[Distribution Statement A] Approved for public release and unlimited distribution.

Copyright 2022 Carnegie Mellon University.

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

The view, opinions, and/or findings contained in this material are those of the author(s) and should not be construed as an official Government position, policy, or decision, unless designated by other documentation.
References herein to any specific commercial product, process, or service by trade name, trade mark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by Carnegie Mellon University or its Software Engineering Institute.

NO WARRANTY. THIS CARNEGIE MELLON UNIVERSITY AND SOFTWARE ENGINEERING INSTITUTE MATERIAL IS FURNISHED ON AN "AS-IS" BASIS. CARNEGIE MELLON UNIVERSITY MAKES NO WARRANTIES OF ANY KIND, EITHER EXPRESSED OR IMPLIED, AS TO ANY MATTER INCLUDING, BUT NOT LIMITED TO, WARRANTY OF FITNESS FOR PURPOSE OR MERCHANTABILITY, EXCLUSIVITY, OR RESULTS OBTAINED FROM USE OF THE MATERIAL. CARNEGIE MELLON UNIVERSITY DOES NOT MAKE ANY WARRANTY OF ANY KIND WITH RESPECT TO FREEDOM FROM PATENT, TRADEMARK, OR COPYRIGHT INFRINGEMENT.
[DISTRIBUTION STATEMENT A] This material has been approved for public release and unlimited distribution. Please see Copyright notice for non-US Government use and distribution.

This material may be reproduced in its entirety, without modification, and freely distributed in written or electronic form without requesting formal permission. Permission is required for any other use. Requests for permission should be directed to the Software Engineering Institute at permission@sei.cmu.edu.

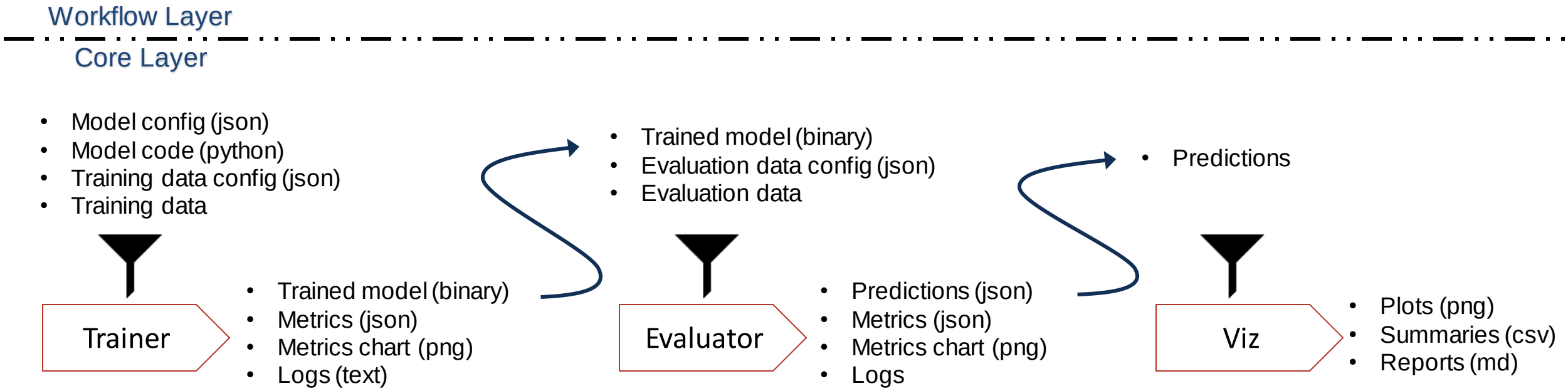
DM22-0889

What is it?

Juneberry aims to improve machine learning experimentation by providing a framework for automating the training, evaluation, and comparison of multiple models against multiple datasets, thereby reducing errors and improving reproducibility.

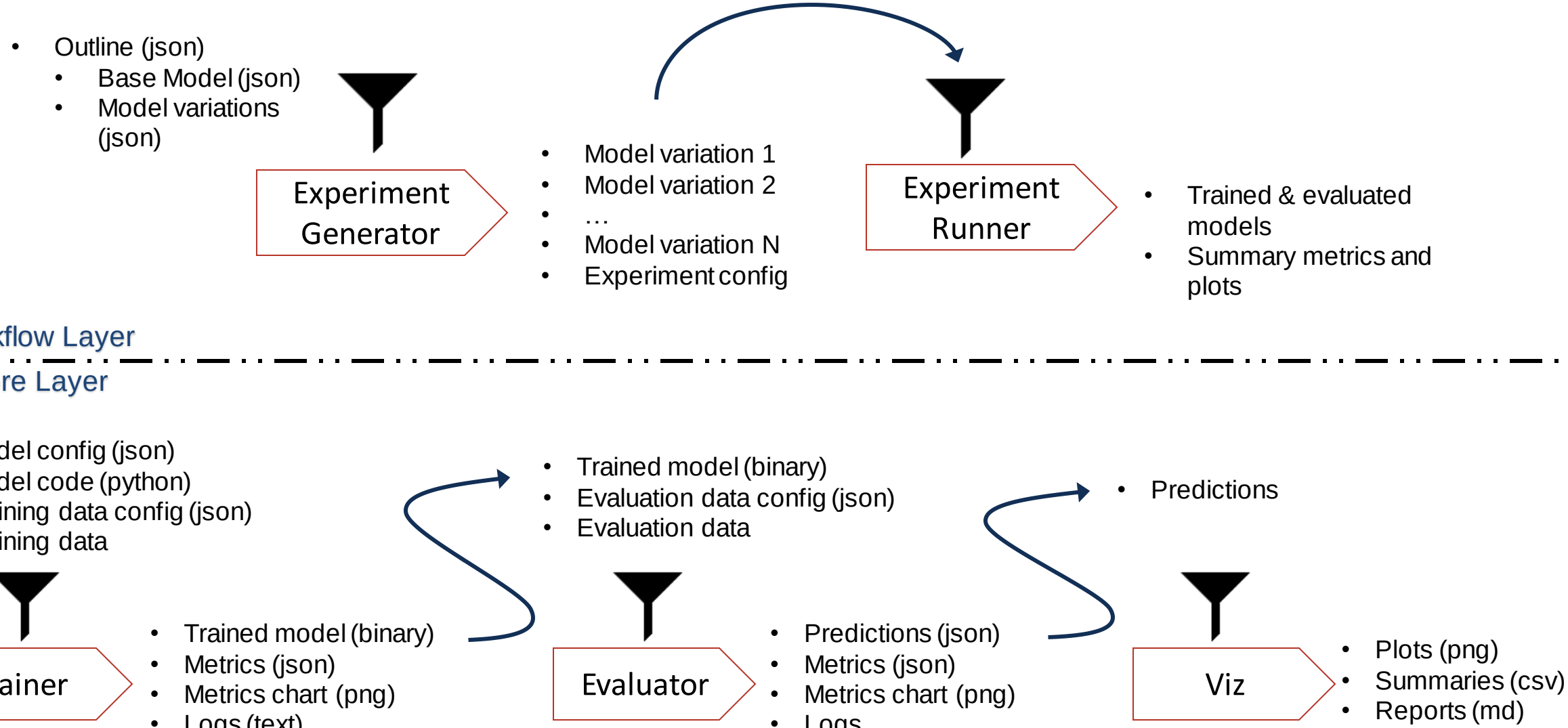
[Distribution Statement A] Approved for public release and unlimited distribution.

Conceptual Layers



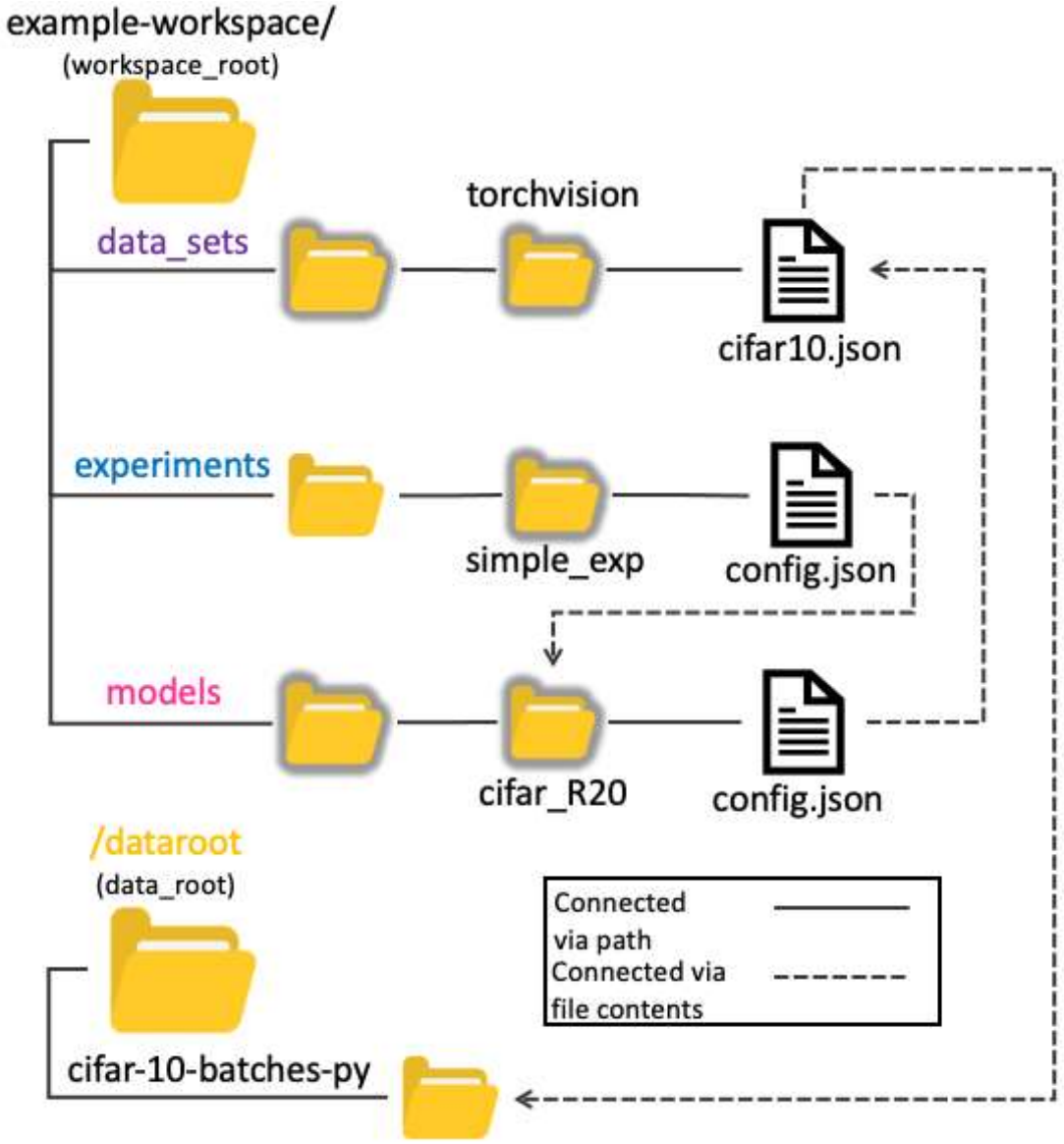
[Distribution Statement A] Approved for public release and unlimited distribution.

Conceptual Layers



[Distribution Statement A] Approved for public release and unlimited distribution.

Workspace Layout



Juneberry provides a well-defined **information architecture** to support the generation, execution, and management of many ML models, along with their inputs and outputs. These consistent locations support the rapid development of tools to quickly adapt to new tasks.

The data_sets directory contains configs describing input data.

Experiment directories contain a config describing models in the experiment, any adjustments to those models, which tests to run, and which reports to produce for comparing results.

Model directories contain a config file describing a machine learning model, and reference which data to use during model training.

The dataroot contains directories organizing input data files.

[Distribution Statement A] Approved for public release and unlimited distribution.

Dataset Configuration

Dataset configurations describe data and its labels. A dataset config can be used for training or evaluation.

Each source entry (which can use globs) specifies a directory of data and its label.

Juneberry can sample the data randomly based on size or fraction.

Maps label integers to strings for logging, reports, and figures.

```
"data_type": "image",
"description": "Imagenette. 160x160, rgb, small training set",
"image_data": {
    "url": "https://github.com/fastai/imagenette",
    "task_type": "classification",
    "sources": [
        { "directory": "imagenette2-160/train/n01440764", "label": 0 },
        { "directory": "imagenette2-160/train/n02102040", "label": 217 },
        { "directory": "imagenette2-160/train/n02979186", "label": 482 },
        { "directory": "imagenette2-160/train/n03000684", "label": 491 },
        { "directory": "imagenette2-160/train/n03028079", "label": 497 }
    ]
},
"label_names": {
    "0": "tench",
    "217": "English_springer",
    "482": "cassette_player",
    "491": "chain_saw",
    "497": "church"
},
"num_model_classes": 1000,
"sampling": {
    "algorithm": "random_quantity",
    "arguments": {
        "count": 10,
        "seed": 98754321 }
}
```

Juneberry repository documents:

- docs/specs/dataset_configuration_specification.md
- juneberry/schemas/dataset_schema.json

[Distribution Statement A] Approved for public release and unlimited distribution.

Model Configuration

Model configurations describe how to construct and train a model. Properties include model architecture, the training dataset, the validation split, evaluation criteria, and training hyperparameters.

The model is defined via code.

PyTorch specific options.

An extensible trainer class.

A sequence of transformations to apply during training.

Instructions for how to generation the validation split.

```
"batch_size": 16,
"epochs": 10,
"evaluator": { "fqcn": "juneberry.pytorch.evaluation.evaluator.Evaluator" },
"model_architecture": {
    "fqcn": "jbexws.architectures.pytorch.pytv_resnet18_pretrained_reference.ReferenceResnet18",
    "kwargs": { "channels": 3, "img_height": 160, "img_width": 160, "num_classes": 1000 }
},
"pytorch": {
    "accuracy_fn": "sklearn.metrics.accuracy_score",
    "accuracy_args": { "normalize": true },
    "deterministic": true,
    "loss_fn": "torch.nn.CrossEntropyLoss",
    "lr_schedule_fn": "MultiStepLR",
    "lr_schedule_args": { "gamma": 0.5, "milestones": [ 3, 5 ] },
    "optimizer_fn": "torch.optim.SGD",
    "optimizer_args": { "lr": 0.01 }
},
"seed": 4210592948,
"trainer": { "fqcn": "juneberry.pytorch.classifier_trainer.ClassifierTrainer" },
"training_dataset_config_path": "data_sets/imagenette_unit_train.json",
"training_transforms": [
    {
        "fqcn": "torchvision.transforms.Normalize",
        "kwargs": { "mean": [ 0.485, 0.456, 0.406 ], "std": [ 0.229, 0.224, 0.225 ] }
    }
],
"validation": {
    "algorithm": "random_fraction",
    "arguments": { "fraction": 0.2, "seed": 3554237221 }
}
```

Common hyperparameters.

Construction follows a common plugin architecture involving a fully qualified class name ("fqcn") and arguments ("kwargs")

[Distribution Statement A] Approved for public release and unlimited distribution.

Report Configuration

Report configurations describe how to synthesize raw training or evaluation data to produce figures, tables, markdown files, and more.

A list of reports to generate

Generates ROC Curves

Generates Precision-Recall curves

Generates a table summarizing metrics data and includes figures

Custom report plugins are possible

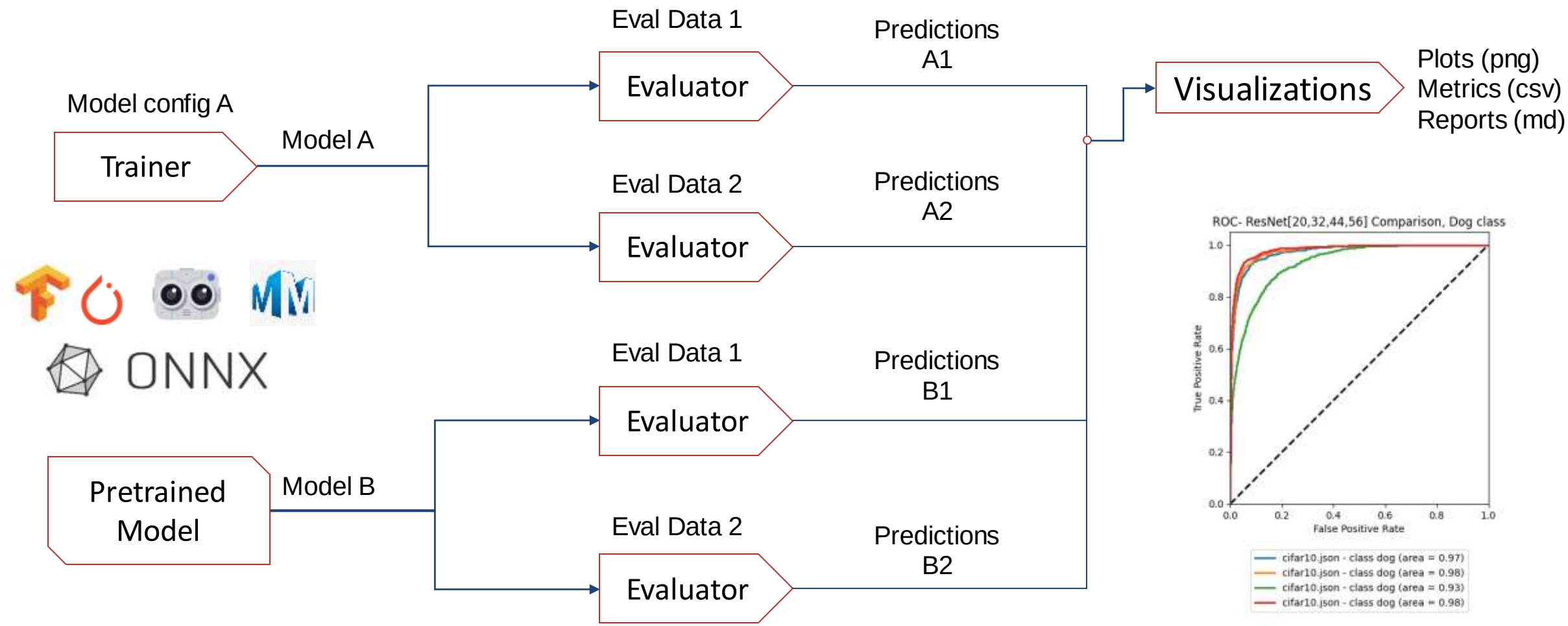
```
"reports": [
    {
        "fqcn": "juneberry.reporting.roc.ROCPlot",
        "kwargs": {
            "output_filename": "experiments/smokeTests/classify/System Test 1 ROC class 0,217,482,491,497.png",
            "plot_title": "Test PyTorch vs TF - First 5 Classes", "legend_font_size": 6,
            "curve_sources": {
                "models/pytorch_model/eval/eval_dataset/predictions.json": "0,English_springer,482,491,church",
                "models/tensorflow_model/eval/eval_dataset/predictions.json": "0,English_springer,482,491,church"
            }
        }
    },
    {
        "fqcn": "juneberry.reporting.pr.PRCurve",
        "kwargs": { "iou": 0.5, "output_dir": "experiments/od/cpu/ut_val_dt2",
            "curve_sources": [
                { "model": "text_detect/dt2/ut", "dataset": "data_sets/text_detect_val.json" }
            ]
        }
    },
    {
        "fqcn": "juneberry.reporting.summary.Summary",
        "kwargs": { "csv_filename": "experiments/smokeTests/od/cpu/Summary.csv",
                    "md_filename": "experiments/smokeTests/od/cpu/Summary.md",
                    "metrics_files": [
                        "models/pytorch_model/eval/eval_dataset/metrics.json",
                        "models/tensorflow_model/eval/eval_dataset/metrics.json",
                        "models/text_detect/dt2/ut/eval/text_detect_val/metrics.json"
                    ],
                    "plot_files": [
                        "experiments/smokeTests/classify/System Test 1 ROC class 0,217,482,491,497.png",
                        "experiments/od/cpu/ut_val_dt2/pr_curve.png",
                        "experiments/od/cpu/ut_val_dt2/pc_curve.png",
                        "experiments/od/cpu/ut_val_dt2/rc_curve.png"
                    ]}
    },
    {
        "fqcn": "juneberry.reporting.custom.CustomReport",
        "kwargs": { "tbd": "tbd" }
    }
]
```

The common plugin architecture of a fully qualified class name (‘fqcn”) combined with arguments (“kwargs”) still applies.

[Distribution Statement A] Approved for public release and unlimited distribution.

Experiments – An Overview

Experiments allow users to organize models together and study their behavior.



The models in this experiment

Use two datasets to evaluate each model

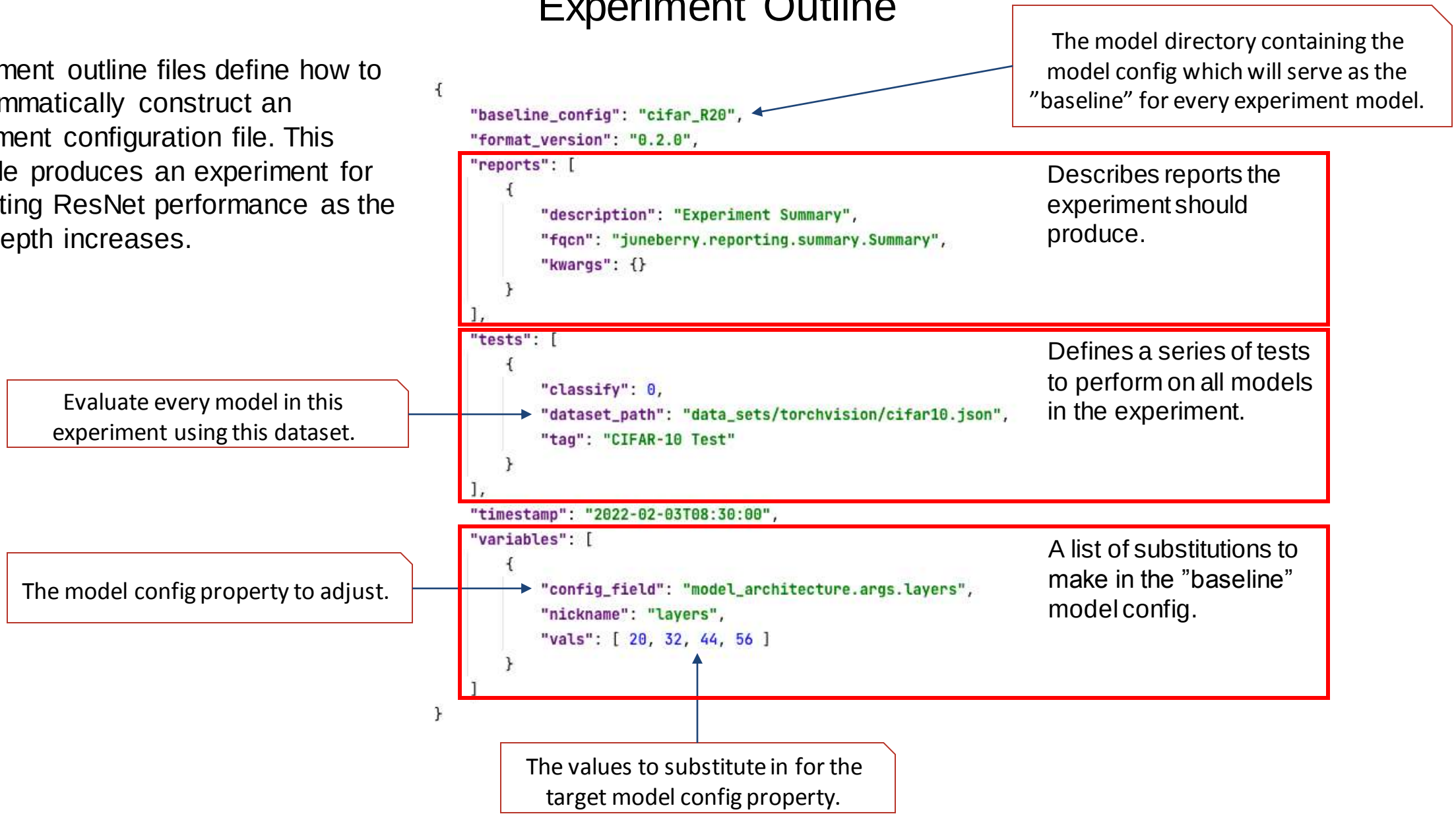
Evaluations produce raw "predictions" data

Use raw data to construct figures and compare performance.

[Distribution Statement A] Approved for public release and unlimited distribution.

Experiment Outline

Experiment outline files define how to programmatically construct an experiment configuration file. This example produces an experiment for evaluating ResNet performance as the layer depth increases.



[Distribution Statement A] Approved for public release and unlimited distribution.

Experiment Config

Experiment config files describe the models involved in an experiment, along with the actions to take. Models can be trained and/or evaluated, and reports can be generated for analysis of the experiment results.

The model directory of a model that will be studied during this experiment.

The experiment will train the model.

Three more models appear because the experiment outline called for four different substitutions in the model config.

Describes the models involved in this experiment.

The trained model will be evaluated using this dataset.

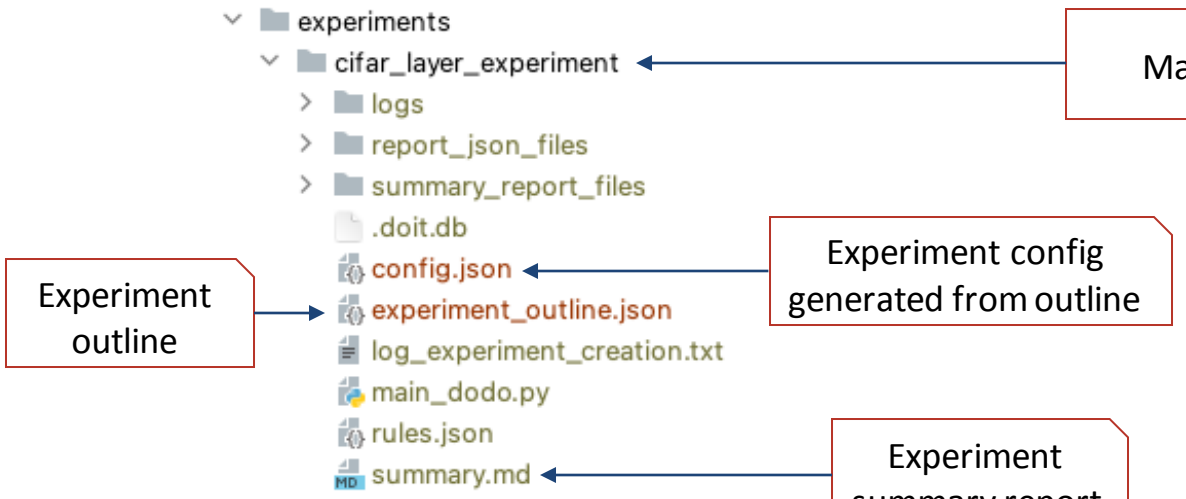
Compressed because the contents are similar to the first model.

Describes reports produced during the experiment

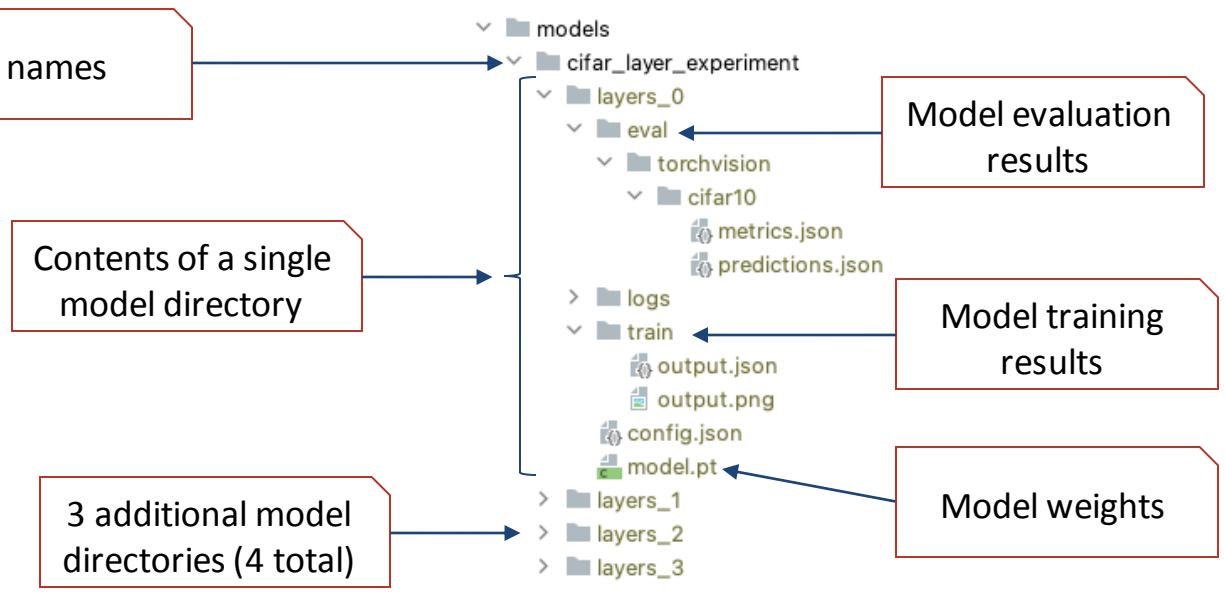
[Distribution Statement A] Approved for public release and unlimited distribution.

Experiment Outputs

Experiment directory after execution



Model directories after execution



Compare

Experiment summary

Model	Duration (seconds)	Eval Dataset	Accuracy	error (%)	Train Chart
cifar_layer_experiment/layers_0	13832.0	data_sets/torchvision/cifar10.json	90.96%	9.04	Training Chart
cifar_layer_experiment/layers_1	20748.0	data_sets/torchvision/cifar10.json	91.33%	8.67	Training Chart
cifar_layer_experiment/layers_2	27573.0	data_sets/torchvision/cifar10.json	92.13%	7.87	Training Chart
cifar_layer_experiment/layers_3	34489.0	data_sets/torchvision/cifar10.json	92.47%	7.53	Training Chart

Table from experiment summary report

method			error (%)
ResNet	20	0.27M	8.75
ResNet	32	0.46M	7.51
ResNet	44	0.66M	7.17
ResNet	56	0.85M	6.97

Table 6 from He et al. (2015)

[Distribution Statement A] Approved for public release and unlimited distribution.

Watermark Evaluation

This experiment tests how much influence a watermark (an inserted image) may have on the target dataset. For a set of watermarks, the tool **`jb_generate_watermark_eval`** produces a series of datasets containing watermarks at different scales and an experiment to evaluate the influence of each watermark.

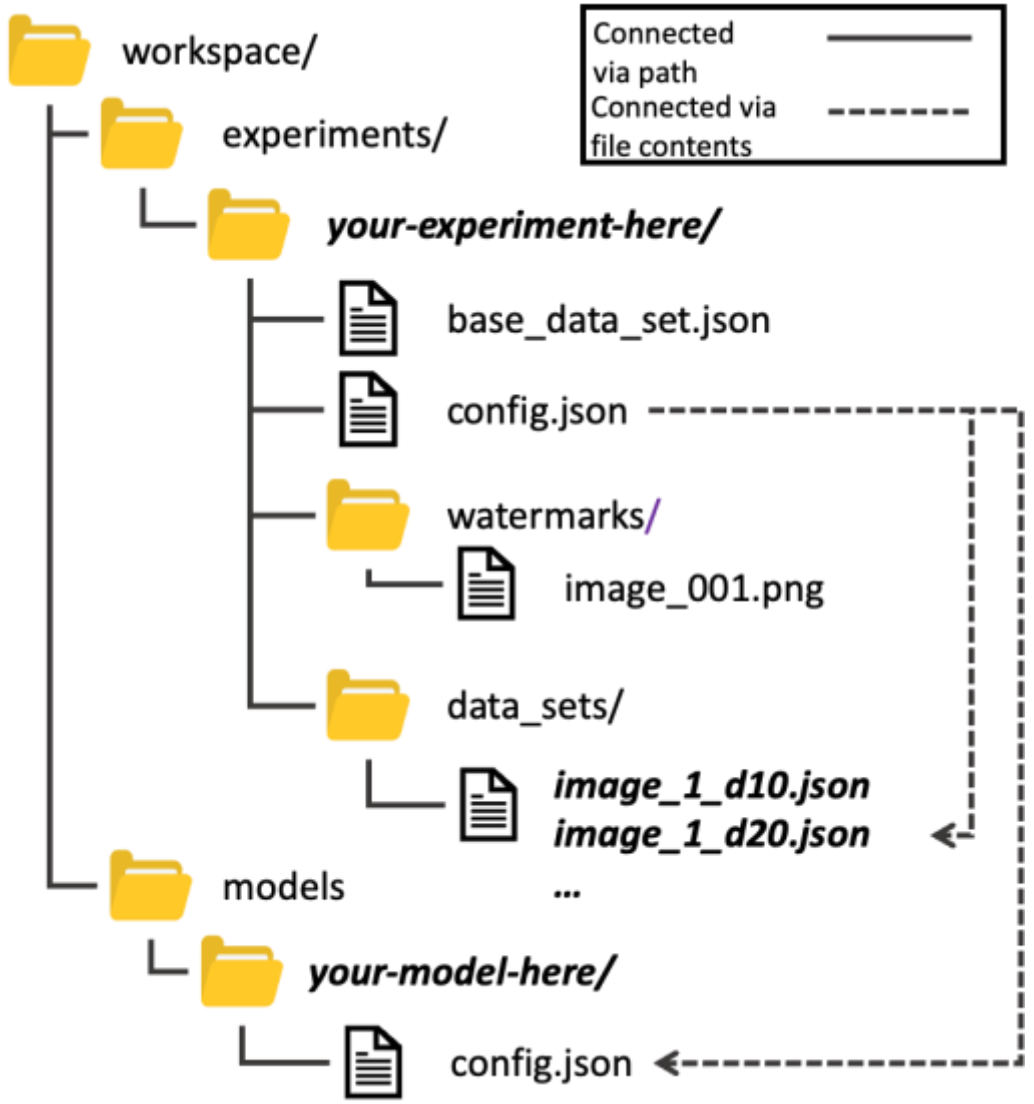
```
jb_generate_watermark_eval \
  your-experiment-here \
  your-model-here \
  0
```

Inputs:

- experiment containing **`base_data_set.json`**
- watermark directory containing N images

Outputs:

- 10 evaluation datasets with watermark scales varying from 10% to 100% in 10% increments
- An experiment **`config.json`** linking the target model and all the evaluation data sets



[Distribution Statement A] Approved for public release and unlimited distribution.

System Attributes

Juneberry is an open-source Python tool that aims to improve machine learning experimentation by reducing errors and improving reproducibility. The Juneberry framework automates tasks related to the training, evaluation, and comparison of multiple machine learning models against multiple datasets.

Key features:

- **Configuration-driven**: Users spend more time designing platform independent experiments and less time writing, testing, and debugging code.
- **Supports multiple ML backends**: Juneberry provides a level playing field for comparing different types of models. Juneberry supports TensorFlow, PyTorch, Detectron2, MMDetection, and ONNX models.
- **Extensible**: Juneberry supports the integration of custom trainers, evaluators, reports, and other plugins, which improves the framework's ability to incorporate the latest innovations in ML research.
- **Emphasizes reproducibility**: Juneberry's structured approach to training and testing simplifies the replication of experiments and the reproduction of results.
- **Evaluates and compares multiple models at a time**: The size of an experiment is constrained only by available computing resources.
- **Reduces the potential for error**: Automation handles the complexity and details when working with large quantities of models and their outputs.
- **Exposes execution tasks**: These allow Juneberry to easily integrate with existing pipelines and workflow tools, such as doit.
- **Supports prototyping**: Juneberry is available when prototyping in Jupyter notebook environments.

[Distribution Statement A] Approved for public release and unlimited distribution.

Reproducibility, Stability, and Reliability

Reproducibility, **stability,** and **reliability** are constant engineering challenges in machine learning. Juneberry promotes these qualities through a **declarative** programming model that is supported by configuration management tools, schema versioning tools, and version control.

Juneberry uses Docker containers to establish an environment for machine learning experimentation that is accessible, consistent, and stable. The Juneberry team maintains publicly available container images on Docker Hub at https://hub.docker.com/r/cmusei/juneberry. These container images include:

- NVIDIA CUDA support - https://docs.nvidia.com/deeplearning/frameworks/
- PyTorch
- TensorFlow2
- Detectron2
- MMDetection
- ONNX
- scikit-learn
- brambox
- Common packages such as: pandas, numpy, jupyter, opencv, etc.

And they all work together in one environment!

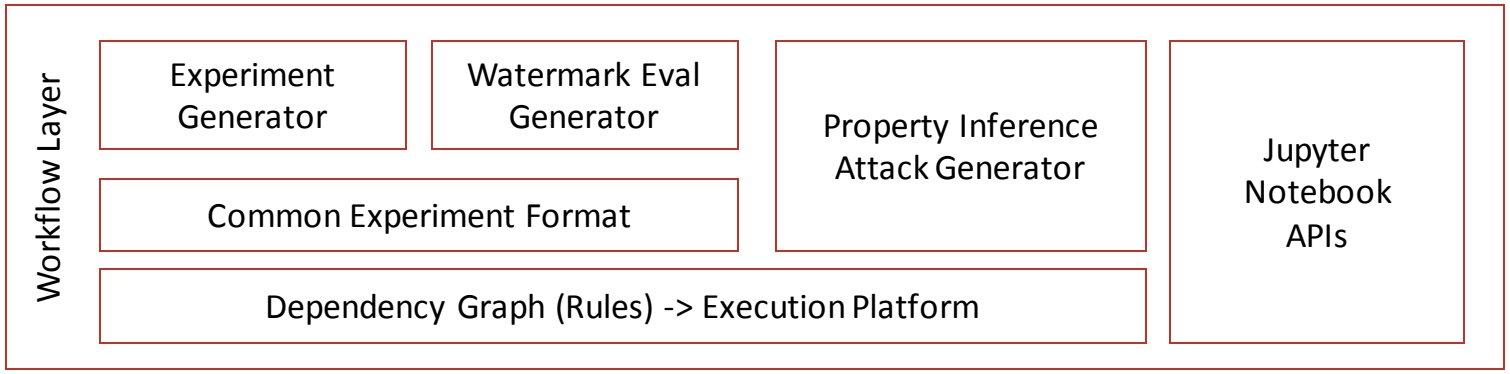
Sample docker file: https://github.com/cmu-sei/juneberry/blob/main/docker/cudadev.Dockerfile

[Distribution Statement A] Approved for public release and unlimited distribution.

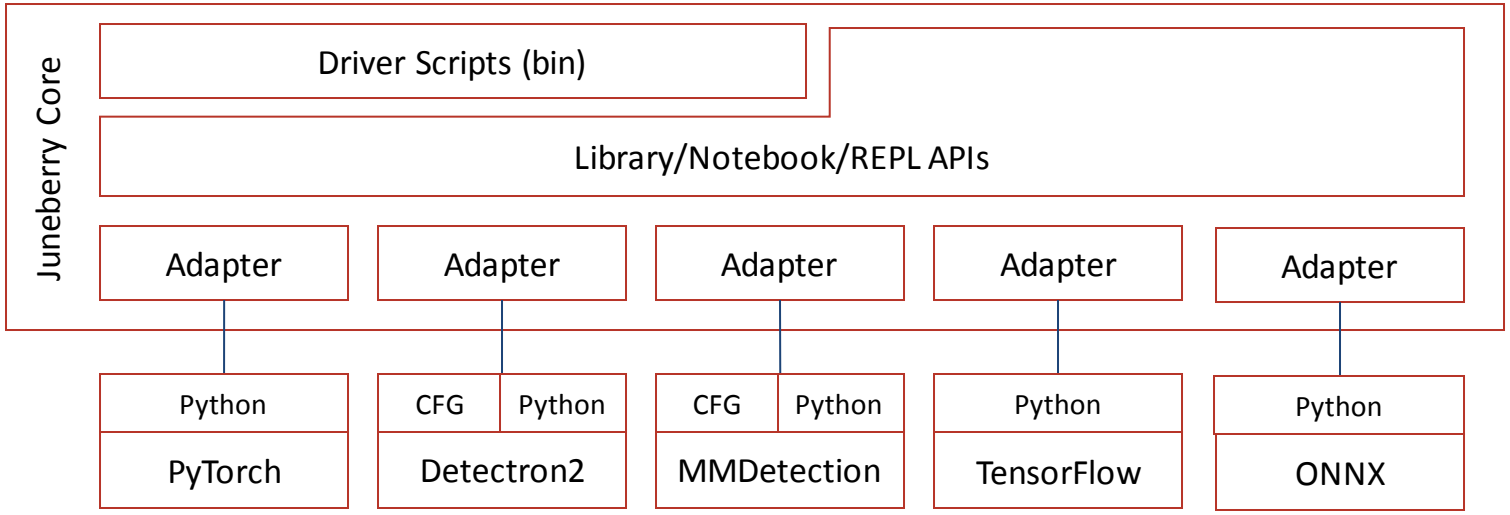
Layer Architecture Details

Juneberry functionality is split into two major layers:

The **workflow layer** contains utilities and schemas which organize tools from the core layer into workflows that produce and execute experiments involving multiple models and processing steps.



The **core layer** consists of the tools required for training, evaluation, and report generation across multiple platforms.



[Distribution Statement A] Approved for public release and unlimited distribution.

Lessons Learned

Research vs Engineering – ML engineering is just getting started and there are too many *leaky abstractions* between engineering and research. Researchers can get distracted addressing issues that should be engineering concerns, leading to user-level one-off solutions and generating technical debt.

Declarative vs Imperative – Using a declarative style creates an abstraction layer and de-couples the ML specifics from the engineering, which increases portability, modifiability, and maintainability.

Reproducibility and Determinism – Most ML platforms use randomizers and distributed computing. Even when properly engineered, these features make it hard to achieve reproducibility, which provides a significant challenge for engineering reliable and predictable tools.

Observability and Understandability – Deep learning models continue to struggle with understandability and observability, which also leads to difficulties in observing when the tooling might be misbehaving.

Missing common API – While some efforts aim to unify the AI experience, most platforms have different approaches to how they train and evaluate data. This inhibits consistency and increases development time when working with multiple platforms.

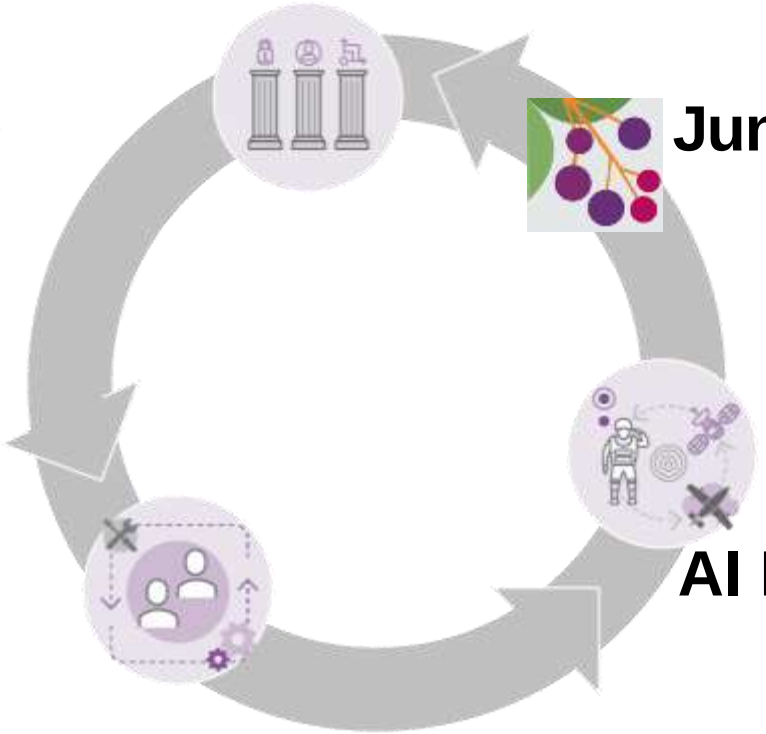
[Distribution Statement A] Approved for public release and unlimited distribution.

Resources

- **Home Page** https://www.sei.cmu.edu/go/juneberry
- **Juneberry** https://github.com/cmu-sei/juneberry
- **Example Workspace** https://github.com/cmu-sei/juneberry-example-workspace
- **Docker Containers** https://hub.docker.com/r/cmusei/juneberry
- **Tutorial** https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=875902

[Distribution Statement A] Approved for public release and unlimited distribution.

Questions and Feedback?



CONTACT

Andrew Mellinger

aomellinger@sei.cmu.edu

GITHUB

github.com/cmu-sei/Juneberry

COME WORK WITH US

sei.cmu.edu/careers/

[Distribution Statement A] Approved for public release and unlimited distribution.