Aim

Release 3.0.1

Gev Sogomonian, Gor Arakelyan et al.

Oct 22, 2021

QUICK START

Aim is open-source, self-hosted AI experiment tracking tool. Use Aim to deeply inspect hundreds of hyperparameter-sensitive training runs at once.

CHAPTER

ONE

OVERVIEW

Aim is open-source, self-hosted AI experiment tracking tool. Use Aim to deeply inspect hundreds of hyperparameter-sensitive training runs at once.

CHAPTER

TWO

AIM IN 3 STEPS

You only need these three steps to get started with Aim

1. Install Aim v3.x:

```
pip3 install aim
```

1. Simple integration with your training code:

```
from aim import Run

# Initialize a new run
r = Run()

# Log run parameters
r["foo"] = {
    "bar": 1,
    "baz": 2,
}

# Log metrics
for i in range(1000):
    r.track(i, name="metric_name")
```

1. Start UP the aim UI to observe the run:

```
aim up
```

CHAPTER

THREE

TRACKING YOUR FIRST RUN

1. Create Run stored in current directory

```
from aim import Run

run = Run()
```

1. Log parameters

```
run['hparams'] = {
    'learning_rate': 0.001,
    'batch_size': 32,
}
```

1. Track metrics

```
for epoch in range(num_epochs):
    for step, sample in enumerate(train_loader):
        # ...
        run.track(loss_val, name='loss', step=step, epoch=epoch, context={ "subset":
↪"train" })
        run.track(acc_val, name='acc', step=step, epoch=epoch, context={ "subset": "train
↪" })
        # ...
```

Congratulations! Your first run is ready!

Query tracked metadata

1. Create a repo instance

```
from aim import Repo

# Read .aim repo located at the current working directory
repo = Repo(".")
```

1. Query logged metrics and params

```
query = "metric.name == \"loss\""

# Get collection of metrics
for run_metrics_collection in repo.query_metrics(query).iter_runs():
    for metric in run_metrics_collection:
```

(continues on next page)

(continued from previous page)

```
        # Get run params
        params = metric.run[...]
        # Get metric values
        steps, metric_values = metric.values.sparse_numpy()
```

CHAPTER

FOUR

INTEGRATIONS

Easily integrate Aim with your favorite framework / tool

4.1 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)
# ...
```

4.2 Hugging Face

```
from aim.hugging_face import AimCallback

# ...
aim_callback = AimCallback(repo='/path/to/logs/dir', experiment='mnli')
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,
    callbacks=[aim_callback],
    # ...
)
# ...
```

4.3 Pytorch Lightning

```
from aim.pytorch_lightning import AimLogger

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

4.4 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')
])
# ...
```

4.5 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])
# ...
```

CHAPTER

FIVE

AIM SDK

5.1 aim.sdk.repo module

class aim.sdk.repo.**Repo**(*path*, *, *read_only=None*, *init=False*)
Aim repository object.

Provides methods for repositories creation/opening/cleanup. Provides APIs for accessing Runs. Provides API for querying Runs/Metrics based on a given expression.

Parameters

- **path** (*str*) – Path to Aim repository.
- **read_only** (bool, optional) – Flag for opening Repo in readonly mode. False by default.
- **init** (bool, optional) – Flag used to initialize new Repo. False by default. Recommended to use `aim init` command instead.

collect_metrics_info()
Utility function for getting metric names and contexts for all runs.

Returns Tree of metrics and their contexts.

Return type dict

collect_params_info()
Utility function for getting run meta-parameters.

Returns All runs meta-parameters.

Return type dict

classmethod default_repo(*init=False*)
Named constructor for default repository.

Searches nearest *.aim* directory from current directory to roo directory. If not found, return Repo for current directory.

Parameters **init** (bool, optional) – Flag used to initialize new Repo. False by default. Recommended to use *aim init* command instead.

Returns *Repo* object.

classmethod exists(*path*)
Check Aim repository existence.

Parameters **path** (*str*) – Path to Aim repository.

Returns True if repository exists, False otherwise.

classmethod from_path(*path*, *read_only=None*, *init=False*)
Named constructor for Repo for given path.

Parameters

- **path** (*str*) – Path to Aim repository.
- **read_only** (bool, optional) – Flag for opening Repo in readonly mode. False by default.
- **init** (bool, optional) – Flag used to initialize new Repo. False by default. Recommended to use `aim init` command instead.

Returns *Repo* object.

get_run(*run_hash*)
Get run if exists.

Parameters **run_hash** (*str*) – Run hash.

Returns Run object if hash is found in repository. *None* otherwise.

iter_runs()
Iterate over Repo runs.

Yields next Run in readonly mode .

query_metrics(*query=''*)
Get metrics satisfying query expression.

Parameters **query** (*str*) – query expression.

Returns Iterable for metrics matching query expression.

Return type MetricCollection

query_runs(*query=''*, *paginated=False*, *offset=None*)
Get runs satisfying query expression.

Parameters

- **query** (str, optional) – query expression. If not specified, query results will include all runs.
- **paginated** (bool, optional) – query results pagination flag. False if not specified.
- **offset** (str, optional) – *hash* of Run to skip to.

Returns Iterable for runs/metrics matching query expression.

Return type MetricCollection

classmethod rm(*path*)
Remove Aim repository.

Parameters **path** (*str*) – Path to Aim repository.

5.2 aim.sdk.run module

class aim.sdk.run.**Run**(*run_hash=None*, *, *repo=None*, *read_only=False*, *experiment=None*, *system_tracking_interval=10*)

Run object used for tracking metrics.

Provides method *track* to track value series [metrics] for multiple metrics and contexts. Provides dictionary-like interface for Run object meta-parameters. Provides API for iterating tracked metrics.

Parameters

- **run_hash** (str, optional) – Run's hash. If skipped, generated automatically.
- **(** (*repo*) – obj:`Union[Repo,str], optional): Aim repository path or Repo object to which Run object is bound. If skipped, default Repo is used.
- **read_only** (bool, optional) – Run creation mode. Default is False, meaning Run object can be used to track metrics.
- **experiment** (str, optional) – Sets Run's *experiment* property. 'default' if not specified. Can be used later to query runs/metrics.
- **system_tracking_interval** (int, optional) – Sets the tracking interval in seconds for system usage metrics (CPU, Memory, etc.). Set to *None* to disable system metrics tracking.

__delitem__(*key*)

Remove key from run meta-params. :param key: meta-parameter path

__getitem__(*key*)

Get run meta-parameter by key.

Parameters **key** – path to Run meta-parameter.

Returns Collected sub-tree of Run meta-parameters.

Examples

```
>>> run = Run('3df703c')
>>> run['hparams']  # -> {'batch_size': 42}
>>> run['hparams', 'batch_size']  # -> 42
```

__setitem__(*key*, *val*)

Set Run top-level meta-parameter.

Parameters

- **key** (str) – Top-level meta-parameter name. Use ellipsis to reset run's all meta-parameters.
- **val** – Meta-parameter value.

Examples

```
>>> run = Run('3df703c')
>>> run[...] = params
>>> run['hparams'] = {'batch_size': 42}
```

add_tag(*value*)
Add tag to run

Parameters **value** (*str*) – Tag to add.

collect_metrics_info()
Retrieve Run's all metrics general overview.

Returns list of metric's *context*, *metric_name* and last tracked value triplets.

Return type list

get_metric(*metric_name*, *context*)
Retrieve metric sequence by it's name and context.

Parameters

- **metric_name** (*str*) – Tracked metric name.
- **context** (Context) – Tracking context.

Returns Metric object if exists, *None* otherwise.

iter_metrics_info()
Iterator for all run metrics info.

Yields tuples of (metric_name, context, run) where run is the Run object itself.

metrics()
Get iterable object for all run tracked metrics.

Returns Iterable for run metrics.

Return type MetricCollection

Examples

```
>>> run = Run('3df703c')
>>> for metric in run.metrics():
>>>     metric.values.sparse_numpy()
```

remove_tag(*tag_id*)
Remove run tag.

Parameters **tag_id** (*str*) – uuid of tag to be removed.

track(*value*, *name*, *step=None*, *epoch=None*, *, *context=None*)
Main method for tracking numeric value series.

Parameters

- **value** – The tracked value.
- **name** (*str*) – Tracked metric name.
- **step** (int, optional) – Metric tracking iteration. Auto-incremented if not specified.

- **epoch** (int, optional) – The training epoch.
- **context** (dict, optional) – Metric racking context.

Appends the tracked value to metric series specified by *name* and *context*.

property archived
Check is run archived or not.

Getter Returns run's archived state.

Setter Archive/un-archive run.

Type bool

property created_at
Run object creation time [UTC] as datetime.

Getter Returns run creation time.

property creation_time
Run object creation time [UTC] as timestamp.

Getter Returns run creation time.

property description
Run description, set by user.

Getter Returns run's description.

Setter Sets run's description.

Type string

property end_time
Run finalization time [UTC] as timestamp.

Getter Returns run finalization time.

property experiment
Run experiment.

Getter Returns run's experiment name.

Setter Sets run's experiment.

Type string

property finalized_at
Run finalization time [UTC] as datetime.

Getter Returns run finalization time.

property name
Run name, set by user.

Getter Returns run's name.

Setter Sets run's name.

Type string

property tags
List of run tags.

Getter Returns run's tag list.

5.3 aim.sdk.metric module

class aim.sdk.metric.**Metric**(*name*, *context*, *run*)

Class representing series of numeric values.

dataframe(*include_name=False*, *include_context=False*, *include_run=False*, *only_last=False*)

Get metric series as pandas DataFrame

Parameters

- **include_name** – (int, optional): If true, include metric name in dataframe. False by default.
- **include_context** – (int, optional): If true, include metric context path:value pairs in dataframe. False by default.
- **include_run** – (int, optional): If true, include run run.hash and run hparams path:value pairs in dataframe. False by default.
- **only_last** – (int, optional): If true return dataframe for only last value, step, timestamp and epoch. False by default.

CHAPTER

SIX

AIM QL

6.1 Query Language

Aim QL is a super simple, python-like search that enables rich search capabilities to search experiments.

6.1.1 Searching runs

`run` instance

Fields	Description
`name`	Run name
`hasname`	Run hash
`experiment`	Run experiment name
`tags`	List of run tags
`archived`	`True` if run is archived, otherwise `False`
`creation_time`	Run creation timestamp
`end_time`	Run end timestamp

Examples:

- `run.name == "my-favorite-run"`
- `run.learning_rate > 0.0001 and "best-model" in run.tags.names`

6.1.2 Searching metrics

`metric` instance

Fields	Description
`name`	Metric name
`context`	Metric context dictionary

Examples:

- `metric.name == "loss"`
- `metric.name == "loss" and metric.context.subset == "train"`
- `metric.name == "loss" and metric.context.subset == "train" and run.name == "my-favorite-run"`

- `run.hparams.dataset == "few_cb" and not metric.context.is_training and run.hparams.num_epochs < 100`

CHAPTER

SEVEN

AIM CLI

7.1 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.
`up`	Runs Aim web UI for the given repo.
`upgrade`	Upgrades legacy Aim repository from `2.x` to `3.0`.
`reindex`	Process runs left in 'in progress' state.

7.1.1 init

****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.

7.1.2 version

Display the Aim version installed.

```
$ aim version
```

7.1.3 up

Start the Aim web UI locally.

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

Args	Description
-h	--host <host>	Specify host address.
-p	--port <port>	Specify port to listen to.
--repo <repo_path>	Path to parent directory of .aim repo. *Current working directory by default*
--dev	Run UI in development mode.

7.1.4 upgrade

Upgrade Aim repository containing data logged with older version of Aim.

```
$ aim upgrade [ARGS] SUBCOMMAND
```

Args	Description
--repo <repo_path>	Path to parent directory of .aim repo. *Current working directory by default*

upgrade subcommands

Upgrade `aim` repository from `2.x` to `3.0`.

```
$ aim ugrade 2to3 [ARGS]
```

Args	Description
--skip-failed-runs	Use this flag to skip runs which are failed/have missing or incomplete data.
--skip-checks	Use this flag to skip new repository consistency checks.
--drop-existing	Use this flag to clear old .aim directory. By default old data is kept in .aim_legacy.

7.1.5 reindex

Update index to include all runs in Aim repo which are left in progress.

```
$ aim reindex [ARGS]
```

Args	Description
--repo <repo_path>	Path to parent directory of .aim repo. *Current working directory by default*

CHAPTER

EIGHT

AIM STORAGE

8.1 aim.storage.arrayview module

class aim.storage.arrayview.**ArrayView**(**args*, ***kwds*)

Array of homogeneous elements with sparse indices. Interface for working with array as a non-sparse array is available for cases when index values are not important.

first()

First index and value of the array.

first_idx()

First index of the array.

first_value()

First value of the array.

indices()

Return sparse indices iterator.

Yields Array's next sparse index.

indices_list()

Get sparse indices as a `list`.

indices_numpy()

Get sparse indices as numpy array.

items()

Return items iterator.

Yields Tuple of array's next sparse index and value.

keys()

Return sparse indices iterator.

Yields Array's next sparse index.

last()

Last index and value of the array.

last_idx()

Last index of the array.

last_value()

Last value of the array.

sparse_list()

Get sparse indices and values as :obj:`list`s.

`sparse_numpy()`
Get sparse indices and values as numpy arrays.

`tolist()`
Convert to values list

`values()`
Return values iterator.

Yields Array's next value.

`values_list()`
Get values as a `list`.

`values_numpy()`
Get values as numpy array.

CHAPTER

NINE

INTEGRATION GUIDES

9.1 Integration with Huggingface

In this guide, we will show you how to integrate Aim with Huggingface. The work we are going to do together is sentiment classification problem, which is the most common text classification task. We choose the IMDB movie review dataset as an experimental dataset, which classifies movie reviews as positive or negative. During the training process, we will show the use of aim to record effective information.

You only need 2 simple steps to employ Aim to collect data

Step 1: Import the sdk designed by Aim for Huggingface.

```
from aim.hugging_face import AimCallback
```

Step 2: Huggingface has a trainer api to help us simplify the training process. This api provides a callback function to return the information that the user needs. Therefore, aim has specially designed SDK to simplify the process of user writing callback functions, we only need to initialize AimCallback object as follows:

```
# Initialize aim_callback
aim_callback = AimCallback(experiment='huggingface_experiment')
# Initialize trainer
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=small_train_dataset,
    eval_dataset=small_eval_dataset,
    compute_metrics=compute_metrics,
    callbacks=[aim_callback]
)
```

9.2 Integration with Keras & tf.Keras

This tutorial leverages the well-known handwritten digit recognition task to describe how to integrate Aim with Keras & tf.Keras to train a digital image classification model based on the mnist dataset.

It only takes 2 steps to easily integrate aim in keras to record experimental information.

```
# call keras as the high api of tensorflow
from aim.tensorflow import AimCallback
# call keras library directly
from aim.keras import AimCallback
```

In keras, we call the fit method of the model object to train the data. The callbacks are provided here. AimCallback inherits the usage specification of callbacks. We just need to add it to the callbacks list.

```
model.fit(x_train, y_train, epochs=5, callbacks=[
          # in case of tf.keras, we use aim.tensorflow.AimCallback
          AimCallback(experiment='aim_on_keras')
])
```

9.3 Integration with Pytorch Lightning

The work is designed to build a image classifier to solve a famous real world problem ——handwritten digit recognition. In this work, we will introduce how to introduce aim logger to manage output information.

We only require 2 steps to simply and easily inject Aim into pytorch lightining:

```
# call aim sdk designed for pl
from aim.pytorch_lightning import AimLogger
```

Pytorch lighting provides trainer objects to simplify the training process of pytorch model. One of the parameters is called logger. We can use the logger function defined by aim to simplify the process of tracking experiments. This process is divided into 2 steps:

Step 1.create AimLogger object

```
# track experimential data by using Aim
aim_logger = AimLogger(
    experiment='aim_on_pt_lightning',
    train_metric_prefix='train_',
    val_metric_prefix='val_',
)
```

Step 2. Pass the aim_logger object to the logger variable

```
# track experimential data by using Aim
trainer = Trainer(gpus=1, progress_bar_refresh_rate=20, max_epochs=5, logger=aim_logger)
```

9.4 Integration with XGboost

In the real world, there is a well-known handwritten digit recognition problem. In this article, we use the machine learning framework xgboost to help us train an image classification model. In this process, we will use Aim to track our experimental data.

Enjoy using aim to track xgboost experimental data only requires two simple steps:

Step 1: Explicitly import the AimCallback for tracking training data.

```
# call sdk aim.xgboost
from aim.xgboost import AimCallback
```

Step 2: XGboost provides the xgboost.train method for model training, in which the callbacks parameter can call back data information from the outside. Here we pass in aimcallbacl designed for tracking data information

```
xgboost.train(param, dtrain, num_round, watchlist,
                            callbacks=[AimCallback(experiment='xgboost_test')])
```

During the training process, you can start another terminal, in the same directory, start aim up, you can observe the information in real time.

CHAPTER

TEN

CHANGELOG

10.1 3.0.1 Oct 22 2021

- Check telemetry_enabled option on segment initialization (VkoHov)
- Draw LineChart Y-axis (horizontal) tick lines on zooming (KaroMourad)
- Sort select options/params based on input value (roubkar)
- Fix query construction issue for multiple context items (roubkar)
- Fix issue with making API call from Web Worker (VkoHov)

10.2 3.0.0 Oct 21 2021

- TODO [GA]: Highlight v3 core improvements

10.3 2.7.1 Jun 30 2021

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

10.4 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)

10.5 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)

10.6 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)

10.7 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)

10.8 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)

10.9 2.2.1 Mar 31 2021

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

10.10 2.2.0 Mar 24 2021

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

10.11 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)

10.12 2.1.5 Jan 7 2021

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

10.13 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)

10.14 2.1.3 Nov 24 2020

- Support comparison of list type contexts (gorarakelyan)

10.15 2.1.2 Nov 24 2020

- Fix empty contexts comparison issue (gorarakelyan)

10.16 2.1.1 Nov 22 2020

- Return only selected params in SelectResult (gorarakelyan)

10.17 2.1.0 Nov 19 2020

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

10.18 2.0.27 Nov 13 2020

- Fix issue with artifact step initializer (gorarakelyan)

10.19 2.0.26 Nov 10 2020

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

10.20 2.0.25 Nov 9 2020

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

10.21 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)

10.22 2.0.23 Nov 5 2020

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

10.23 2.0.22 Nov 5 2020

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

10.24 2.0.21 Nov 1 2020

- Resolve compatibility issues with python 3.5.0 (gorarakelyan)

10.25 2.0.20 Oct 26 2020

- Enable pypi aim package name (gorarakelyan)

10.26 2.0.19 Oct 25 2020

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

10.27 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)

10.28 2.0.17 Oct 5 2020

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

10.29 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)

10.30 2.0.15 Sep 21 2020

- Set Map artifact default namespace (gorarakelyan)

10.31 2.0.14 Sep 21 2020

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

10.32 2.0.13 Sep 21 2020

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

10.33 2.0.12 Sep 12 2020

- Update Session to handle exceptions gracefully (gorarakelyan)

10.34 2.0.11 Sep 11 2020

- Add alias to keras adapter (gorarakelyan)

10.35 2.0.10 Sep 10 2020

- Show progress bar when pulling AimDE image (gorarakelyan)

10.36 2.0.9 Sep 10 2020

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

10.37 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)

10.38 2.0.7 Aug 21 2020

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

10.39 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)

10.40 2.0.5 Jul 18 2020

- Fix issue with PyPI reStructuredText format compatibility (gorarakelyan)

10.41 2.0.4 Jul 18 2020

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

10.42 2.0.3 Jul 8 2020

- Pass project path to development environment container (gorarakelyan)

10.43 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)

10.44 2.0.1 Jun 24 2020

- Fix inconsistent DE naming (gorarakelyan)

10.45 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)

10.46 1.2.17 May 8 2020

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

10.47 1.2.16 Apr 29 2020

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

10.48 1.2.15 Apr 29 2020

- Add basic image list tracking (gorarakelyan)

10.49 1.2.14 Apr 27 2020

- Optimize segmentation tracking insight to load faster (gorarakelyan)

10.50 1.2.13 Apr 25 2020

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

10.51 1.2.12 Apr 20 2020

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

10.52 1.2.11 Apr 16 2020

- Make epoch property optional for Metric (gorarakelyan)

10.53 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)

10.54 1.2.9 Mar 16 2020

- Alert prerequisites installation message for running board (gorarakelyan)

10.55 1.2.8 Mar 15 2020

- Update profiler interface for keras (gorarakelyan)

10.56 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)

10.57 1.2.6 Feb 28 2020

- Add learning rate update tracking (gorarakelyan)

10.58 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)

10.59 1.2.4 Feb 20 2020

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

10.60 1.2.3 Feb 13 2020

- Set minimum required python version to 3.5.2 (gorarakelyan)

10.61 1.2.2 Feb 13 2020

- Downgrade required python version (gorarakelyan)

10.62 1.2.1 Feb 13 2020

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

10.63 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)

10.64 1.1.1 Jan 14 2020

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

10.65 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)

10.66 1.0.2 Jan 7 2020

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

10.67 1.0.1 Dec 26 2019

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

10.68 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)

10.69 0.2.9 Nov 30 2019

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

10.70 0.2.8 Nov 30 2019

- Update aim auth public key search algorithm (gorarakelyan)

10.71 0.2.7 Nov 14 2019

- Update dependencies torch and torchvision versions (sgevorg)

10.72 0.2.6 Nov 5 2019

- Update aim track logger (gorarakelyan)

10.73 0.2.5 Nov 4 2019

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

10.74 0.2.4 Nov 3 2019

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

10.75 0.2.3 Nov 3 2019

- Update package requirements (gorarakelyan)

10.76 0.2.2 Nov 1 2019

- Update package requirements (sgevorg)

10.77 0.2.1 Nov 1 2019

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

10.78 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)

10.79 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)

CHAPTER

ELEVEN

INDICES AND TABLES

- genindex
- modindex

PYTHON MODULE INDEX

a

INDEX

Symbols

A

C

D

E

F

G

I

K

L

M

N

Q

R

S

T

V