Ray Wrapper: FCNN and CNN

Ray Wrapper: FCNN and CNN

Полносвязная сеть (MNIST) и свёрточная сеть (CIFAR-10), обе — через `ray.train` (распределённое обучение) и `ray.tune` (подбор гиперпараметров поверх распределённого обучения).

Датасеты (`MNIST`/`CIFAR-10`) скачиваются автоматически через `torchvision` в `data_config["data_root"]` (`/tmp/mlumina_data/mnist` и `/tmp/mlumina_data/cifar10` в примерах) — при первом запуске узлам кластера нужен доступ в интернет; при повторных запусках данные переиспользуются из этой директории на каждом узле.

FCNN — `ray_wrapper/examples/fcnn/train.py`

```
train_config = {"hidden1": 256, "hidden2": 128, "dropout": 0.2, "lr": 1e-3, "epochs": 1,
"batch_size_per_worker": 32}
scale_config = {"num_workers": 2, "use_gpu": False}
data_config = {"data_root": "/tmp/mlumina_data/mnist"}

train_fcnn(train_config=train_config, scale_config=scale_config, data_config=data_config)
```

`scale_config` управляет количеством Ray Train воркеров и GPU; `train_config` — гиперпараметрами архитектуры/обучения. Один запуск = одна конфигурация, без перебора.

`ray_wrapper/examples/fcnn/tune.py`

```
search_space = {
    "hidden1": tune.choice([128, 256, 512]),
    "hidden2": tune.choice([64, 128, 256]),
    "dropout": tune.choice([0.1, 0.2, 0.3]),
    "lr": tune.loguniform(1e-4, 1e-2),
    "epochs": 1,
```

```
    "batch_size_per_worker": tune.choice([16, 32, 64]),
}

tune_fcnn(
    search_space=search_space, num_samples=1,
    scale_config={"num_workers": 2, "use_gpu": False},
    data_config={"data_root": "/tmp/mlumina_data/mnist"},
    driver_resources={"CPU": 1}, max_concurrent_trials=2,
    search_algorithm="hyperopt",
)
```

`search_algorithm="hyperopt"` — использует `mlumina.ray_wrapper.build_search_algorithm` (см. книгу «Документация MLumina», страница Ray Wrapper) вместо дефолтного семплера Ray Tune. `num_samples=1` в примере — по сути демонстрация механики, а не реальный подбор (для реального тюнинга это значение обычно увеличивают на порядки).

CNN (CIFAR-10) — `ray_wrapper/examples/cnn/train.py`

```
scale_config = {"num_workers": 2, "use_gpu": False}
train_config = {"c1": 32, "c2": 64, "c3": 128, "lr": 1e-3, "epochs": 3,
"batch_size_per_worker": 16}
data_config = {"data_root": "/tmp/mlumina_data/cifar10"}

train_cnn(
    scale_config=scale_config, train_config=train_config, data_config=data_config,
    mlflow_tracking_uri="http://mlflow-server:5001",
    mlflow_experiment_name="mlumina-cnn-cifar10-train",
    mlflow_run_name="cnn_cifar10_train_example",
)
```

В отличие от FCNN, CNN-пример сразу подключает MLflow-логирование (метрики, не модель — см. страницу Deployment/XGBoost в «Документация MLumina» про то, какие пути в фреймворке реально сохраняют артефакты в MLflow/S3, а какие только метрики). `c1` / `c2` / `c3` — число каналов в трёх свёрточных блоках.

`ray_wrapper/examples/cnn/tune.py` устроен аналогично `fcnn/tune.py` — варьирует `c1` / `c2` / `c3` / `lr` / `batch_size_per_worker` через `tune.choice` / `tune.loguniform`.

??? ?????????

Изнутри кластера (`head`/`worker`), с рабочей директорией `src/mlumina`:

```
python ray_wrapper/examples/fcnn/train.py
python ray_wrapper/examples/fcnn/tune.py
python ray_wrapper/examples/cnn/train.py
python ray_wrapper/examples/cnn/tune.py
```

Дольше всего в первый раз — скачивание датасета; сама тренировка при `epochs=1`/`3` и `num_workers=2` рассчитана на быстрый прогон-демонстрацию, а не на получение хорошей модели.

Revision #1
Created 2026-09-23 06:54:48 UTC by Маслов Алексей
Updated 2026-09-23 06:54:49 UTC by Маслов Алексей