Build your high-performance model inference solution with DJL and ONNXRuntime

Qing Lan
SDE
AWS

DJL – The Machine Learning framework in Java

- NumPy like operators
- Engine Agnostic: Apache MXNet, TensorFlow, PyTorch and ONNX
- Model Zoo support: more than 70 pretrained models.
- Service ready: Long run with enhanced multi-thread support and memory control

DJL – our customers using ONNX(Runtime)

- Apache MXNet -> ONNX
  - Amazon Advertising: latency (avg) dropped from 22ms to 15ms for CTR model
  - E^-7 diff
- PyTorch -> ONNX
  - HyperFactors
- TensorFlow -> ONNX
  - Amazon Search: latency dropped from 12ms -> 4ms
- PaddlePaddle -> ONNX
  - Top bank in China: latency dropped from 1s to 400ms on OCR tasks

- Key challenge: No built-in operator support?

DJL – Hybrid Engine

- 1 master inference engine, 1 process engine
  - PyTorch/TensorFlow/MXNet + ONNXRuntime
- Efficient data passing
  - DirectBuffer allocation: avoid data-copying
- Resource Management: NDManager
  - Built-in resource management, stable in long run

DJL – Sample code

```
String modelUrl =
"https://mlrepo.djl.ai/model/tabular/softmax_regression/ai/djl/onnxruntime/iris_flowers/0.0.1/iris_flowers.zip";
Criteria<IrisFlower, Classifications> criteria = Criteria.builder()
        .setTypes(IrisFlower.class, Classifications.class)
        .optModelUrls(modelUrl)
        .optTranslator(new MyTranslator())
        .optEngine("OnnxRuntime") // use OnnxRuntime engine by default
        .build();
ZooModel<IrisFlower, Classifications> model = criteria.loadModel();
```

```
Predictor<IrisFlower, Classifications> predictor = model.newPredictor();
IrisFlower info = new IrisFlower(1.0f, 2.0f, 3.0f, 4.0f);
predictor.predict(info);
```

DJL – What's next with ONNXRuntime?

- ARM server support
- Android support