💡 Introduction to Ray AI Libraries

About

Welcome to the Intro to Ray AI Libraries class at Ray Summit 2023! We're so glad to have you along for this gentle introduction. The road to production-grade ML at scale can be riddled with hairpin challenges: infrastructure complexities, vendor headaches, and the constant specter of cost management. Today is all about putting you in the driver's seat in this fun and practical class that enables you to build scalable machine learning pipelines at your own pace.

The Ray AI Libraries are a collection of open source libraries designed to simplify and enhance scaling machine learning workloads. Built on top of the Ray distributed framework, they inherit all of performance optimizations of Ray Core while providing an user-friendly, Python-native interface for development. Today, we'll walk you through how to scale deep learning applications, covering the most widely used workloads in the MLOps lifecycle: data handling, fine-tuning and training, hyperparameter tuning, batch prediction, and serving. Let's get started!

Meet the team

Emmy Li	Balaji Veeramani	Yunxuan Xiao
emmy@anyscale.com	balaji@anyscale.com	yunxuanx@anyscale.com

How to participate

Join live polls and engage in Q&A by going to **app.sli.do** and enter code `#Ray-AILibs`. You can also ask questions live by calling over an instructor.

How to access Anyscale

Anyscale, the company founded by the creators of Ray, simplifies the development of distriubted applications for machine learning. Simply put, it's the best place to run Ray, and today, you'll have access to a provisioned cluster with zero set-up.

1	Log-in via **console.anyscale.com** with the credentials sent to your email.
2	Select "Workspaces" in the side panel and open your workspace.

3	Click on the Jupyter or VSCode icon to access the course materials.

0_Quick_Introduction.ipynb

This notebook condenses the general pattern for composing the Ray AI Libraries into an end-to-end scalable pipeline for machine learning. We'll use a basic XGBoost model and tabular NYC taxi data as an example, but the focus is really on how to visualize the way these components interact with one another to distribute different kinds of workloads.

🟦 **Terminology**

Head node: A node (only one) that runs extra cluster-level processes like development tools and is responsible for distributing workloads to worker nodes at runtime.

Worker nodes: A physical or virtual machine, orchestrated by the head node to run worker processes, or jobs.

Ray Cluster: A set of worker nodes connected to a common Ray head node. They have the ability to autoscale up and down according to the resources requested by applications running on the cluster.

🔎 **Insight: Integrations**

Ray is a *very general* framework for distributed computing. The best tools are the ones that integrate well with existing solutions, and Ray excels in bringing together this ML ecosystem under a common interface for distributed workloads.

- Reflect on your current tech stack for ML workloads. Can you identify integrations and connections that Ray has with your current tools? Ask a TA if you're unsure!

✏️ **Your notes**

_____

_____

_____

_____

1_Base_Hugging_Face.ipynb

Remember, we're walking through this example so that you have a foundational reference point to compare what a machine learning workload looks like before and after you adapt it for use

with Ray. Of course, Ray is not specific to Hugging Face or transformer architectures. This is merely an example to get you started!

✏️ **Your notes**

_____

_____

_____

_____

2_Add_Ray_Train.ipynb

✒️ **Spotlight on Ray Train**

Ray Train is a scalable machine learning library built on top of Ray, designed to facilitate distributed training and fine-tuning of machine learning models across multiple GPUs and machines. It provides scalability, resource efficiency, and seamless integration with existing ML frameworks like PyTorch and TensorFlow. Its capabilities ease the transition from development to production by maximizing hardware utilization and simplifying the management of large-scale, distributed training tasks.

🏋️ **Exercises**

1. Observability
   a. When running production-grade fine-tuning and training jobs, having a solid observability story is key. Try rerunning the training job in this notebook with the Ray Dashboard open. Pay attention to your GPU utilization, memory, logging, or anything else you're interested in.
   b. Try changing the `ScalingConfig` to use a different number of workers or a different type of compute. Run it again while looking at the Ray Dashboard. How does Ray handle these different configurations?

✨ **Fun Fact**

In terms of usage, the fastest growing Ray AI Library is Ray Train (now GA!), with the most popular composition of libraries being Train + Data.

✏️ **Your notes**

_____

_____

3_Add_Ray_Data.ipynb

🔦 **Spotlight on Ray Data**

Ray Data is a scalable data processing library designed for machine learning workloads, providing flexible and performant APIs for tasks like batch inference, data preprocessing, and data ingestion. It enables efficient data loading and parallel data processing, reducing bottlenecks and accelerating model training. By seamlessly integrating with Ray Train, Ray Data streamlines the end-to-end machine learning workflow, making it easier to scale and productionize ML models.

🏋️ **Exercises**

1. Batch mapping is a common ML operation, whether it be for generating offline predictions or applying more general data transformations.
   a. The Ray Data ActorPoolStrategy specifies the autoscaling behavior of a Dataset transformation. Instead of a fixed-size pool, try setting a `min_size` and a `max_size` and observe the resulting behavior in the Ray Dashboard.
   b. We often choose the size of each batch according to available GPU memory through trial and error. Try adjusting the batch size when featurizing images, and pull up the Ray Dashboard to see the results.

✏️ **Your notes**

4_Add_Ray_Serve.ipynb

📑 **Glossary**

Ray Serve: A scalable and flexible model-serving library built on top of the Ray distributed computing framework, designed to simplify the deployment and management of machine learning models and other services.

Ray Cluster: A set of interconnected nodes managed by the Ray framework to distribute and parallelize computation tasks and data across multiple machines.

Node: A Ray node is a physical or virtual machine that can run one or multiple Ray processes to execute tasks.

Deployment: A user-defined, versioned unit of code, that can contain ML models or business logic and is encapsulated as a Python class or function. It can be horizontally scaled and accessed via HTTP or Python APIs.

Replica: An individual instance of a deployment, running as a separate Ray Actor, that handles incoming requests and can be autoscaled to adjust to the volume of incoming traffic.

ServeHandle: A ServeHandle is a reference to a bound deployment that allows for programmatic interaction with the deployment. This allows multiple independent deployments to call into each other and facilitates flexible and complex model composition where bound deployments can reference other bound deployments.

Application: An application is composed of one or more deployments and can be accessed via HTTP routes or Python handles, working together to provide a specific service or functionality.

🏋️ **Exercises**

1. Write some code to generate a large number of requests to this service.
   a. Open the Ray Dashboard and look at the “Serve” tab. Look through the logs, metrics, and autoscaling behavior for your service.
   b. Open Grafana for additional metrics and advanced dashboard visualization.

✨ **Fun Fact**

Ray Serve (now GA!) is the most popular Ray AI Library, and that’s reflected in the general industry trends. The enduring sentiment is, “You train once, but serve forever.”

✏️ **Your notes**

_____

_____

_____

_____

Feedback survey

We're so glad to have you along in our classroom. To improve our content and presentation for the next batch of future learners, we would love to hear how your experience was.

Please visit **bit.ly/ray-summit-feedback** to let us know how today went!

More resources

Self-Paced Ray & Anyscale Education	Access the best course materials from Ray Summit and get a sneak preview of technical content releases before they become public!
Docs	Your one-stop-shop for all things Ray at **docs.ray.io**
Blogs	Read our latest news and findings at **anyscale.com/blog**
YouTube	Watch a curated set of tutorials at **youtube.com/anyscale**
Anyscale	Interested in a managed Ray service? Go to **anyscale.com/sign-up**