Why Machine Learning Pipelines?

Why do you need pipelines for ML workflows? Training and deploying a PyTorch-based model encapsulates a sequence of tasks such as:
- processing data,
- training a model,
- hyperparameter tuning,
- evaluation, packaging the model artifacts,
- model deployment and retraining cycle.

Each of these steps has different dependencies; treating the workflow as a monolith can quickly become unwieldy.

TorchX with Kubeflow Pipelines

TorchX is intended to allow making cross platform components. As such, it has standard definition that uses adapters to convert it to the specific pipeline platform. TorchX provides an adapter, torchx.pipelines.kfp.adapter, to run TorchX components as part of Kubeflow Pipelines. Kubeflow Pipelines is a platform for building and deploying portable, scalable machine learning (ML) workflows based on Docker containers.

In order to build your end-to-end pipeline you need to:
1. Set the input arguments
2. Create the components
3. Write the pipeline definition

Productionizing Pytorch with Vertex AI Pipelines

```python
from kfp.v2 import compiler
from kfp import compiler
import google.cloud.aiplatform as vertex_ai
from torchx import torchx

@vertex_ai.pipeline
def pipeline():
    ...
```

Each of these steps has different dependencies; treating the workflow as a monolith can quickly become unwieldy.

As the ML systems and processes begin to scale, you should share your ML workflow with others on your team to execute the workflows or contribute to the code. Automating these tasks and orchestrating them across multiple services enables repeatable and reproducible ML workflows that can be shared between data scientists and engineers.

Pipelines are also crucial to MLOps when formalizing training and deployment operationalization to automatically retrain, deploy and monitor models. For example, triggering a pipeline run when new training data is available, retraining a model when the model’s performance starts decaying, and more such scenarios.

Productionizing Pytorch with Vertex AI Pipelines

```python
from kfp.v2 import compiler
from kfp import compiler
import google.cloud.aiplatform as vertex_ai
from torchx import torchx

@vertex_ai.pipeline
def pipeline():
    ...
```

As the ML systems and processes begin to scale, you should share your ML workflow with others on your team to execute the workflows or contribute to the code. Automating these tasks and orchestrating them across multiple services enables repeatable and reproducible ML workflows that can be shared between data scientists and engineers.

Pipelines are also crucial to MLOps when formalizing training and deployment operationalization to automatically retrain, deploy and monitor models. For example, triggering a pipeline run when new training data is available, retraining a model when the model’s performance starts decaying, and more such scenarios.