Write your own model handler for Run Inference
About me

- Based in Durham, NC
- Contributions:
  - Beam Go SDK 🐶
  - Python/ML 🤖
- Cricket, F1
Why do we need a model handler?

Parameter to RunInference Transform

```python
with beam.Pipeline() as p:
    _ = (p | beam.Create(value_to_predict)
        | RunInference(model_handler)
        | beam.ParDo(FormatOutput())
        | beam.Map(print)
    )
```
What is a model handler?

- Class with defined input and output types

The base class looks like

```python
class ModelHandler(Generic[ExampleT, PredictionT, ModelT]):

where ExampleT - type of Input (Numpy)

PredictionT - type of output

ModelT - type of Model class (tf.Module)
```
What is a model handler?

- Specific Framework
- Avoids repetitive steps like:
  - loading and initializing a model
  - defining inference function
- Automatic model refresh with Beam side inputs.
- Share model between processes
- Write once to do inference with a single line of code.
Key components of a Model Handler

- **input/output types**
- **load_model(self)**
  - load and return the model
- **run_inference(self, batch, inference_args)**
  - perform run inference inference
- **update_model_path(self, new_path)**
  - replace the old model with newly trained model
- **get_num_bytes(self)**
  - return the size of batch of elements. (Used internally by RunInference)

Reference: [https://github.com/apache/beam/blob/639dcf70b27b667cca0816a0d35ef7fb992f758c/sdks/python/apache_beam/ml/inference/base.py#L122](https://github.com/apache/beam/blob/639dcf70b27b667cca0816a0d35ef7fb992f758c/sdks/python/apache_beam/ml/inference/base.py#L122)
Let’s write our model handler

Step 1: Decide the input types to support. Eg: Numpy, tensors, etc.
  - Let’s take tensorflow tensors for the example

Step 2: Extend the base `ModelHandler` class
Let’s write our model handler

Example:

Let’s write a TensorFlow model handler that could take `tf.Tensor` input, output as `PredictionResult` and `tf.Module` as the model class.

class TFModelHandlerTensor(ModelHandler[tf.Tensor, PredictionResult, tf.Module]):

where `PredictionResult` is a `NamedTuple` that stores `example` and `inference` for that example.
Let’s write our model handler

Step 3: Figure out what would be needed to load the model

In case of TensorFlow we could load the model in two ways:

1. By using an model URI (either from TensorFlow Hub or other filesystem)
2. Path to weights and a function to create the model

Let’s just focus on point 1) for the sake of an example
Let’s write our model handler

Example:

class TFModelHandlerTensor(ModelHandler[tf.Tensor, PredictionResult, tf.Module]):
    def __init__:
        self,
        model_uri: str,
        *,
        load_model_args: Optional[Dict[str, Any]] = None,
        inference_fn: TensorInferenceFn = default_tensor_inference_fn,
        **kwargs):
            self. model_uri = model_uri
            self. inference_fn = inference_fn
            self._load_model_args = {} if not load_model_args else load_model_args
Let’s write our model handler

Step 4: Let’s write our load model function

It should load and return the model from this method

def load_model(self) -> tf.Module:
    model = tf.keras.models.load_model(hub.resolve(self.model_uri), **self.load_model_args)
    return model
Step 5: Run Inference

```python
def run_inference(self, batch: Sequence[numpy.ndarray], model: tf.Module, inference_args: Optional[Dict[str, Any]] = None) -> Iterable[PredictionResult]:
inference_args = {} if not inference_args else inference_args
return self._inference_fn(model, batch, inference_args, self._model_uri)
```

```python
    vectorized_batch = tf.stack(batch, axis=0)
predictions = model(vectorized_batch, **inference_args)
return utils._convert_to_result(batch, predictions, model_id)
```
Let’s write our model handler

Step 6: Automatic model refresh

def update_model_path(self, model_path: Optional[str] = None):
    self._model_uri = model_path if model_path else self._model_uri

Talk on ML model updates by Anand Inguva in Horizon at 15:30.
Other methods for model handler

- **get_metrics_namespace**
  - returns a string namespace

- **get_resource_hints**
  - returns resource hints as a dictionary for model handler

- **batch_elements_kwargs**
  - return a dictionary {'min_batch_size': 1, 'max_batch_size': 32}

- **share_model_across_processes**
  - return a boolean value
  - for large models
class TFModelHandlerTensor(ModelHandler[tf.Tensor, 
PredictionResult, tf.Module]):
    def __init__ ( 
        self, 
        model_uri: str, 
        *, 
        load_model_args: Optional[Dict[str, Any]] = None, 
        inference_fn: TensorInferenceFn = 
default_tensor_inference_fn, 
        **kwargs):
        self._model_uri = model_uri 
        self._inference_fn = inference_fn 
        self.load_model_args = () if not load_model_args else load_model_args 

def load_model(self) -> tf.Module: 
    model = 
    tf.keras.models.load_model(hub.resolve(self._model_uri), 
**(self.load_model_args) 
    return model

def run_inference( 
    self, 
    batch: Sequence[numpy.ndarray], 
    model: tf.Module, 
    inference_args: Optional[Dict[str, Any]] = None 
) -> Iterable[PredictionResult]:
    inference_args = {} if not inference_args else inference_args 
    return self._inference_fn(model, batch, inference_args, 
self._model_uri)

def update_model_path(self, model_path: Optional[str] = None): 
    self._model_uri = model_path if model_path else 
self._model_uri 

def get_num_bytes(self, batch: Sequence[numpy.ndarray]) -> int: 
    return sum(sys.getsizeof(element) for element in batch)
Let’s write our model handler

Example pipeline:

```python
import apache_beam as beam
from apache_beam.ml.inference.base import RunInference

test_examples = [20, 40, 60, 90]
value_to_predict = tf.constant(test_examples, dtype=tf.float32)

model_handler = TFModelHandlerTensor(saved_model_path)

with beam.Pipeline() as p:
    _ = (p | beam.Create(value_to_predict)
         | RunInference(model_handler)
         | beam.ParDo(FormatOutput())
         | beam.Map(print)
    )
```
As simple as

```python
from apache_beam.ml.inference.base import KeyedModelHandler
model_handler = KeyedModelHandler(TFModelHandlerTensor)
```
Current model handlers

- onnx
- pytorch
- sklearn
- tensorflow
- tensorrt
- xgboost

Coming soon...

- Hugging Face
- Vertex AI

Related Resources

- Demo Notebook
- RunInference in Beam talk from Beam Summit 2022
- Example notebook with TensorFlow Model Handler
- Example ML notebooks
- Design Doc of Run Inference API
QUESTIONS?

linkedin.com/in/ritesh-ghorse

github.com/riteshghorse