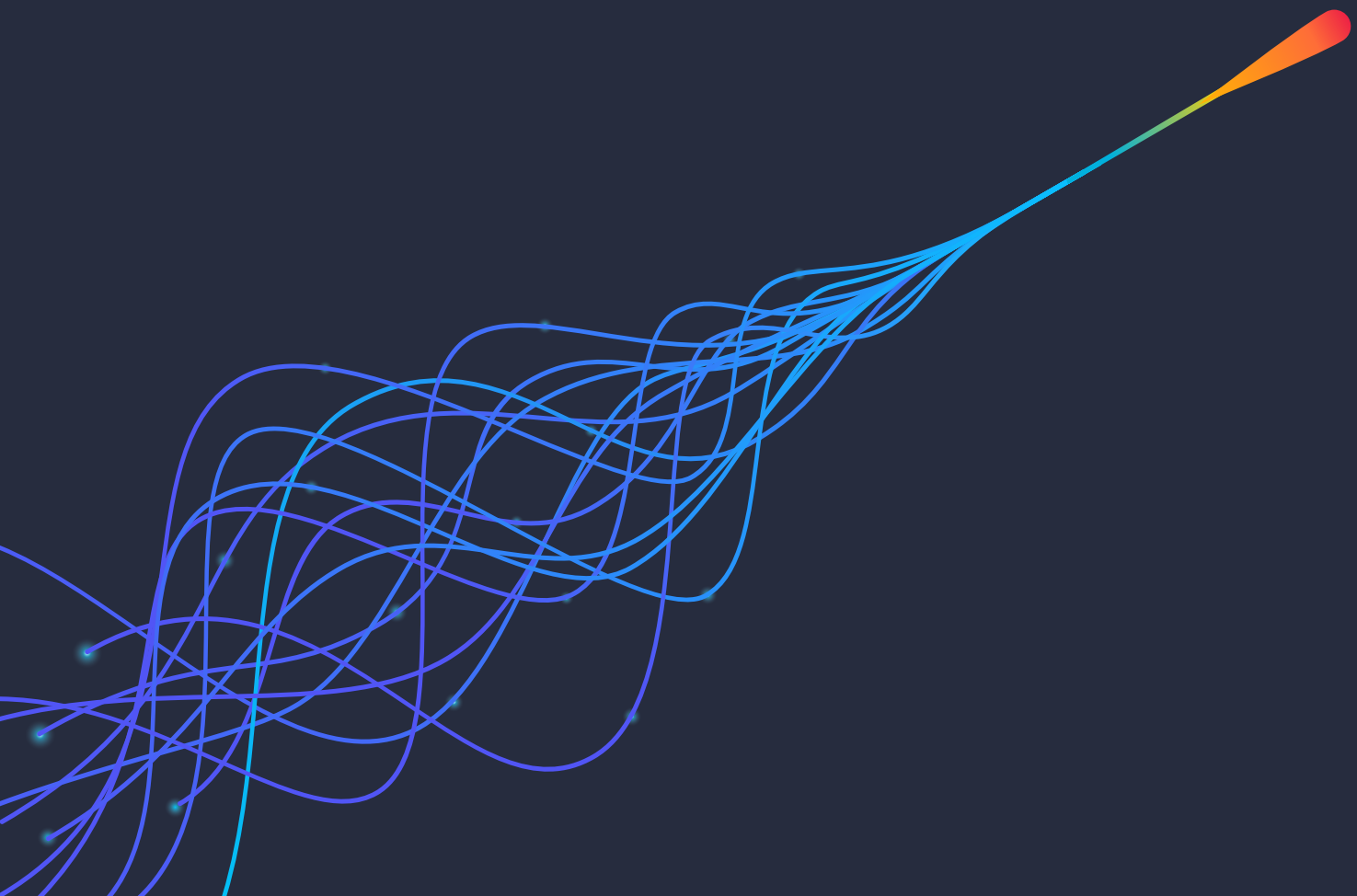
comet

Sagemaker



Introduction

Sagemaker is Amazon's end-to-end machine learning service that targets a large swath of data science and machine learning practitioners. With Sagemaker, data scientists and developers can build and train machine learning models, and directly deploy them into a production-ready hosted environment. Sagemaker's product offerings span the needs of data/business analysts, data scientists, and machine learning engineers.

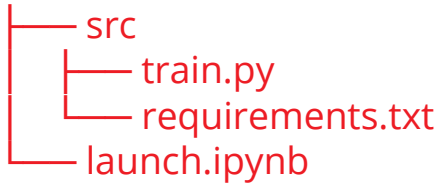
Comet is an excellent complement to Sagemaker, enhancing the developer experience by allowing users to easily track experiments, collaborate with team members, and visualize results in an intuitive and easy-to-understand way while using the frameworks and tools that they are most comfortable with. Additionally, the platform provides a wide range of customization options, including the ability to create custom visualizations and dashboards, so that users can tailor their experience to meet their specific needs.

By using Comet, users can streamline their workflows while benefiting from Sagemaker's powerful infrastructure orchestration and model deployment capabilities.

How does Comet work with Sagemaker?

Logging Data from the Sagemaker SDK

Comet requires minimal changes to your existing Sagemaker workflow in order to get up and running. Let's take a look at a simple example that uses the Sagemaker SDK and Notebook instances to run a custom script.



Your `src` directory would contain the model specific code needed to execute your training run, while `launch.ipynb` would run in your Notebook instance, and contain code related to configuring and launching your job with the Sagemaker SDK.

To enable Comet logging in this workflow, simply

1. Add `comet_ml` as a dependency in your `requirement.txt` file
2. Import the `comet_ml` library of the `train.py` script
3. Create a Comet `Experiment` object within the training script

```
# import sagemaker_containers
import torch
import torch.distributed as dist
import torch.nn as nn
import torch.optim as optim
import torch.utils.data
import torch.utils.data.distributed
from torchvision import datasets, transforms
```

```
+ import comet_ml
# import sagemaker_containers
import torch
import torch.distributed as dist
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torch.utils.data
import torch.utils.data.distributed
from torchvision import datasets, transforms
+
+ experiment = comet_ml.Experiment()
```

1. Pass in your Comet Credentials to the Sagemaker Estimator using the `experiment` argument

```
estimator = PyTorch(
    source_dir="src",
    entry_point="mnist.py",
    role=role,
    py_version="py38",
    framework_version="1.11.0",
    metric_definitions=[
        {'Name':'train:loss', 'Regex':'Train Loss:
(.*?);'},
        {'Name':'test:loss', 'Regex':'Test Average Loss:
(.*?);'},
        {'Name':'test:accuracy', 'Regex':'Test Accuracy:
(.*?)%;'}
    ],



)
```

```
estimator = PyTorch(
    source_dir="src",
    entry_point="mnist.py",
    role=role,
    py_version="py38",
    framework_version="1.11.0",
    metric_definitions=[
        {'Name':'train:loss', 'Regex':'Train Loss:
(.*?);'},
        {'Name':'test:loss', 'Regex':'Test Average Loss:
(.*?);'},
        {'Name':'test:accuracy', 'Regex':'Test Accuracy:
(.*?)%;'}
    ],
+   environment={
+       "COMET_API_KEY": "<Your API Key>",
+       "COMET_PROJECT_NAME": "<Your Project Name>"
+   }
  )
```

To log your Sagemaker run with Comet, execute it as you normally would using `estimator.fit` . Comet will handle the logging for you. If you want to customize the logged data, such as confusion matrices, gradient histograms, or multimedia data, add the relevant functionality to the `train.py` script using any of the logging methods available in the Experiment object.

For example, we can log predictions and images to Comet's Interactive Confusion Matrix from our script by adding the following function.

```
def log_predictions(model, dataloader):
    to_pil = transforms.ToPILImage()

    dataset = dataloader.dataset

    labels = []
    predictions = []
    images = []

    for image, label in dataset:
        images.append(to_pil(image))
        labels.append(label)

        output = model(image.unsqueeze(0))
        pred = output.argmax(dim=1, keepdim=True)
        predictions.append(int(pred.item()))

    experiment.log_confusion_matrix(labels, predictions, images=images)
```

Since Comet runs in the same environment as your training script, it has access to all the Sagemaker environment variables as well. This allows you to log Sagemaker specific metadata such as the Training Job Name, S3 Resource URIs, Instance Type etc.

If you would like to see examples of how Comet and Sagemaker can be used together, please visit our Sagemaker Examples page.

Note: The extent of Comet's auto-logging capabilities can vary depending on the framework being used. For more information on logging for specific frameworks, please refer to our integration documentation.

Migrating Existing Sagemaker Training Jobs to Comet

Another option for logging data from Sagemaker runs to Comet is to use Comet's built-in Sagemaker integration to migrate data from completed Sagemaker Training Jobs. The following snippet can be run in a Sagemaker Notebook instance or standalone script.

```
from comet_ml.integration.sagemaker import log_sagemaker_training_job_v1

COMET_API_KEY = "<Your Comet API Key>"
COMET_WORKSPACE = "<Your Comet Workspace>"
COMET_PROJECT_NAME =  "Your Comet Project Name"

TRAINING_JOB_NAME = "<Your training job name>"

log_sagemaker_training_job_by_name_v1(
    TRAINING_JOB_NAME,
    api_key=COMET_API_KEY,
    workspace=COMET_WORKSPACE,
    project_name=COMET_PROJECT_NAME
)
```

Find out more about migrating Sagemaker runs in our Sagemaker Examples page.

Why use Comet with Sagemaker?

Simplicity

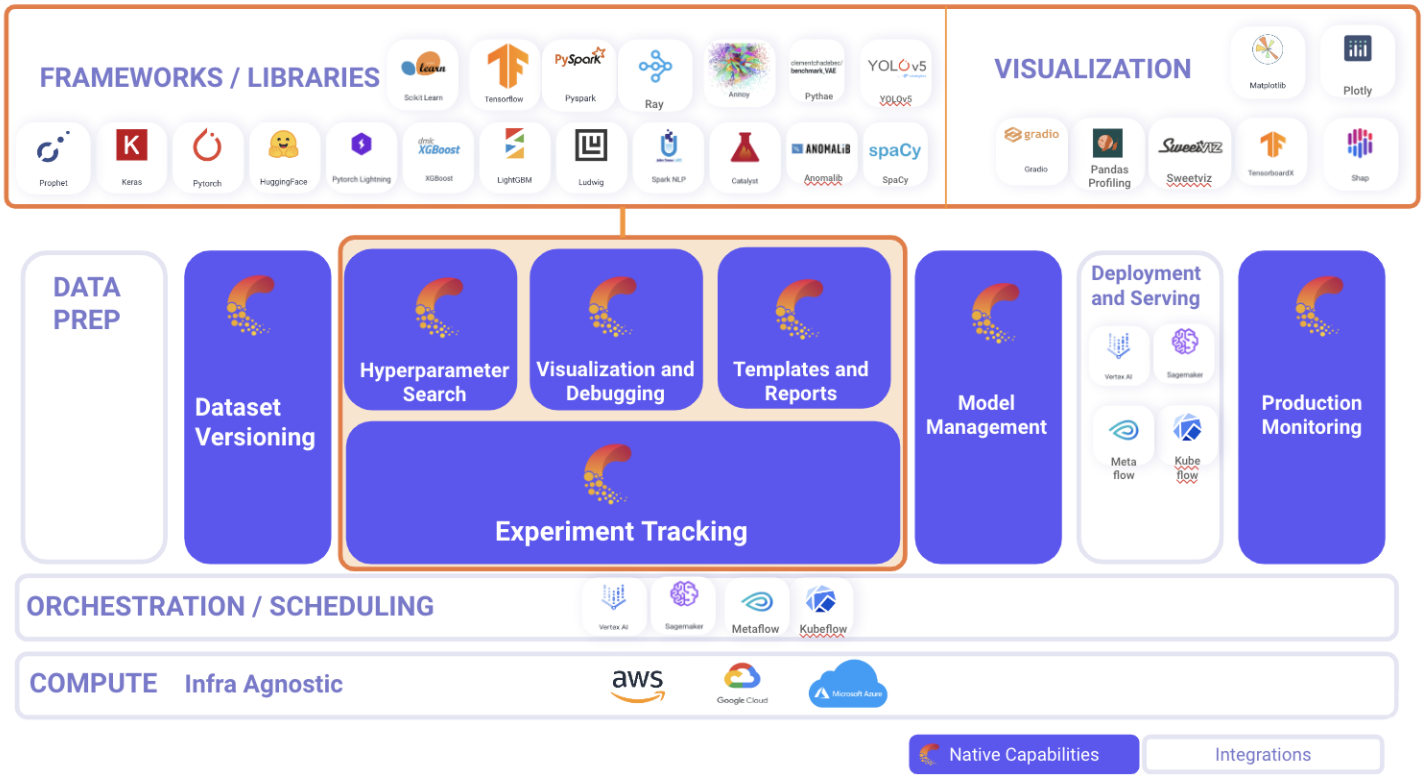
Comet is a comprehensive and user-friendly machine learning platform that emphasizes a lightweight, API-focused approach. Its features are accessible through a single SDK, which is available in multiple programming languages, as well as through a REST API.

Comet serves as a knowledge base for ML projects, offering practitioners a single tool to log, visualize, and share any type of data related to their experiments. This enables them to spend less time piecing together disparate technologies and infrastructure for tracking ML runs, and more time developing models.

In contrast, bundled platforms are typically composed of different services from the platform provider. For example, Sagemaker requires users to understand IAM roles and permissions to enable the use of specific services, knowledge of the AWS S3 SDK to access run assets such as plots or models, and Cloudwatch to understand resource utilization. Users tend to be forced into using multiple services to accomplish the singular task of training a model.

Extensibility

Comet integrates with a variety of external libraries and tools, allowing users to customize the components of the ML stack according to their specific requirements. The pace of development in ML is rapid, and it is difficult for any single organization to anticipate what workflows, tools, and best practices will be dominant in the future. The extensibility of the Comet platform ensures that it will continue to adapt and evolve with the field of ML.



Bundled platforms such as Sagemaker offer a wide range of features that cover a broad set of functions. However, it is challenging to keep this broad feature set up to date since every new component needs to work with the rest of the platform. Additionally the choice of components being integrated into the platform might not match every user's needs or expectations.

On the other hand, best-of-breed services offer greater flexibility and customization options for those with specific needs. These services can be tailored to meet the unique requirements of individual projects, enabling users to achieve greater precision and accuracy in their results.

To find out more about our supported integrations, please visit the integrations page in our documentation.

Avoiding Vendor Lock In

When committing to an end-to-end platform, it often means that your workflow is closely tied to the product offerings of the platform. This can limit your flexibility in terms of the technologies you can use to develop your model. Interested in using TPUs or IPU's instead of GPUs? Do you want to use a way of storing data that isn't S3? These tasks would not be possible when using a bundled platform, and would require a costly migration to another provider.

However, Comet stands out in that it is infrastructure agnostic. This means that code which utilizes Comet can run in any cloud provider's environment, as well as on-premise. This flexibility enables users to customize their infrastructure according to their specific needs, without worrying about vendor lock-in.

Conclusion

Sagemaker and Comet offer complementary sets of tools for building a powerful machine learning platform. Sagemaker excels in resource management, computing, and deployment, while Comet leads in experiment management, artifact management, and production monitoring. Comet also provides a flexible user interface for analysis and reporting. By using both Comet and Sagemaker together, teams can access a comprehensive set of tools that enable them to effectively and efficiently manage, monitor, and deploy their ML models.