repository will prompt the user for re-initialization.

*****Beware:***** Re-initialization of the repo clears `.aim` folder from previously saved data and initializes new repo. *****Note:***** This command is not necessary to be able to get started with Aim as aim is automatically initializes with the first aim function call.

Display the Aim version installed.

```
$ aim version
```

Create new experiments to organize the training runs. Here is how it works:

```
$ aim experiment COMMAND [ARGS]
```

Command	Args	Description
`add`	-n	`--name <exp_name>`	Add new experiment with a given name.
`checkout`	-n	`--name <exp_name>`	Switch/checkout to an experiment with given name.
`ls`		List all the experiments of the repo.
`rm`	-n	`--name <exp_name>`	Remove an experiment with the given name.

*****Disclaimer:***** Removing the experiment also removes the recorded experiment runs data.

Start the Aim web UI locally.

```
$ aim up [ARGS]
```

Args	Description
`-h &#124; --host <host>`	Specify host address.
`-p &#124; --port <port>`	Specify port to listen to.
`--repo <repo_path>`	Path to parent directory of `.aim` repo. *Current working directory by default*
`--tf_logs <logs_dir_path>`	Use Aim to search and compare TensorBoard experiments. More details in *TensorBoard Experiments*
`--dev`	Run UI in development mode.

Please make sure to run ``aim up`` in the directory where ``.aim`` is located.

CHAPTER

SIX

USE CASES

6.1 Searching Experiments

AimQL is a super simple, python-like search that enables rich search capabilities to search experiments. Here are the ways you can search on Aim:

- **Search by experiment name** - `experiment == {name}`
- **Search by run** - `run.hash == "{run_hash}"` or `run.hash in ("{run_hash_1}", "{run_hash_2}")` or `run.archived is True`
- **Search by param** - `params.{key} == {value}`
- **Search by context** - `context.{key} == {value}`

6.1.1 Search Examples

- Display the losses and accuracy metrics of experiments whose learning rate is 0.001:
  - `loss, accuracy if params.learning_rate == 0.001`
- Display the train loss of experiments whose learning rate is greater than 0.0001:
  - `loss if context.subset == train and params.learning_rate > 0.0001`

Check out this demo project deployment to play around with search.

6.2 TensorBoard Experiments

Easily run Aim on experiments visualized by TensorBoard. Here is how:

```
$ aim up --tf_logs path/to/logs
```

This command will spin up Aim on the TensorFlow summary logs and load the logs recursively from the given path. Use `tf:` prefix to select and display metrics logged with tf.summary in the dashboard, for example `tf:accuracy`.

Tensorboard search example here

6.3 Anonymized Telemetry

We constantly seek to improve Aim for the communty. Telementry data helps us immensely by capturing anonymous usage analytics and statistics. You will be notified when you run `aim up`. The telemetry is collected only on the UI. The python package **does not** have any telemetry associated with it.

6.3.1 Motivation

Aim UI uses segment's analytics toolkit to collect basic info about the usage:

- Anonymized stripped-down basic usage analytics;
- Anonymized number of experiments and run. We constantly improve the storage and UI for performance in case of many experiments. This type of usage analytics helps us to stay on top of the performance problem. *Note: **no* analytics is installed on the Aim Python package.*

6.3.2 How to opt out

You can turn telemetry off by setting the `AIM_UI_TELEMETRY_ENABLED` environment variable to `0`.

Contributor Guide

CHAPTER

SEVEN

CHANGELOG

7.1 2.7.4 Oct 11 2021

- Resolve UI version bug (gorarakelyan)

7.2 2.7.3 Oct 6 2021

- Add VERSION file to MANIFEST (gorarakelyan)

7.3 2.7.2 Oct 6 2021

- Integrate readthedocs (gorarakelyan)

7.4 2.7.1 Jun 30 2021

- Fix bookmark navigation issue (roubkar)
- Empty metric select on X-axis alignment property change (roubkar)

7.5 2.7.0 Jun 23 2021

- Add ability to export table data as CSV (KaroMourad)
- Add ability to bookmark explore screen state (roubkar)
- Add dashboards and apps API (mihran113)

7.6 2.6.0 Jun 12 2021

- Resolve namedtuple python 3.5 incompatibility (gorarakelyan)
- Add ability to align X-axis by a metric (mihran113, roubkar)
- Add tooltip popover for the chart hover state (roubkar)

7.7 2.5.0 May 27 2021

- Set gunicorn timeouts (mihran113)
- Remove redundant deserialize method (gorarakelyan)
- Move the Flask server to main repo to support 'docker'less UI (mihran113)

7.8 2.4.0 May 13 2021

- Bump up Aim UI to v1.6.0 (gorarakelyan)
- Add xgboost integration (khazhak)
- Update keras adapter interface (khazhak)
- Convert tensors to python numbers (gorarakelyan)

7.9 2.3.0 Apr 10 2021

- Bump up Aim UI to v1.5.0 (gorarakelyan)
- Set default interval of sys tracking to 10 seconds (gorarakelyan)
- Add ability to track system metrics (gorarakelyan)

7.10 2.2.1 Mar 31 2021

- Bump up Aim UI to v1.4.1 (gorarakelyan)

7.11 2.2.0 Mar 24 2021

- Bump up Aim UI to v1.4.0 (gorarakelyan)
- Add Hugging Face integration (Khazhak)
- Reorganize documentation (Tatevv)

7.12 2.1.6 Feb 26 2021

- Add ability to opt out telemetry (gorarakelyan)
- Remove experiment name from config file when calling repo.remove_branch method (gorarakelyan)

7.13 2.1.5 Jan 7 2021

- Handle NaN or infinite floats passed to artifacts (gorarakelyan)

7.14 2.1.4 Dec 2 2020

- Add ability to specify session run hash (gorarakelyan)
- Initialize repo if it was empty when opening session (gorarakelyan)
- Add validation of map artifact parameters (gorarakelyan)

7.15 2.1.3 Nov 24 2020

- Support comparison of list type contexts (gorarakelyan)

7.16 2.1.2 Nov 24 2020

- Fix empty contexts comparison issue (gorarakelyan)

7.17 2.1.1 Nov 22 2020

- Return only selected params in SelectResult (gorarakelyan)

7.18 2.1.0 Nov 19 2020

- Add AimRepo select method (gorarakelyan)
- Implement SelectResult class (gorarakelyan)

7.19 2.0.27 Nov 13 2020

- Fix issue with artifact step initializer (gorarakelyan)

7.20 2.0.26 Nov 10 2020

- Add `block_termination` argument to aim.Session (gorarakelyan)
- Convert infinity parameter to string in artifacts (gorarakelyan)

7.21 2.0.25 Nov 9 2020

- Reconstruct run metadata file when running close command (gorarakelyan)

7.22 2.0.24 Nov 8 2020

- Add SIGTERM signal handler (gorarakelyan)
- Run `track` function in a parallel thread (gorarakelyan)
- Add SDK session flush method (gorarakelyan)
- Flush aggregated metrics at a given frequency (gorarakelyan)
- Update run metadata file only on artifacts update (gorarakelyan)

7.23 2.0.23 Nov 5 2020

- Make experiment name argument required in SDK close command (gorarakelyan)

7.24 2.0.22 Nov 5 2020

- Add SDK `close` method to close dangling experiments (gorarakelyan)

7.25 2.0.21 Nov 1 2020

- Resolve compatibility issues with python 3.5.0 (gorarakelyan)

7.26 2.0.20 Oct 26 2020

- Enable pypi aim package name (gorarakelyan)

7.27 2.0.19 Oct 25 2020

- Add PyTorch Lightning logger (gorarakelyan)
- Add TensorFlow v1 and v2 keras callbacks support (gorarakelyan)

7.28 2.0.18 Oct 7 2020

- Add ability to run Aim UI in detached mode (gorarakelyan)
- Add ability to specify repo path when running Aim UI (gorarakelyan)

7.29 2.0.17 Oct 5 2020

- Rename `AimDE` to `Aim UI` (gorarakelyan)

7.30 2.0.16 Oct 2 2020

- Add ability to specify host when running AimDE (gorarakelyan)
- Disable `AimContainerCommandManager` (gorarakelyan)
- Remove `aimde` command entry point (gorarakelyan)
- Remove `de` prefix from development environment management commands (gorarakelyan)

7.31 2.0.15 Sep 21 2020

- Set Map artifact default namespace (gorarakelyan)

7.32 2.0.14 Sep 21 2020

- Set Metric hashable context to None if no kwarg is passed (gorarakelyan)

7.33 2.0.13 Sep 21 2020

- Add ability to query runs by metric value (gorarakelyan)
- Add ability to query runs via SDK (gorarakelyan)

7.34 2.0.12 Sep 12 2020

- Update Session to handle exceptions gracefully (gorarakelyan)

7.35 2.0.11 Sep 11 2020

- Add alias to keras adapter (gorarakelyan)

7.36 2.0.10 Sep 10 2020

- Show progress bar when pulling AimDE image (gorarakelyan)

7.37 2.0.9 Sep 10 2020

- Add ability to start multiple sessions (gorarakelyan)
- Add Aim adapter for keras (gorarakelyan)

7.38 2.0.8 Aug 26 2020

- Set SDK to select only unarchived runs by default (gorarakelyan)
- Add ability to archive/unarchive runs (gorarakelyan)
- Enable search by run attributes (gorarakelyan)
- Add `is not` keyword to AimQL (gorarakelyan)

7.39 2.0.7 Aug 21 2020

- Validate Artifact values before storing (gorarakelyan)
- Add sessions to SDK (gorarakelyan)

7.40 2.0.6 Aug 13 2020

- Add ability to retrieve metrics and traces from repo (gorarakelyan)
- Add SDK `select` method to select runs and artifacts (gorarakelyan)
- Implement search query language (gorarakelyan)

7.41 2.0.5 Jul 18 2020

- Fix issue with PyPI reStructuredText format compatibility (gorarakelyan)

7.42 2.0.4 Jul 18 2020

- Add ability to attach tf.summary logs to AimDE (gorarakelyan)

7.43 2.0.3 Jul 8 2020

- Pass project path to development environment container (gorarakelyan)

7.44 2.0.2 Jul 7 2020

- Make `epoch` argument optional for `Metric` artifact (gorarakelyan)
- Add ability to automatically commit runs after exit (gorarakelyan)
- Add `aim up` shortcut for running development environment (gorarakelyan)
- Remove first required argument(artifact name) from sdk track function (gorarakelyan)
- Add general dictionary artifact for tracking `key: value` parameters (gorarakelyan)

7.45 2.0.1 Jun 24 2020

- Fix inconsistent DE naming (gorarakelyan)

7.46 2.0.0 Jun 18 2020

- Tidy up aim and remove some artifacts (gorarakelyan)
- Update AimContainerCMD to open connection on custom port (gorarakelyan)
- Save passed process uuid to commit configs (gorarakelyan)
- Ability to query processes (gorarakelyan)
- Execute process and store logs into a commit of specific experiment (gorarakelyan)
- Kill running process and its children recursively (gorarakelyan)

- Keep executed processes for monitoring and management (gorarakelyan)
- Add container command handler to exec commands on the host (gorarakelyan)
- Refactor Text artifact to store sentences using protobuf and aimrecords (jamesj-jiao)
- Add ability to pass aim board port as an argument (gorarakelyan)

7.47 1.2.17 May 8 2020

- Add config command (gorarakelyan)
- Tune artifacts: images, metric_groups, params (gorarakelyan)

7.48 1.2.16 Apr 29 2020

- Add ability to pass numpy array as a segmentation mask (gorarakelyan)

7.49 1.2.15 Apr 29 2020

- Add basic image list tracking (gorarakelyan)

7.50 1.2.14 Apr 27 2020

- Optimize segmentation tracking insight to load faster (gorarakelyan)

7.51 1.2.13 Apr 25 2020

- Remove GitHub security alert (gorarakelyan)
- Add image semantic segmentation tracking (gorarakelyan)

7.52 1.2.12 Apr 20 2020

- Add missing init file for aim.artifacts.proto (@mike1808)

7.53 1.2.11 Apr 16 2020

- Make epoch property optional for Metric (gorarakelyan)

7.54 1.2.10 Apr 16 2020

- Serialize and store `Metric` records using protobuf and aimrecords (gorarakelyan)
- Create RecordWriter factory which handles artifact records saving (gorarakelyan)
- Extract artifact serialization to ArtifactWriter (mike1808)

7.55 1.2.9 Mar 16 2020

- Alert prerequisites installation message for running board (gorarakelyan)

7.56 1.2.8 Mar 15 2020

- Update profiler interface for keras (gorarakelyan)

7.57 1.2.7 Mar 14 2020

- Add board pull command (gorarakelyan)
- Change board ports to 43800,1,2 (gorarakelyan)
- Add ability to profile graph output nodes (gorarakelyan)
- Remove issue with autograd inside while loop (gorarakelyan)
- Add aim board development mode (gorarakelyan)
- Update board name hash algorithm to md5 (gorarakelyan)
- Add board CLI commands: up, down and upgrade (gorarakelyan)
- Add ability to tag version as a release candidate (gorarakelyan)

7.58 1.2.6 Feb 28 2020

- Add learning rate update tracking (gorarakelyan)

7.59 1.2.5 Feb 25 2020

- Add autocommit feature to push command: `aim push -c [-m <msg>]` (gorarakelyan)
- Add cli status command to list branch uncommitted artifacts (gorarakelyan)
- Add an ability to aggregate duplicated nodes within a loop (gorarakelyan)
- Remove gradient break issue when profiling output nodes (gorarakelyan)

7.60 1.2.4 Feb 20 2020

- Enable profiler to track nodes inside loops (gorarakelyan)
- Ability to disable profiler for evaluation or inference (gorarakelyan)

7.61 1.2.3 Feb 13 2020

- Set minimum required python version to 3.5.2 (gorarakelyan)

7.62 1.2.2 Feb 13 2020

- Downgrade required python version (gorarakelyan)

7.63 1.2.1 Feb 13 2020

- Edit README.md to pass reStructuredText validation on pypi (gorarakelyan)

7.64 1.2.0 Feb 13 2020

- Make aim CLI directly accessible from main.py (gorarakelyan)
- Add disk space usage tracking (gorarakelyan)
- Add profiler support for Keras (gorarakelyan)
- Add TensorFlow graph nodes profiler (gorarakelyan)
- Add command to run aim live container mounted on aim repo (gorarakelyan)
- Update profiler to track GPU usage (gorarakelyan)
- Add machine resource usage profiler (gorarakelyan)

7.65 1.1.1 Jan 14 2020

- Remove aim dependencies such as keras, pytorch and etc (gorarakelyan)

7.66 1.1.0 Jan 12 2020

- Update code diff tracking to be optional (gorarakelyan)
- Add default False value to aim init function (gorarakelyan)
- Update aim repo to correctly identify cwd (gorarakelyan)
- Update push command to commit if msg argument is specified (gorarakelyan)
- Add ability to initialize repo from within the sdk (gorarakelyan)

7.67 1.0.2 Jan 7 2020

- Remove objects dir from empty .aim branch index (gorarakelyan)

7.68 1.0.1 Dec 26 2019

- Add cil command to print aim current version (gorarakelyan)

7.69 1.0.0 Dec 25 2019

- Add aim version number in commit config file (gorarakelyan)
- Update push command to send username and check storage availability (gorarakelyan)
- Add hyper parameters tracking (gorarakelyan)
- Update push command to print shorter file names when pushing to remote (gorarakelyan)
- Update tracking artifacts to be saved in log format (gorarakelyan)
- Add pytorch cuda support to existing sdk artefacts (gorarakelyan)
- Add cli reset command (gorarakelyan)
- Add nested module tracking support to aim sdk (gorarakelyan)
- Add code difference tracking to aim sdk (gorarakelyan)
- Update aim push command to send commits (gorarakelyan)
- Add commit structure implementation (gorarakelyan)
- Add aim commit command synchronized with git commits (gorarakelyan)
- Add version control system factory (gorarakelyan)
- Update all insights example (gorarakelyan)
- Add model gradients tracking (gorarakelyan)
- Add model weights distribution tracking (gorarakelyan)
- Add aim correlation tracking (gorarakelyan)

7.70 0.2.9 Nov 30 2019

- Update push tolerance when remote origin is invalid (gorarakelyan)

7.71 0.2.8 Nov 30 2019

- Update aim auth public key search algorithm (gorarakelyan)

7.72 0.2.7 Nov 14 2019

- Update dependencies torch and torchvision versions (sgevorg)

7.73 0.2.6 Nov 5 2019

- Update aim track logger (gorarakelyan)

7.74 0.2.5 Nov 4 2019

- Add branch name validation (gorarakelyan)
- Add single branch push to aim push command (gorarakelyan)

7.75 0.2.4 Nov 3 2019

- Update aim auth print format (gorarakelyan)
- Update setup.py requirements (gorarakelyan)

7.76 0.2.3 Nov 3 2019

- Update package requirements (gorarakelyan)

7.77 0.2.2 Nov 1 2019

- Update package requirements (sgevorg)

7.78 0.2.1 Nov 1 2019

- Add paramiko to required in setup.py (sgevorg)

7.79 0.2.0 Nov 1 2019

- Update the repo to prep for open source pypi push (sgevorg)
- Add error and activity logging (sgevorg)
- Add push command robustness (gorarakelyan)
- Add cli auth command (gorarakelyan)
- Add public key authentication (gorarakelyan)
- Update push to send only branches (gorarakelyan)
- Add branching command line interface (gorarakelyan)
- Update skd interface (gorarakelyan)
- Add pytorch examples inside examples directory (gorarakelyan)
- Add model load sdk method (gorarakelyan)
- Add model checkpoint save tests (gorarakelyan)
- Update file sending protocol (gorarakelyan)
- Add model tracking (gorarakelyan)

7.80 0.1.0 - Sep 23 2019

- Update setup py to build cython extensions (gorarakelyan)
- Update tcp client to send multiple files through one connection (gorarakelyan)
- Update tcp client to send images (gorarakelyan)
- Update sdk track functionality to support multiple metrics (gorarakelyan)
- Update push command for sending repo to a given remote (gorarakelyan)
- Add cli remote commands (gorarakelyan)
- Update cli architecture from single group of commands to multiple groups (gorarakelyan)
- Add testing env first skeleton and versions (sgevorg)
- Add dummy exporting files from .aim-test (sgevorg)
- Add description for Testing Environment (sgevorg)
- Update metadata structure and handling (sgevorg)
- Add support for seq2seq models (sgevorg)
- Update the output of doker image build to be more informative and intuitive (sgevorg)
- Update README.MD with changed Aim messaging (sgevorg)
- Remove setup.cfg file (maybe temporarily) (sgevorg)

- Update the location for docker build template files, move to data/ (sgevorg)
- Update the `docs/cli.md` for aim-deploy docs (sgevorg)
- Add docker deploy `.aim/deploy_temp/<model>` cleanup at the end of the build (sgevorg)
- Add Docker Deploy via `aim-deploy` command (sgevorg)
- Add Docker image generate skeleton (sgevorg)
- Add AimModel.load_mode static function to parse `.aim` files (sgevorg)
- Update exporter to decouple from specifics of exporting and framework (sgevorg)
- Add model export with `.aim` extension (sgevorg)
- Remove pack/unpack of the metadata (sgevorg)
- Add pack/unpack to add metadata to model for engine processing (sgevorg)
- Add aim-deploy command configuration in cli (sgevorg)
- Add basic cli (sgevorg)
- Update setup.py for cli first version (sgevorg)
- Add initial cli specs (sgevorg)
- Add directories: the initial skeleton of the repo (sgevorg)
- Add gitignore, license file and other basics for repo (sgevorg)