Announcements · TFX ·

Introducing TensorFlow Model Analysis: Scaleable, Sliced, and Full-Pass Metrics

March 30, 2018

Posted by Clemens Mewald, Product Manager for TFX

Today we've launched TensorFlow Model Analysis (TFMA), an open-source library that combines the power of TensorFlow and Apache Beam to compute and visualize evaluation metrics. Before deploying any machine learning (ML) model, ML developers need to evaluate it to ensure that it meets specific quality thresholds and behaves as expected for all relevant slices of data. Additionally, this computation should seamlessly scale from a small dataset that fits into memory to large, distributed computation. In this post we will give an overview of TFMA and how developers can use it to address all of the aforementioned challenges.

Not all evaluation metrics are created equal...

Before we jump into how TFMA works, we will first look into how it differs from the evaluation metrics that can already be computed and observed in TensorBoard.

During training vs after training

TFMA exports a SavedModel containing the eval graph and additional metadata to compute metrics, which means it computes metrics once using the exported model. It is important to evaluate performance on the final model because that's the model that will be deployed.

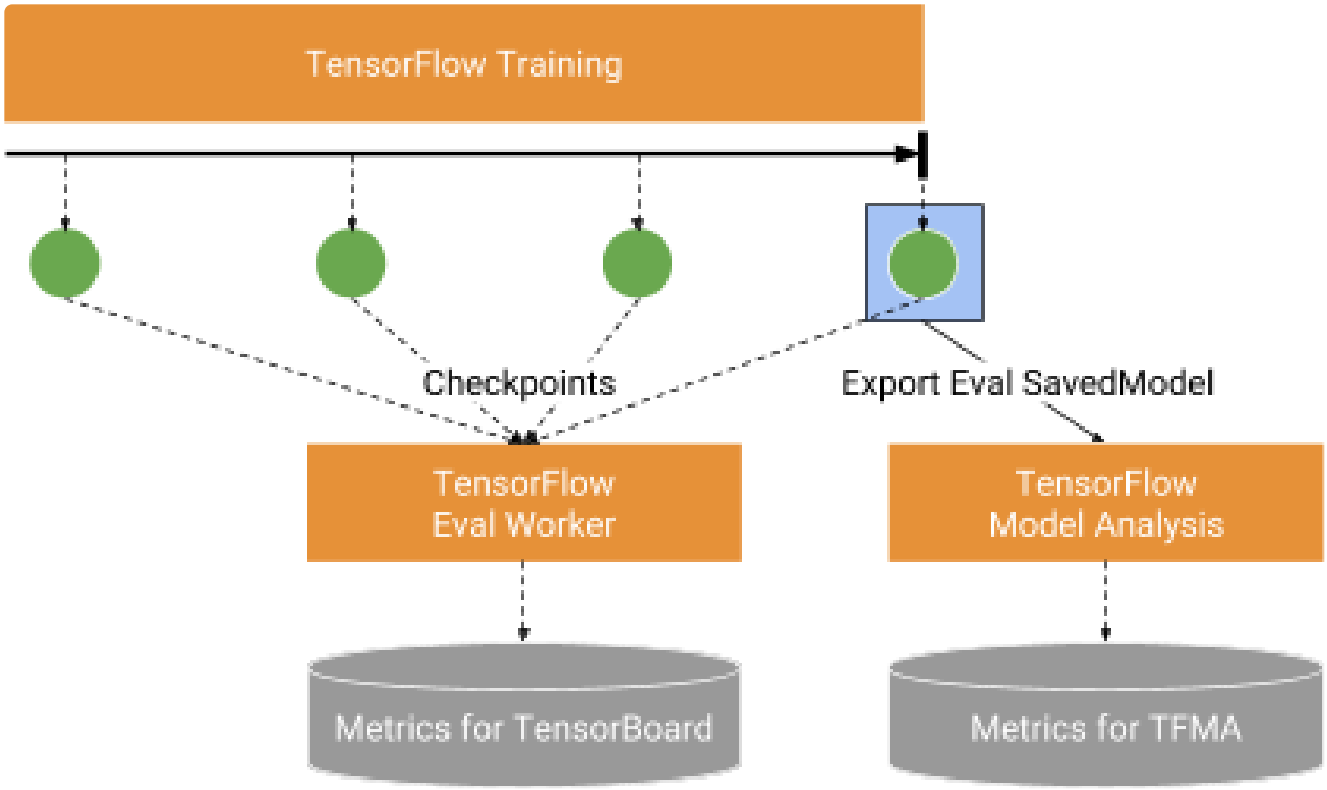

Figure 1: TensorBoard visualizes streaming metrics that are computed from checkpoints. TFMA computes and visualizes metrics using an exported SavedModel.

One model vs multiple models over time

TensorBoard is commonly used to inspect the training progress of a single model. It can also be used to visualize metrics for more than one model, with performance for each plotted against their global training steps as they are training.

TFMA also allows developers to visualize model metrics over time in a time series graph. The difference between TensorBoard and TFMA lies within the horizontal axis. TensorBoard visualizes streaming metrics of multiple models over global training steps, whereas TFMA

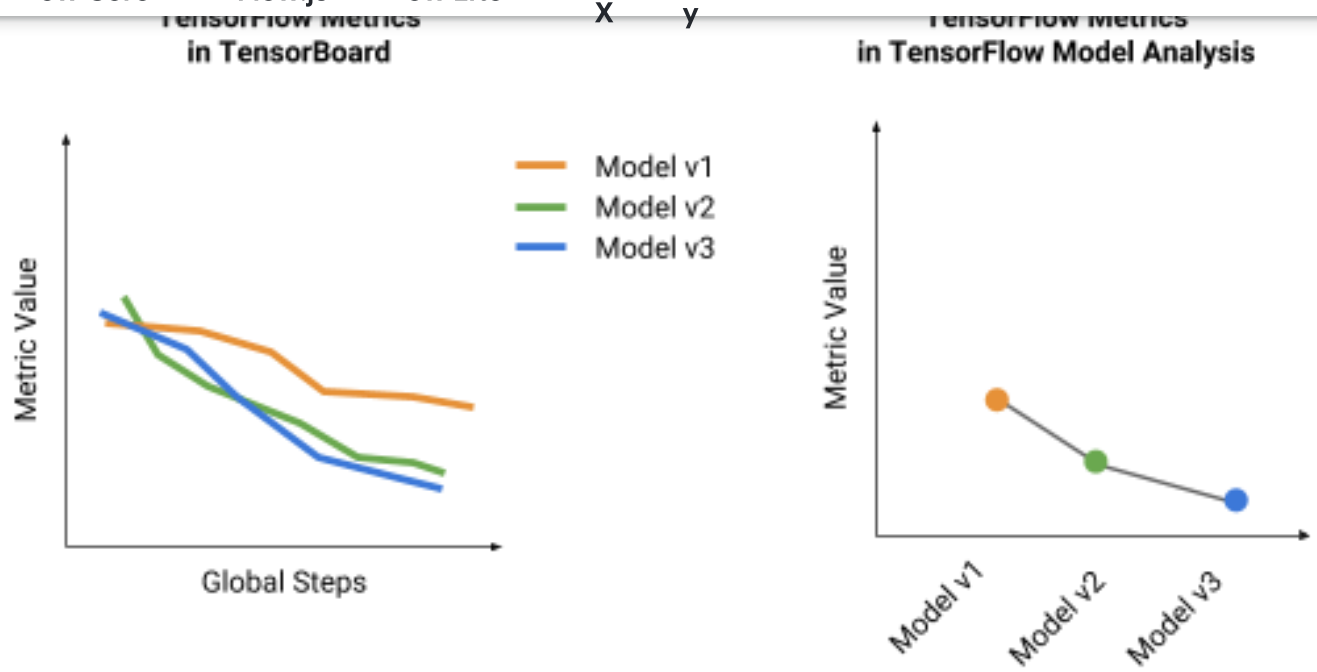

Figure 2: TensorBoard can overlay streaming metrics of multiple models over global training steps. TFMA only shows metrics computed based on the exported SavedModel, but can visualize those as a time series across multiple model versions.

Aggregate vs sliced metrics

Most model evaluation results look at aggregate metrics. A model may have an acceptable AUC over the entire eval dataset, but underperform on specific slices. In general, a model with good performance “on average” may exhibit failure modes that are not apparent by looking at an aggregate metric.

Slicing metrics allows us to analyze the performance of a model on a more granular level. This functionality enables developers to identify slices where examples may be mislabeled, or where the model over- or under-predicts. For example, TFMA could be used to analyze whether a model that predicts the generosity of a taxi tip works equally well for riders that take the taxi during day hours vs night hours (if sliced by the feature hour).

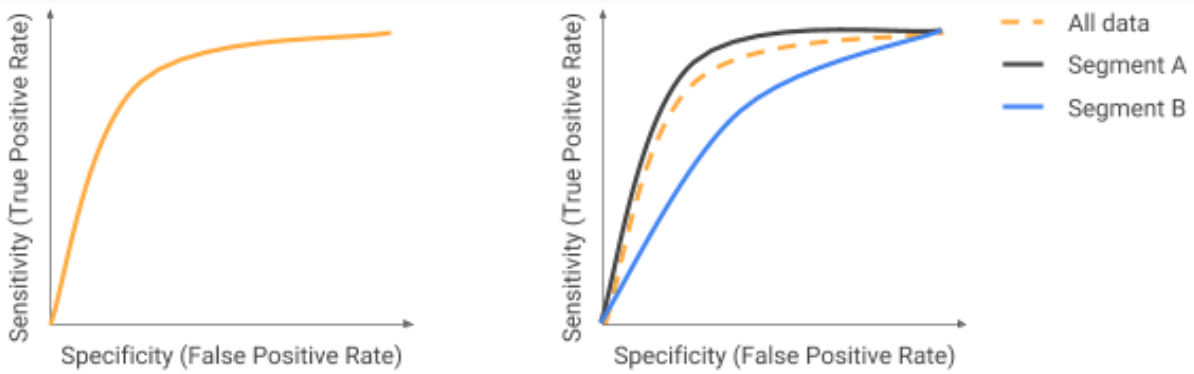

Figure 3: TFMA allows us to slice a metric by different segments of our eval dataset, enabling more fine grained analysis of a model and how it performs on different slices.

Streaming vs full-pass metrics

TensorFlow metrics that are visualized in TensorBoard are commonly computed on a mini-batch basis during training. They are called “streaming metrics” and are approximations based on those observed mini-batches.

TFMA uses Apache Beam to do a *full pass over the specified evaluation dataset*. This not only allows more accurate calculation of metrics, but also scales up to massive evaluation datasets, since Beam pipelines can be run using distributed processing back-ends.

Note that TFMA computes the same TensorFlow metrics that are computed by the TensorFlow eval worker, just more accurately by doing a full pass over the specified dataset. TFMA can also be configured to compute additional metrics that were not defined in the model.

Furthermore, if evaluation datasets are sliced to compute metrics for specific segments, each of those segments may only contain a small number of examples. To compute accurate metrics, a deterministic full pass over those examples is important.

How does TensorFlow Model Analysis work?

trainer into its own SavedModel. TFMA uses the graph in this SavedModel to compute (sliced) metrics and provides visualization tools to analyze those metrics.

Exporting the eval graph

If developers use TensorFlow Estimators, they are already familiar with exporting a SavedModel for TensorFlow Serving, using the export_savedmodel method. TFMA provides an analogous export_eval_savedmodel method that exports a SavedModel containing the eval metrics and all additional information needed to compute them. The eval_input_fn may also contain analysis-only features, i.e. features that were not trained on but that are used for slicing the eval metrics.

```
# use TFMA to export an eval graph from the TensorFlow Estimator
tfma.export.export_eval_savedmodel(estimator=estimator,
                                   eval_input_receiver_fn=eval_input_fn, …)
```

Computing (sliced) metrics

Once the evaluation SavedModel has been exported, developers can either immediately run TFMA, which will compute the metrics that have been configured in the Estimator, or additionally configure slices they want to have computed. The code example below shows how to configure a feature for slicing.

TFMA uses Apache Beam to evaluate the model using the entire dataset. We use the Beam SDK to define a data-parallel processing pipeline which can be run using several distributed processing back-ends (or runners). On a local machine (or in a notebook), a DirectRunner can be used to run this pipeline locally. One of the major benefits of using the Beam SDK is that the same data processing pipeline can be scaled up on different backends, e.g. in the cloud with the DataflowRunner.

```
# specify slicing spec
slice_spec = [slicer.SingleSliceSpec(columns=['column_name'])]
# configure Beam pipeline
with beam.Pipeline(…) as pipeline:
  raw_data = pipeline | 'ReadData' >> beam.io.Read(…)
  # run TFMA Beam job
  _ = raw_data | 'EvaluateAndWriteResults' >> tfma.EvaluateAndWrit
                                  slice_spec,
                                  result_path)
```

Visualize metrics in Jupyter

```
# load evaluation results
result = tfma.load_eval_result(result_path)
# render results
tfma.viewer.render_slicing_metrics(result)
```

The render_slicing_metrics call loads the slicing browser component in the Jupyter notebook (see Figure 4) and allows developers to inspect the results of the TFMA eval run.

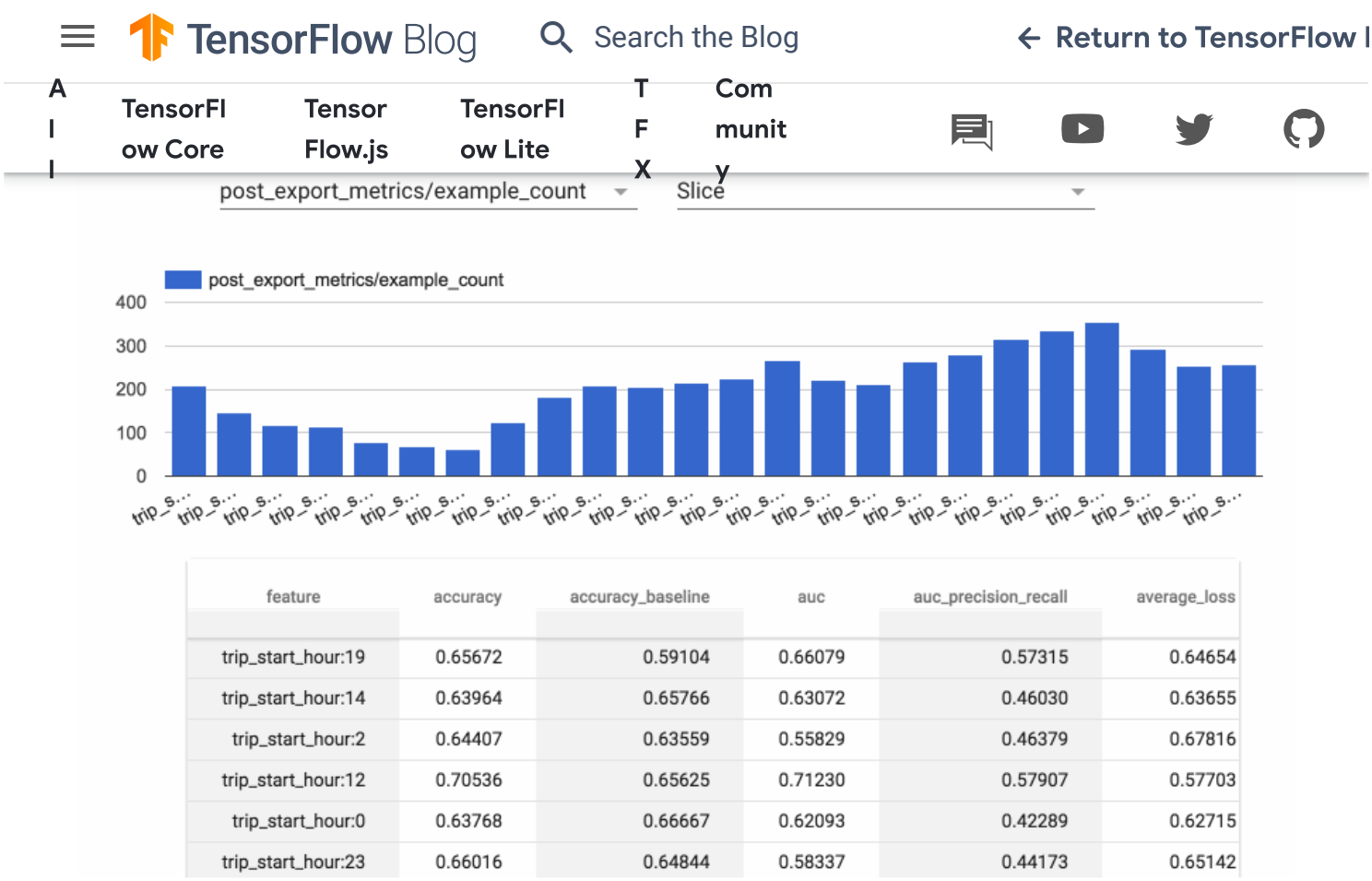

Figure 4: Slicing browser component loaded in a Jupyter notebook. It allows developers to display a histogram of metrics sliced by the feature value and to identify underperforming slices.

The animation in Figure 4 demonstrates a workflow of how to identify a slice with the lowest AUC by first selecting “Metrics Histogram” for visualization, filtering to show only slices with 100 or more examples, selecting the metric “auc”, sorting the table by “auc” and selecting the row with the lowest AUC. In this example it is the slice with the feature value 2 for trip_start_hour (using the model and data from our end-to-end example). Once an interesting slice has been identified, developers can investigate whether this is expected or needs to be mitigated.

Try out TensorFlow Model Analysis today!

We’ve open-sourced TFMA and published it on GitHub at github.com/tensorflow/model-analysis under the Apache 2.0 License. This release includes everything needed for exporting an evaluation SavedModel, computing sliced metrics using Apache Beam, and visualizing them in a Jupyter notebook.

Serving together.

To help developers get started, we suggest reading and trying out the end-to-end example on GitHub.

What's next for TensorFlow Model Analysis

As mentioned in the introduction, developers can use TFMA for evaluating model quality against thresholds, analyzing model behavior on different slices of data, and identifying failure modes that need to be addressed either via data cleaning or improved modeling.

With this release we make TFMA available to the TensorFlow community to analyze TensorFlow models and improve the quality of models that are deployed. We will soon add the capability of computing metrics that cannot be computed in a TensorFlow graph, e.g. ranking metrics.

In addition to being able to run TFMA locally (with the DirectRunner) and on Google Cloud (with the DataflowRunner), the Apache Flink and Apache Beam communities are nearing completion of a Flink Runner. Follow the corresponding JIRA ticket, Apache Beam blog, or mailing lists to get notifications about availability of the Flink Runner. The community has also started work on the Spark Runner which will be available in a few months.

At Google, we use Tensorflow Model Analysis as part of a larger ML platform called TensorFlow Extended (TFX). In continuously running TFX pipelines we use it for automated validation of TensorFlow models and interactive model analysis workflows (e.g. using the slicing browser component shown in Figure 4.) As announced at the TensorFlow Dev Summit, we will be open sourcing more of TFX in the future. Stay tuned to the TensorFlow Blog for an upcoming TFX post, and keep in touch by following TensorFlow on Twitter.

Next post

Announcements · TFX ·

Highlights from the TensorFlow Developer Summit, 2018

March 30, 2018 — Posted by Sandeep Gupta, Product Manager for TensorFlow, on behalf of the TensorFlow team

Today, we're holding the second TensorFlow…