TensorFlow & Apache Flink™

An early look at a community project

Eron Wright

@eronwright

https://github.com/cookieai/flink-tensorflow

Background

Why TensorFlow?

- A powerful & flexible platform for machine intelligence
- Reusable machine learning models
- C++ core / Java language binding
- Ease of integration with Apache Flink

TF Scenarios

- Language Understanding
  - “syntaxnet”, Google Translate
- Image/Video/Audio Recognition
  - “Inception”
- Creative Arts
  - “Magenta”

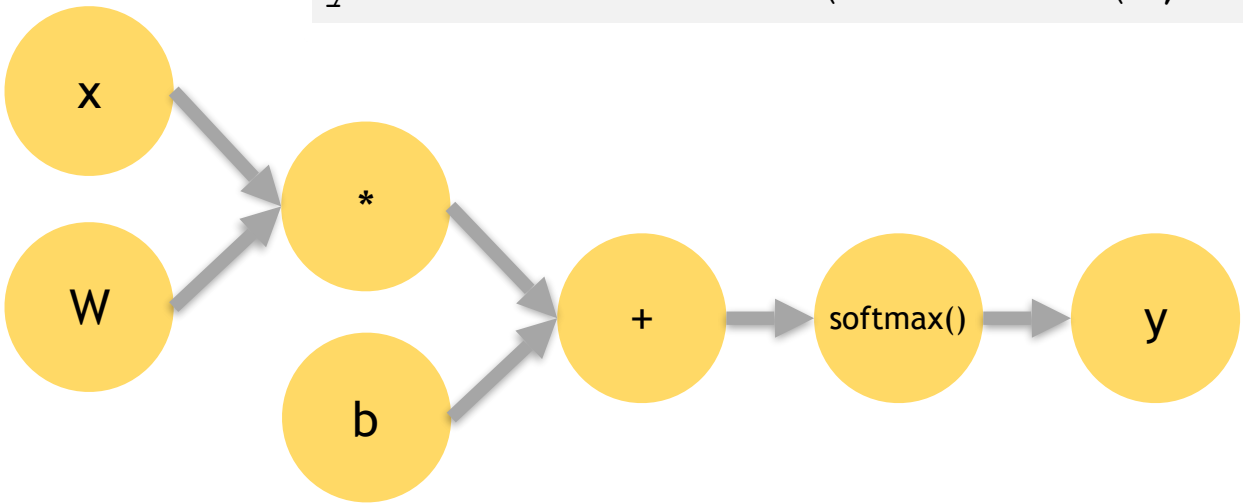
TF Models

- Portable using “Saved Model” format
  - Train on GPU-equipped cluster
  - Perform inference anywhere
- Well-defined interactions and data types using “signatures”
- Moving towards a Model Zoo

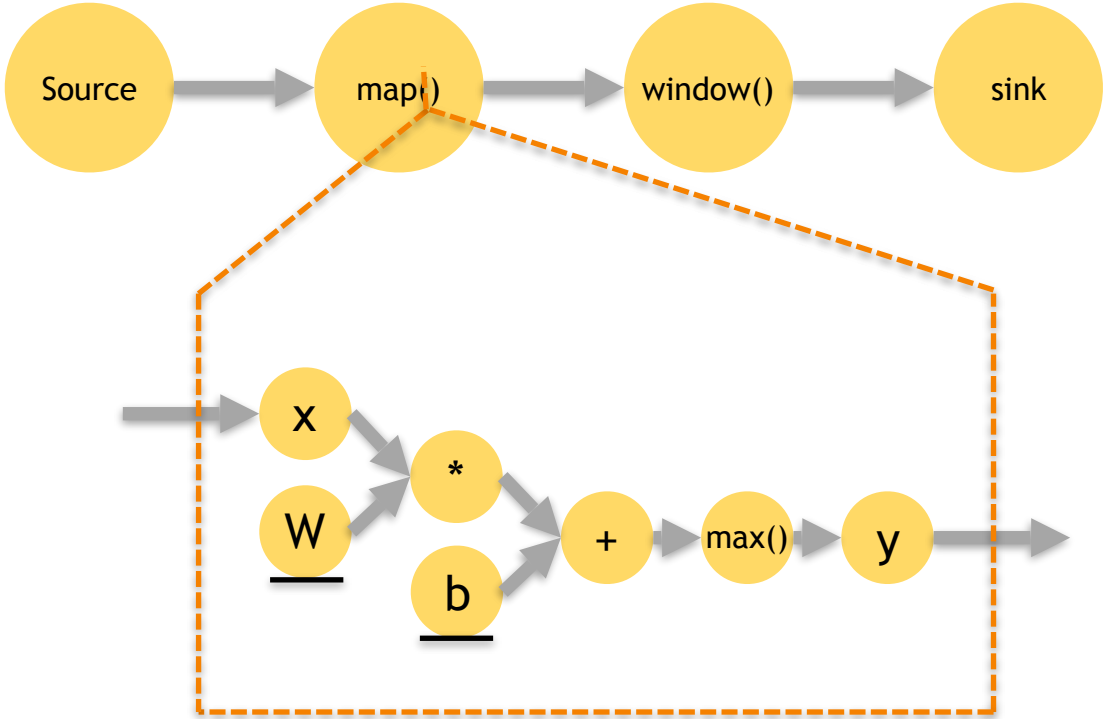
TF Graphs

$$y = \text{softmax}(Wx + b)$$

```
x = tf.placeholder(tf.float32, [None, 784])
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
y = tf.nn.softmax(tf.matmul(x, W) + b)
```



TF in Flink



Introducing Flink-Tensorflow

Project Status

- A prototype focused on inference using pre-trained TF models
- Scala-only (for now)
- A community effort

Basic Idea

- Use TF functionality in a Flink program; Not a TF compatibility layer
- “TF graph as a Flink map function”
- Support inference today, online learning in the future

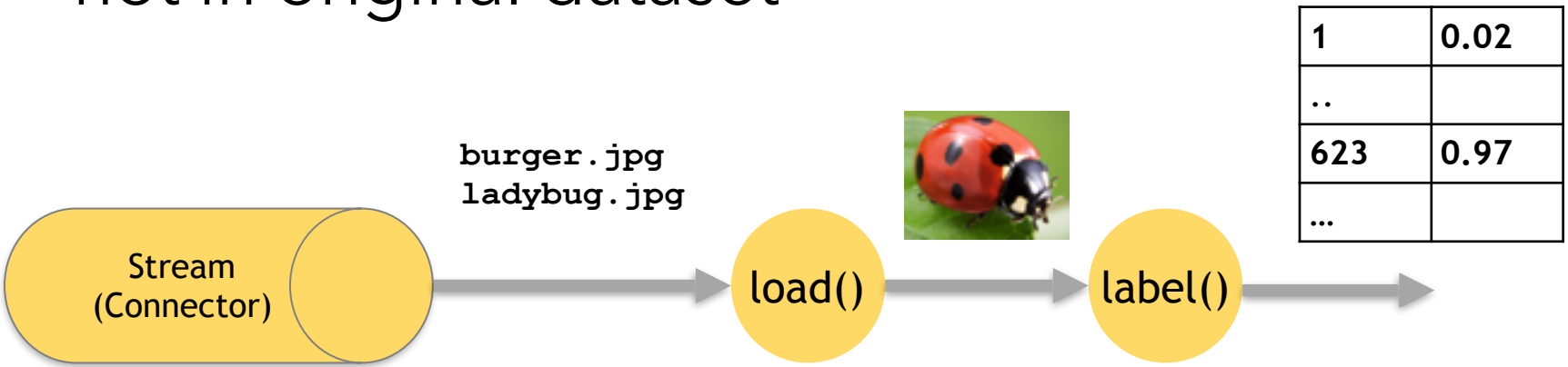
Demo

“Johnny”

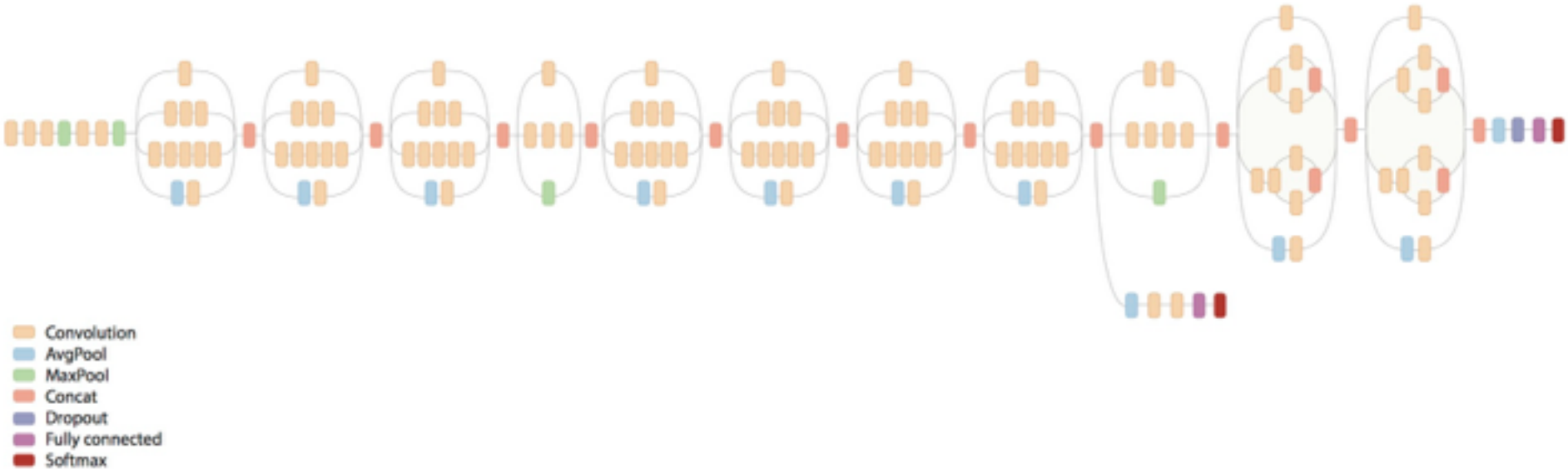
- A hypothetical security system based on picture passwords
  - Present three specific pictures within one minute: *Access Granted!*
  - On timeout: *Access Denied!*
- TF model for image labeling (“Inception”)
- Flink CEP library for sequence detection

Inception Model

- Pretrained with ImageNet dataset
- Supports “retraining” for learning new objects not in original dataset



Inception Model (Con't)



Inception v3

Using the Library

Basic Usage

1. Import a TensorFlow model
2. Write code to convert your domain objects to/from tensors
3. Use the model in a batch or streaming function

Importing a Model

1. Define the graph method(s) supported by the model (ref)
2. Specify how to load the model
    1. "saved model" loader, or
    2. graphdef loader, or
    3. ad-hoc graph builder

Importing a Model (Con't)

```
1  trait ClassificationMethod extends GraphMethod {
2    val name = "classify"
3    override type Input = Tensor[`2D`, Float]
4    override type Output = Tensor[`2D`, Float]
5  }
6
7  class MnistModel extends TensorFlowModel[MnistModel] {
8    /**
9      * Predicts labels for input images.
10     */
11   def predict = ModelFunction[ClassificationMethod](session, signatureDef)
12 }
```

Working with Tensors

- Tensors are off-heap, AutoCloseable multi-dimensional arrays
- You: convert input records to tensor
- Use Scala Automatic Resource Management (ARM)
  - Supports both imperative and monadic style

Writing a Flink Function

- Design goal: use TF in any transformation function
  - `MapFunction`
  - `ProcessFunction` (with event-time timers!)
  - `WindowFunction`
- Required: model lifecycle support
  - `ModelAwareFunction`

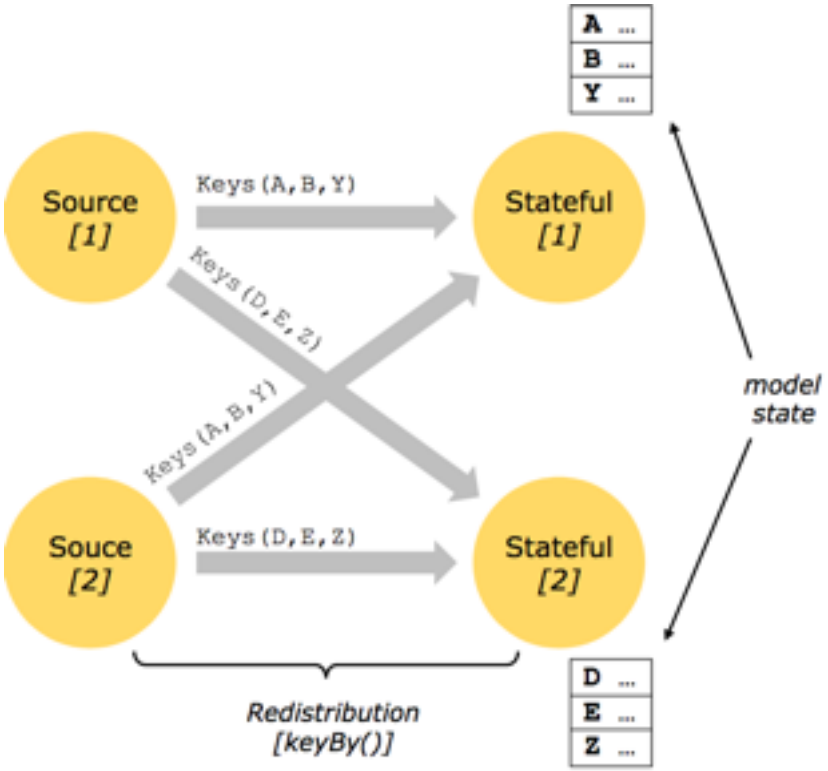
Runtime

- TF embedded within the Flink JobManager / TaskManager
- One model instance per sub-task
- Large unmanaged memory blocks
- No Python needed

Future Directions

Stateful Models

- Integrated checkpointing
- Support for keyed streams
- Emphasize online learning



Graph Builder (DSL)

- Construct TensorFlow graphs from scratch (ref)
- Code generation for TF operations
- Incorporate other libraries, high-level APIs (e.g. TF Keras)

Other

- Instrumentation
  - TF Summaries (for TensorBoard)
  - Flink Metrics
- Model versioning
  - Leverage Flink job versioning methods
  - Treat models as side-input

TensorFlow & Apache Flink™

An early look at a community project

Eron Wright

@eronwright

https://github.com/cookieai/flink-tensorflow

Thanks!