Beam ML past, present and future

Kerry Donny-Clark & Reza Rokni
I’m Kerry
I’m Reza
Agenda

- Intro
- The Past: RunInference is born
- The Present: Model handlers, model updates, model zoos, and more!
- The Future: Make ML tasks easy
- Q&A
Turn key solutions....
Apache Beam

Primitives are great... but...
Patterns are great ... but

Common pipeline patterns
Patterns are great ... but

Which set of patterns do I need...?
Patterns are great ... but

Common pipeline patterns

Set reminder..
Patterns are great ... but

Common pipeline patterns

Using the Documentation

Writing that code!
Writing same code!
Yup same again!
Déjà vu?
with beam.Pipeline(options=pipeline_options) as p:
    (p
    | beam.io.fileio.MatchFiles(gs://my_bucket/*)
    | beam.io.fileio.ReadMatches()
    | beam.Map(preprocess_image)
    | beam.xxx_pattern_xxx(Configuration)
    ...

Turn Key transforms
Self-service Machine Learning Workflows and Scaling MLOps with Apache Beam

Apache Beam has future-proofed Credit Karma’s data and ML platform for scalability and efficiency, enabling MLOps with unified pipelines, processing 5-10 TB daily at 5K events per second, and managing 20K+ ML features.

Real-time ML with Beam at Lyft

Lyft Marketplace team aims to improve our business efficiency by being nimble to real-world dynamics. Apache Beam has enabled us to meet the goal of having a robust and scalable ML infrastructure for improving model accuracy with features in real-time. These real-time features support critical functions like...
## Beam Summit 2022:

<table>
<thead>
<tr>
<th>Topic</th>
<th>Presenter(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Machine learning design patterns between Beam and a hard place</td>
<td>Lak Lakshmanan</td>
</tr>
<tr>
<td>Vega: Scaling <strong>MLOps Pipelines</strong> at Credit Karma using Apache Beam and Dataflow</td>
<td>Debasish Das, Vishnu Venkataraman &amp; Raj Katakam</td>
</tr>
<tr>
<td>Implementing Cloud Agnostic <strong>Machine Learning Workflows</strong> with Apache Beam on Kubernetes</td>
<td>Charles Adetiloye &amp; Alexander Lerma</td>
</tr>
<tr>
<td>Streaming <strong>NLP infrastructure</strong> on Dataflow</td>
<td>Alex Chan &amp; Angus Neilson</td>
</tr>
<tr>
<td>Improving Beam-Dataflow Pipelines for <strong>Text Data Processing</strong></td>
<td>Sayak Paul &amp; Nilabhra Roy Chowdhury</td>
</tr>
</tbody>
</table>
Beam RunInference
with beam.Pipeline(options=pipeline_options) as p:
  (p
   | beam.io.fileio.MatchFiles(gs://my_bucket/images*)
   | beam.io.fileio.ReadMatches())
with beam.Pipeline(options=pipeline_options) as p:

    (p
     | beam.io.fileio.MatchFiles(gs://my_bucket/images*)
     | beam.io.fileio.ReadMatches()
     | beam.Map(preprocess_image)
class MyComplicatedPredictionStuff(beam.DoFn):

Removing boiler plate chore!
class MyComplicatedPredictionStuff(beam.DoFn):
    def setup():
        # Code for loading once
        ...

    def process(self, element):
        # Use model handle to call
class MyComplicatedPredictionStuff(beam.DoFn):
    def setup():
        #Code for loading once
        ...

    def process(self, element):
        #Use model handle to call
        ...

        #Handle errors, do nice error logging
class MyComplicatedPredictionStuff(beam.DoFn):
    def setup():
        # Code for loading once
        ...

    def process(self, element):
        # Use model handle to call
        ...
        ...
        # Handle errors, do nice error logging
        ...
        # Output useful metrics from the process
        ...
Removing boiler plate chore!

class MyComplicatedPredictionStuff(beam.DoFn):
    def setup():
        #Code for loading once
        ...

    def process(self, element):
        #Use model handle to call
        ...
        #Handle errors, do nice error logging
        ...
        #Output useful metrics from the process
        ...
        TODO Oh wait! I need to batch stuff first ...
class MyComplicatedPredictionStuff(beam.DoFn):
    def setup():
        TODO Code for loading once ..... 

    def process(self, element):
        TODO Use model handle to call
        TODO Handle errors, do nice error logging
        TODO Output useful metrics from the process
        TODO Oh wait! I need to batch stuff first ...
        TODO Wait.. I need model configuration ....
with beam.Pipeline(options=pipeline_options) as p:
    (p
    | beam.io.fileio.MatchFiles('gs://my_bucket/images*')
    | beam.io.fileio.ReadMatches()
    | beam.Map(preprocess_image)
    | beam.ml.inference.RunInference(model_handler)
    ...
Pytorch Model Handler Tensor

Framework  Type
The Present: Model Handlers

- TensorFlow
- PyTorch
- XGBoost
- scikit-learn
- TensorRT
- ONNX
The Present: Notebooks

https://github.com/apache/beam/tree/master/examples/notebooks/beam-ml

- Prediction and inference with pretrained models
- Custom inference
- Automatic Model Refresh
- Multi-model pipelines
- Model Evaluation
- Data processing
DEMO

LLM Demo Link
with pipeline as p:
predictions = (p
  | ReadFromText(known_args.input)
  | RunInference(
    TensorHubhandler(
      {uri=CLASSIFIER_URL}
    )))
DEMO

notebooks/beam-ml/run_inference_with_tensorflow_hub.ipynb
inference = pcoll | RunInference(model_handler.with_postprocess_fn(lambda x : do_something_to_result(x)))

inference = pcoll | RunInference(model_handler.with_preprocess_fn(lambda x : do_something(x)))

inference = pcoll | RunInference(
    model_handler.with_preprocess_fn(
        lambda x : do_something(x)
    ).with_preprocess_fn(
        lambda x : do_something_else(x)
    ).with_postprocess_fn(
        lambda x : do_something_after_inference(x)
    ).with_postprocess_fn(
        lambda x : do_something_else_after_inference(x)
    ))
main, other = pcoll | RunInference(model_handler).with_exception_handling()
other.failed_inferences | beam.Map(print) # insert logic to handle failed records here

main, other = pcoll | RunInference(model_handler.with_preprocess_fn(f1).with_postprocess_fn(f2)).with_exception_handling()
other.failed_preprocessing[0] | beam.Map(print) # handles failed preprocess operations, indexed in the order in which they were applied
other.failed_inferences | beam.Map(print) # handles failed inferences
other.failed_postprocessing[0] | beam.Map(print) # handles failed postprocess operations, indexed in the order in which they were applied
RunInference auto-model update

Two modes:
1. Watch Mode
   Upload updated model to files stores like GCS and RunInference will auto pull the new model for you
2. Event Mode
   Push an update message to RunInference via a streaming source such as Kafka
The Present: Efficient large models

```python
share_model_across_processes() → bool
```

*apache_beam.utils.multi_process_shared module*
The Present: Multi-Model Ensembles

Voice
- Speech to Text
- Language Understanding
- Sentiment Analysis
- Logs

Text
- Text to Speech
- Language Understanding
- Product Recommender
- Support Recommender

Response
The Present: Branched (A/B) models

```
data = p | beam.io.textio(files)
data | RunInference(model_a_handler)
data | RunInference(model_b_handler)
```
The Present: Sequential Models

data = p | beam.io.textio(files)

model_a_output =
data | RunInference(model_a_handler)

model_a_output | Map(postprocess) | RunInference(model_b_handler)
Hugging Face Model Handler for RunInference

*Ritesh Ghorse* ([ritesghorse@apache.org](mailto:ritesghorse@apache.org))

```python
with pipeline as p:
    predictions = (
        p
        | beam.ReadFromSource('a_source')
        | RunInference(
            HuggingFaceModelHandler(...))
```
Future: Models from endpoints

[WIP] Vertex AI Remote Model Handler #27091

Draft jrmccluskey wants to merge 3 commits into apache:master from jrmccluskey:vertexAI
Beyond Inference!
Beyond Inference!

1. **Question**
2. **Explore Data, extract features**
   - **Yes**: Train ML model
   - **No**: Get new data
3. **Can the data answer the question?**
   - **Yes**: Train ML model
   - **No**: Get new data
4. **Train ML model**
5. **Evaluate the model**
   - **OK!**: Serve the model
   - **!OK**: New model is better
5. **Update the model**
5. **Compare models**
5. **Train an updated model**
5. **Evaluate the model**
   - **OK!**: Serve the model
   - **!OK**: New model is better
   - **OK!**: Serve the model
Beyond Inference!

Serve the model

- Framework, data shape
- User defined metrics
- Fine tuning, or de novo?

Update the model

- Rollout strategy

Evaluate the model

- Same metrics

Train an updated model

Compare models

- Rollout strategy

Same metrics
Coming soon

Beam MLTransform

https://s.apache.org/beam-mltransform

Anand Inguva (anandinguva@google.com)

Last updated: May 30th, 2023
beam.MLTransform(
    process_handler=ProcessHandler(
        scale_to_0_1(
            columns={'x': int, 'y': List[int]},
        compute_and_apply_vocab(
            columns={'x': int, 'y': List[int]})
    )
)
Coming soon
QUESTIONS?