Wandb: Weights and Biases

Rwiddhi Chakraborty
UiT Machine Learning Group,
7th November, 2022

Preliminaries

- Deep learning experiments can be hard to track
- Loss functions, metrics, hyperparameters, models - Combinatorially large
- Enter Wandb (Weights and Biases)
  - Experiment tracking and logging
  - Custom charts, data structures, export options
  - Sweeps
  - Convenient for both group based and individual projects

Experiment tracking and logging - Init

(1)

(2)

(3)

```python
def main(arg_list=None):
    args = parse_args(arg_list)
    logging.basicConfig(level=args.log_level, format=config.LOG_FORMAT)
    logger.info(f"Running evaluation with config:\n{str(args)}")
    wandb_logger.init(args)

    if args.use_cached:
        # No need to load backbone model when we're using cached features.
        model = None
    else:
        model = get_model(arch=args.arch, checkpoint_file=args.checkpoint, dataset_name=args.dataset)

    # Compute additional tensors required for the evaluation.
    extra_tensors = get_extra_tensors(args, model)

    # Get test features
    if args.use_cached:
        # No need to create a test loader when we're using cached features
        test_loader = None
    else:
        test_dataset = get_dataset(args.dataset, split=args.eval_split)
        test_loader = DataLoader(
            test_dataset, batch_size=args.batch_size, num_workers=args.n_workers, shuffle=False, drop_last=False
        )
    features, labels = get_features(model, test_loader, args.cache_dir, args.use_cached, args.eval_split)

    start = time.time()
    metrics = evaluate(features, labels, args, extra_tensors)

    episode_time = (time.time() - start) / args.n_episodes
    logger.info(f"Evaluation finished. {round(episode_time, 4)} seconds/episode")
    wandb.summary["episode_time"] = episode_time

    metrics = helpers.dict_cat(metrics)
    log_metrics(metrics)
```

```
config = config/templates/mini_resnet18.yml
dataset = mini
norm = support_vmfshot
classifier = simpleshot
arch = resnet18
checkpoint = tim/mini/softmax/resnet18/model_best.pth.tar
batch_size = 128
n_workers = 4
use_cached = True
cache_dir = tim/mini/softmax/resnet18
n_episodes = 500
episode_batch_size = 2000
n_shots = 1
n_ways = 5
n_queries = 15
eval_split = test
log_level = INFO
support_vmfshot.
  init = pca
  out_dims = 32
  initial_dims = None
  kappa = 2.0
  perplexity = 30.0
  re_norm = True
  eps = 1e-12
  p_sim = vmf
  p_rel_tol = 0.01
  p_abs_tol = None
  p_betas = (None, None)
  pca_mode = base
  n_iter = 50
  learning_rate = 0.1
  loss_vmf = 0.5
```

Experiment tracking and logging - wandblogger (I)

```python
import os
import wandb
import logging
import helpers

logger = logging.getLogger(__name__)


def get_experiment_id():
    try:
        eid = os.environ["EXPERIMENT_ID"]
    except KeyError:
        eid = wandb.util.generate_id()
        os.environ["EXPERIMENT_ID"] = eid
        logger.warning(f"Could not find EXPERIMENT_ID in environment variables. Using generated id '{eid}'.")
    return eid


class _WandBLogger:
    def __init__(self):
        # Below values will be set when `init` is called.
        self.name = None
        self.args = None
        self.run = None

        # Constants from environment variables
        self.eid = None
        self.entity = os.environ["WANDB_ENTITY"]
        self.project = os.environ["WANDB_PROJECT"]
```

Experiment tracking and logging - wandblogger (II)

```python
        self.project = os.environ["WANDB_PROJECT"]

        self._accumulated_logs = {}

    def init(self, args, job_type="evaluate", tags=None):
        self.eid = get_experiment_id()
        self.name = f"{args.dataset}-{args.arch}-{args.norm}-{args.classifier}-{self.eid}"

        if args.wandb_tags is not None:
            tags = (tags or []) + self._parse_tags(args.wandb_tags)

        cfg = args.to_dict()
        cfg.update(**helpers.versions())
        del cfg["wandb_tags"]

        init_kwargs = dict(
            name=self.name,
            job_type=job_type,
            config=cfg,
            entity=self.entity,
            project=self.project,
            tags=tags,
            reinit=True,
        )

        try:
            self.run = wandb.init(**init_kwargs)
        except wandb.errors.UsageError as err:
            logger.warning(f"Got error: '{str(err)}' when calling wandb.init. Attempting to init with "
```

```python
        try:
            self.run = wandb.init(**init_kwargs)
        except wandb.errors.UsageError as err:
            logger.warning(f"Got error: '{str(err)}' when calling wandb.init. Attempting to init with "
                        f"'settings=wandb.Settings(start_method=''fork'')'")
            self.run = wandb.init(settings=wandb.Settings(start_method="fork"), **init_kwargs)

        return self.run

    @staticmethod
    def _parse_tags(tag_str):
        # Assumes comma-delimited tags
        tags = [tag.strip() for tag in tag_str.split(",")]
        return tags

    def accumulate(self, dct, global_step, local_step, max_local_steps):
        total_step = (global_step * max_local_steps) + local_step
        if total_step in self._accumulated_logs:
            self._accumulated_logs[total_step].update(dct)
        else:
            self._accumulated_logs[total_step] = dct

    def log_accumulated(self):
        for step, logs in sorted(self._accumulated_logs.items(), key=lambda item: item[0]):
            self.run.log(logs, step=step)


wandb_logger = _WandBLogger()
```

Experiment tracking and logging - Eval

```python
def main(arg_list=None):
    args = parse_args(arg_list)
    logging.basicConfig(level=args.log_level, format=config.LOG_FORMAT)
    logger.info(f"Running evaluation with config:\n{str(args)}")
    wandb_logger.init(args)

    if args.use_cached:
        # No need to load backbone model when we're using cached features.
        model = None
    else:
        model = get_model(arch=args.arch, checkpoint_file=args.checkpoint, dataset_name=args.dataset)

    # Compute additional tensors required for the evaluation.
    extra_tensors = get_extra_tensors(args, model)

    # Get test features
    if args.use_cached:
        # No need to create a test loader when we're using cached features
        test_loader = None
    else:
        test_dataset = get_dataset(args.dataset, split=args.eval_split)
        test_loader = DataLoader(
            test_dataset, batch_size=args.batch_size, num_workers=args.n_workers, shuffle=False, drop_last=False
        )
    features, labels = get_features(model, test_loader, args.cache_dir, args.use_cached, args.eval_split)

    start = time.time()
    metrics = evaluate(features, labels, args, extra_tensors)

    episode_time = (time.time() - start) / args.n_episodes
    logger.info(f"Evaluation finished. {round(episode_time, 4)} seconds/episode")
    wandb.summary["episode_time"] = episode_time

    metrics = helpers.dict_cat(metrics)
    log_metrics(metrics)
```

(2)

```python
for i in range(vmf.n_iter):
    loss = vmf.train_step(optimizer=opt)
    losses[i] = loss

    if i % log_interval == 0:
        # Log to WandB and console
        _loss = helpers.npy(loss)
        logger.debug(f"VMF-iter = {i} - Loss = {_loss}")
        if log_wandb:
            wandb_logger.accumulate({"loss.vmf": _loss}, global_step=global_step, local_step=i,
                                    max_local_steps=vmf.n_iter)

    if profiler is not None:
        profiler.step()
```

Experiment tracking and logging - Logging

(3)

```python
def main(arg_list=None):
    args = parse_args(arg_list)
    logging.basicConfig(level=args.log_level, format=config.LOG_FORMAT)
    logger.info(f"Running evaluation with config:\n{str(args)}")
    wandb_logger.init(args)

    if args.use_cached:
        # No need to load backbone model when we're using cached features.
        model = None
    else:
        model = get_model(arch=args.arch, checkpoint_file=args.checkpoint, dataset_name=args.dataset)

    # Compute additional tensors required for the evaluation.
    extra_tensors = get_extra_tensors(args, model)

    # Get test features
    if args.use_cached:
        # No need to create a test loader when we're using cached features
        test_loader = None
    else:
        test_dataset = get_dataset(args.dataset, split=args.eval_split)
        test_loader = DataLoader(
            test_dataset, batch_size=args.batch_size, num_workers=args.n_workers, shuffle=False, drop_last=False
        )
    features, labels = get_features(model, test_loader, args.cache_dir, args.use_cached, args.eval_split)

    start = time.time()
    metrics = evaluate(features, labels, args, extra_tensors)

    episode_time = (time.time() - start) / args.n_episodes
    logger.info(f"Evaluation finished. {round(episode_time, 4)} seconds/episode")
    wandb.summary["episode_time"] = episode_time

    metrics = helpers.dict_cat(metrics)
    log_metrics(metrics)
```

3(a)

3(b)

```python
def log_metrics(metrics):
    # Log metrics (losses, etc) accumulated during evaluations
    wandb_logger.log_accumulated()

    # Log aggregated metrics
    for metric_name, values in metrics.items():
        wandb.summary.update(aggregate_metrics(mtc=values, name=metric_name))

    # Log metric histogram
    data = np.stack(list(metrics.values()), axis=1)
    # Jitter the data a little to avoid duplicate values being filtered out by WandB
    data += np.random.normal(0, 1e-4, size=data.shape)
    # Create table and log histogram
    table = wandb.Table(data=data.tolist(), columns=list(metrics.keys()))
    for metric_name in metrics.keys():
        wandb.log({
            f"{metric_name}.histogram": wandb.plot.histogram(table, metric_name,
                                                             title=f"{metric_name.capitalize()} histogram")
        })
```

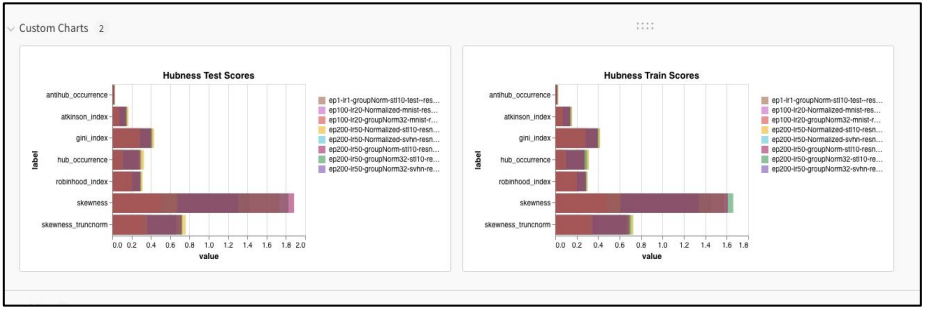
Custom Charts and Data Structures

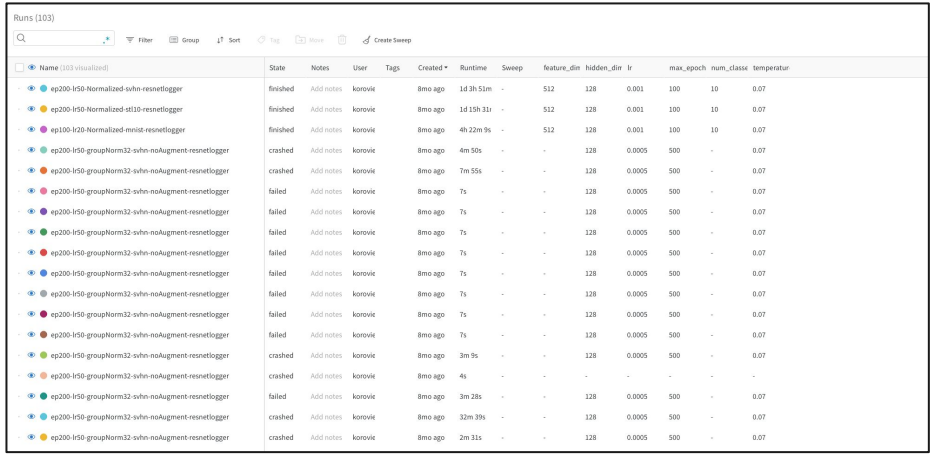
Wandb Tables

Wandb Plots

```python
def log_metrics(metrics):
    # Log metrics (losses, etc) accumulated during evaluations
    wandb_logger.log_accumulated()

    # Log aggregated metrics
    for metric_name, values in metrics.items():
        wandb.summary.update(aggregate_metrics(mtc=values, name=metric_name))

    # Log metric histogram
    data = np.stack(list(metrics.values()), axis=1)
    # Jitter the data a little to avoid duplicate values being filtered out by WandB
    data += np.random.normal(0, 1e-4, size=data.shape)
    # Create table and log histogram
    table = wandb.Table(data=data.tolist(), columns=list(metrics.keys()))
    for metric_name in metrics.keys():
        wandb.log({
            f"{metric_name}.histogram": wandb.plot.histogram(table, metric_name,
                                                             title=f"{metric_name.capitalize()} histogram")
        })
```

UI: Filtering and Tagging





Name (103 visualized)	State	Notes	User	Tags	Created	Runtime	Sweep	feature_dim	hidden_dim	lr	max_epoch	num_classe	temperatur
ep200-lr50-Normalized-svhn-resnetlogger	finished	Add notes	korovie		8mo ago	1d 3h 51m	-	512	128	0.001	100	10	0.07
ep200-lr50-Normalized-stl10-resnetlogger	finished	Add notes	korovie		8mo ago	1d 15h 31	-	512	128	0.001	100	10	0.07
ep100-lr20-Normalized-mnist-resnetlogger	finished	Add notes	korovie		8mo ago	4h 22m 9s	-	512	128	0.001	100	10	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	crashed	Add notes	korovie		8mo ago	4m 50s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	crashed	Add notes	korovie		8mo ago	7m 55s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	7s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	crashed	Add notes	korovie		8mo ago	3m 9s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	crashed	Add notes	korovie		8mo ago	4s	-	-	-	-	-	-	-
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	failed	Add notes	korovie		8mo ago	3m 28s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	crashed	Add notes	korovie		8mo ago	32m 39s	-	-	128	0.0005	500	-	0.07
ep200-lr50-groupNorm32-svhn-noAugment-resnetlogger	crashed	Add notes	korovie		8mo ago	2m 31s	-	-	128	0.0005	500	-	0.07

Sweeps

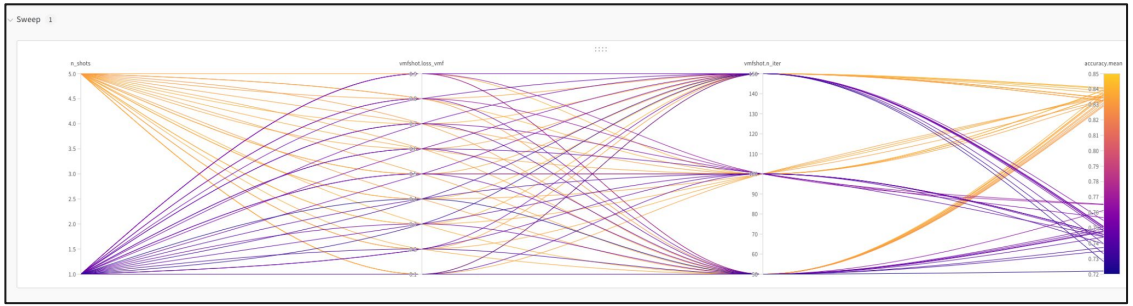
(1) Yml file for setting up a sweep

```yaml
command:
  - python
  - evaluate.py
  - "--config"
  - "config/templates/mini_resnet18.yml"
  - "--norm"
  - "vmf"
  - "--n_episodes"
  - "500"
  - ${args}
method: grid

parameters:
  n_shots:
    values: [1, 5]

  vmf.out_dims:
    values: [2, 8, 32, 128]

  vmf.pca_mode:
    values: [base, episode]

  vmf.re_norm:
    values: [true, false]

  vmf.initial_dims:
    values: [50, None]

  vmf.p_sim:
    values: [vmf, rbf]

  vmf.kappa:
    values: [1, 2, 5]

  vmf.perplexity:
    values: [15, 30, 45]
```

(2) Running a sweep

```
(TIM) rch015@uit-mac-1037 few_shot_learning % wandb sweep -p hubness -e uitmlg sweep/tiered_s2m2_vmfshot.yml
wandb: Creating sweep from: sweep/tiered_s2m2_vmfshot.yml
wandb: Created sweep with ID: is13wt2w
wandb: View sweep at: https://wandb.ai/uitmlg/hubness/sweeps/is13wt2w
wandb: Run sweep agent with: wandb agent uitmlg/hubness/is13wt2w
```

(3) Visualising and filtering



Things to consider

- Good to incorporate in everyday workflow.
- Slow. Very.
- Not the one stop solution: Neptune exists as an alternative.

Thank You