# TensorFlow


monteggia probability:
0.966248631477356
galeazzi probability:
0.03375125303864479
webcam captures
model predicts
info is 1D
Thinking...


monteggia probability:
0.966248631477356
galeazzi probability:
0.03375125303864479
webcam captures
model predicts
info is 1D
Is that a Monteggia fracture ?


Monteggia Fractures - Trauma - Ort...
orthobullets.com
Monteggia fracture
en.wikipedia.org
Monteggia fracture-dislocation - ty...
L

| | | | |
|---|---|---|---|
| 76,000,000 | 80,000+ | 13,000+ | 2,400+ |
| downloads | commits | pull requests | contributors |

TensorFlow
2.2

Emphasis on performance

Compatibility with the rest of the TensorFlow ecosystem

Stability in the core library

# TensorFlow
## Ecosystem

From research

To production

Deployed everywhere

Empowering Responsible AI

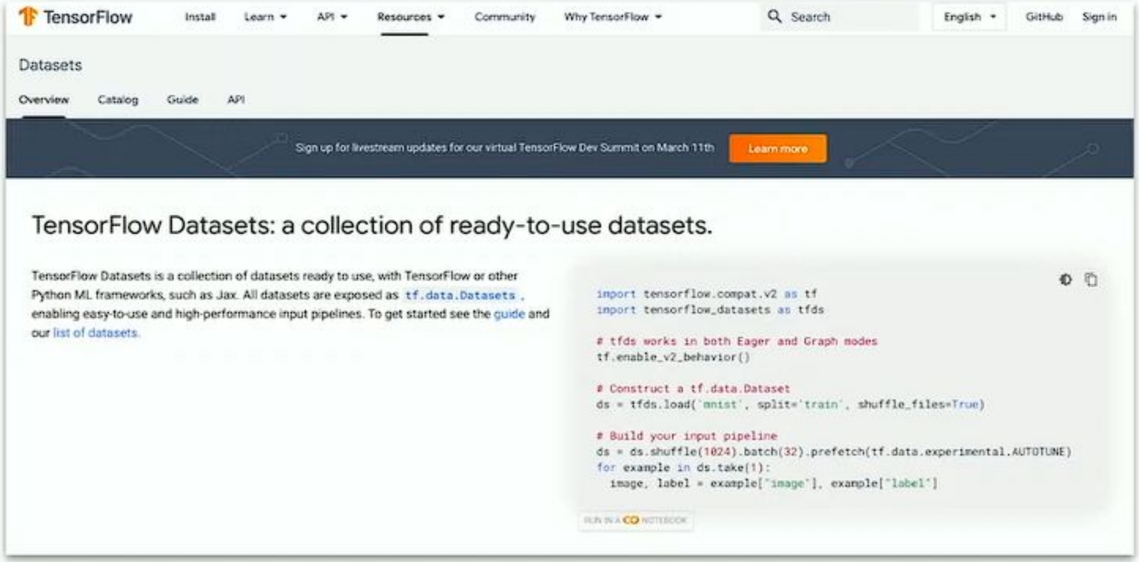
Powered by the community

# Eager at the core and simple, performant data input pipelines in 2.x

eager execution, supporting numpy arrays

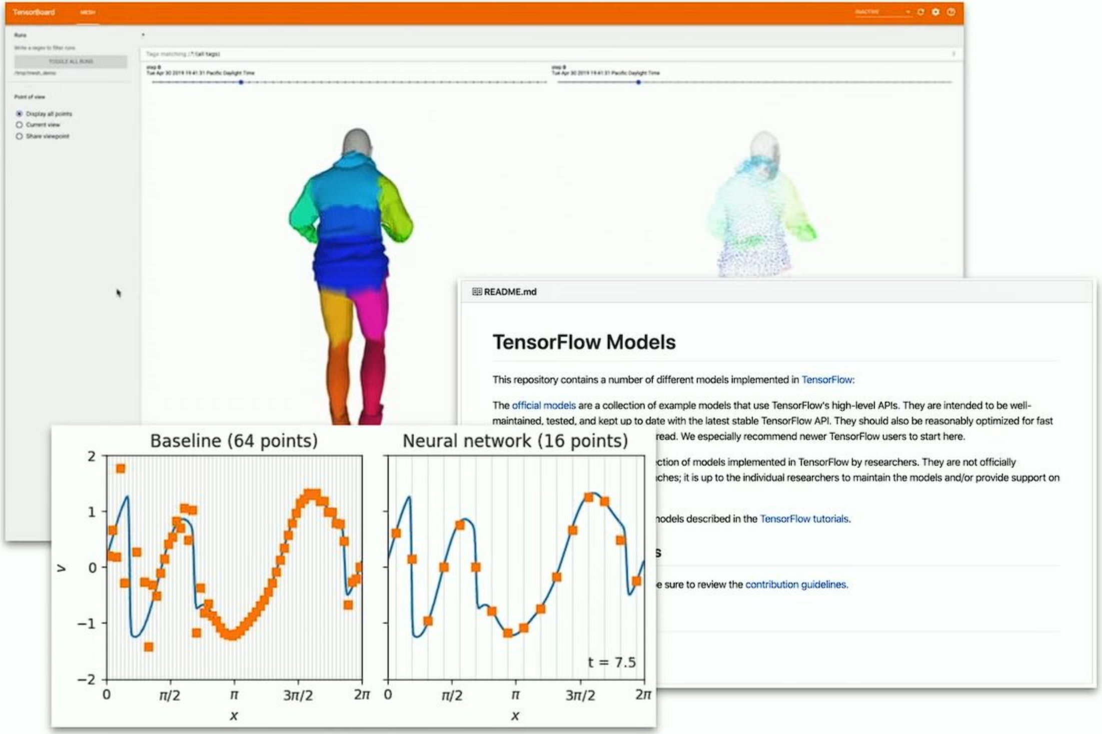
tf.data

TensorFlow Datasets



# Add-ons and extensions to the TensorFlow ecosystem

- TF Probability
- TF Graphics
- Mesh TensorFlow
- TF Model Garden
- TF Agents
- TF Text
- Swift for TensorFlow
- Sonnet
- JAX
- Neural Structured Learning
- TF Quantum
- ...and more on tensorflow.org!

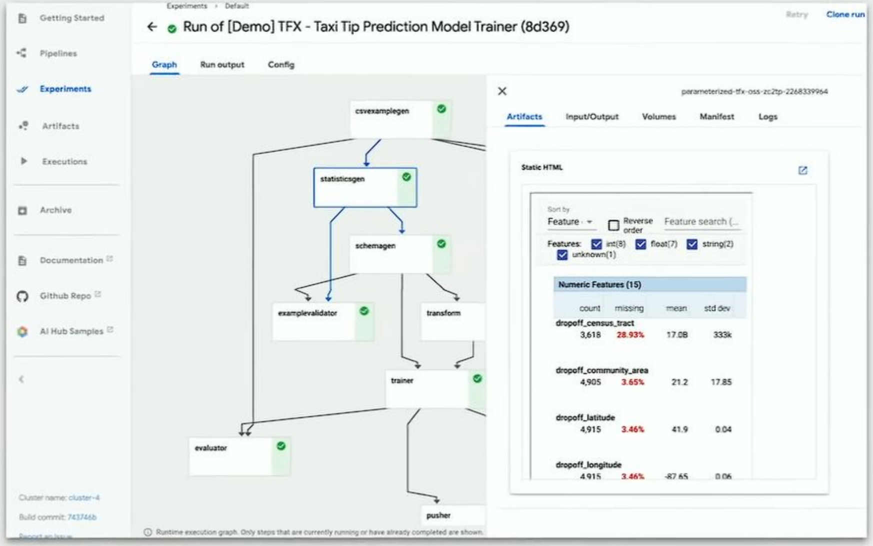


TensorFlow Extended

Kubeflow

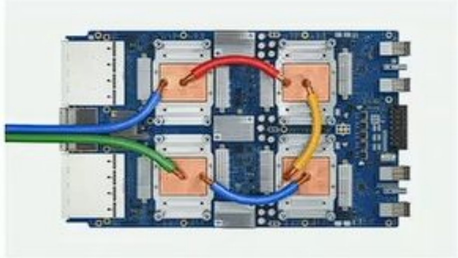
Google Cloud AI Platform Pipelines

TensorFlow
Enterprise



# TensorFlow 2.1 supports Cloud TPUs

+

# Challenge

arXiv:1810.04805v2 [cs.CL] 24 May 2019

**BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
{jacobdevlin,mingweichang,kentonl,kristout}@google.com

**Abstract**

We introduce a new language representation model called **BERT**, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. Unlike recent language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-the-art results on eleven natural language processing tasks, including pushing the GLUE score to 80.5% (7.7% point absolute improvement), MultiNLI accuracy to 86.7% (4.6% absolute improvement), SQuAD v1.1 question answering Test F1 to 93.2 (1.5 point absolute improvement) and SQuAD v2.0 Test F1 to 83.1 (5.1 point absolute improvement).

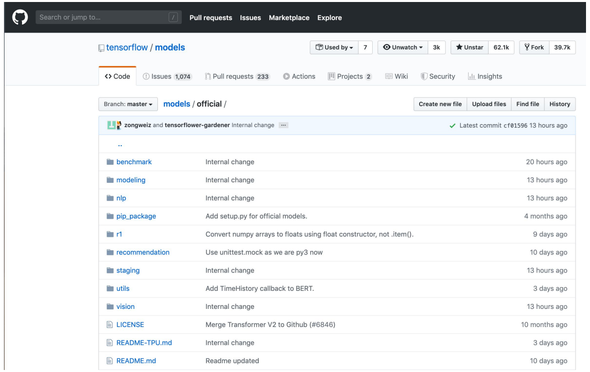
**1 Introduction**

Language model pre-training has been shown to be effective for improving many natural language processing tasks (Dai and Le, 2015; Peters et al., 2018a; Radford et al., 2018; Howard and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Dolan and Brockett, 2005), which aim to predict the relationships between sentences by analyzing them holistically, as well as token-level tasks such as named entity recognition and question answering, where models are required to produce fine-grained output at the token level (Tjong Kim Sang and De Meulder, 2003; Rajpurkar et al., 2016).

There are two existing strategies for applying pre-trained language representations to downstream tasks: *feature-based* and *fine-tuning*. The feature-based approach, such as ELMo (Peters et al., 2018a), uses task-specific architectures that include the pre-trained representations as additional features. The fine-tuning approach, such as the Generative Pre-trained Transformer (OpenAI GPT) (Radford et al., 2018), introduces minimal task-specific parameters, and is trained on the downstream tasks by simply fine-tuning *all* pre-trained parameters. The two approaches share the same objective function during pre-training, where they use unidirectional language models to learn general language representations.

We argue that current techniques restrict the power of the pre-trained representations, especially for the fine-tuning approaches. The major limitation is that standard language models are unidirectional, and this limits the choice of architectures that can be used during pre-training. For example, in OpenAI GPT, the authors use a left-to-right architecture, where every token can only attend to previous tokens in the self-attention layers of the Transformer (Vaswani et al., 2017). Such restrictions are sub-optimal for sentence-level tasks, and could be very harmful when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions.

In this paper, we improve the fine-tuning based approaches by proposing BERT: **B**idirectional **E**ncoder **R**epresentations from **T**ransformers. BERT alleviates the previously mentioned unidirectionality constraint by using a "masked language model" (MLM) pre-training objective, inspired by the Cloze task (Taylor, 1953). The masked language model randomly masks some of the tokens from the input, and the objective is to predict the original vocabulary id of the masked




**How do I use it?**

**Is it safe?**

**Is it fair?**

**Is it the latest version?**

tfhub.dev

# TensorFlow Hub

A comprehensive collection of models

Image

Text

Video

Audio

# Ready to use

Pre-trained models ready for transfer learning on your own datasets and deployable anywhere you want

TensorFlow Extended

TensorFlow .JS

TensorFlow Lite

Coral

TensorFlow Hub

Search

Send feedback

## ← imagenet/mobilenet_v2_050_96/feature_vector

Problem domain

Image feature vector

Architecture

MobileNet V2

Publisher

Google

Dataset

ImageNet (ILSVRC-2012-CLS)

Format: TF2.0 Saved Model

Fine tunable: Yes

License: Apache-2.0

Last updated: 2020-02-20

# Model formats

Saved Model | .JS (v1, default) | .JS (v2, default) | .JS (v3, default)

### Want to use this model?

To use this model, take a look at the example code, or at our user guide.

You can also try out the associated Colab.

Copy URL to clipboard

Download Model

Open Colab Notebook

Asset size: 2.62MB

# TF2 SavedModel

This is a SavedModel in TensorFlow 2 format. Using it requires TensorFlow 2 (or 1.15) and TensorFlow Hub 0.5.0 or newer.

# Overview

MobileNet V2 is a family of neural network architectures for efficient on-device image classification and related tasks, originally published by

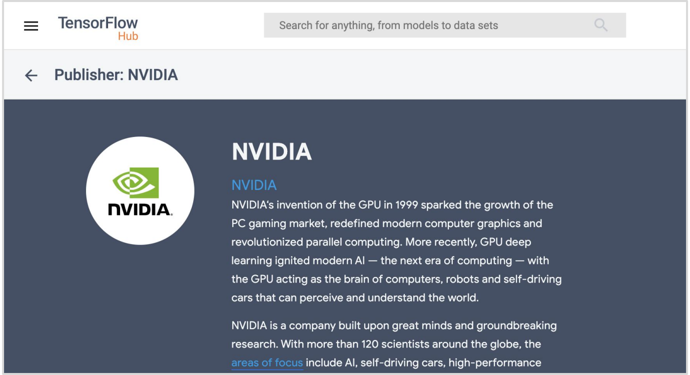
- Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chieh Chen: "Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation", 2018.

Mobilenets come in various sizes controlled by a multiplier for the depth (number of features) in the convolutional layers. They can also be trained for various sizes of input images to control inference speed.

# Powered by the community

Built, trained, and deployed already by the TensorFlow community

- DeepMind
- Google
- Microsoft AI for Earth
- NVIDIA
- The Metropolitan Museum of Art
- Global Biodiversity Information Facility
- Kaggle
- And more...

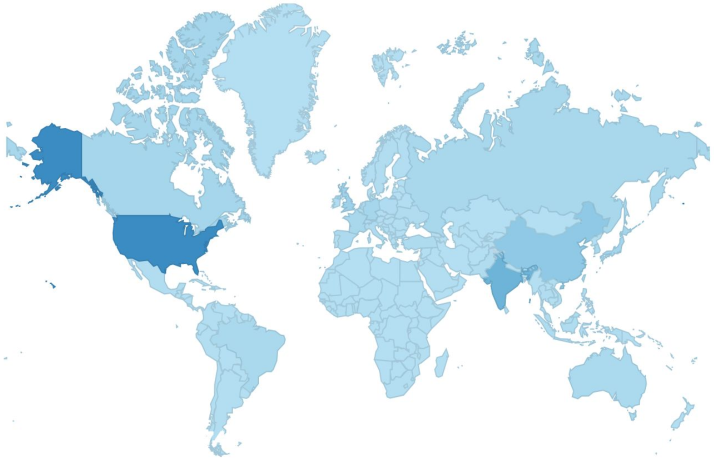


Community

# Kaggle competition now supports 2.x

- No setup
- No provisioning
- Datasets in optimal formats
- TPUs and GPUs provided at no cost to users

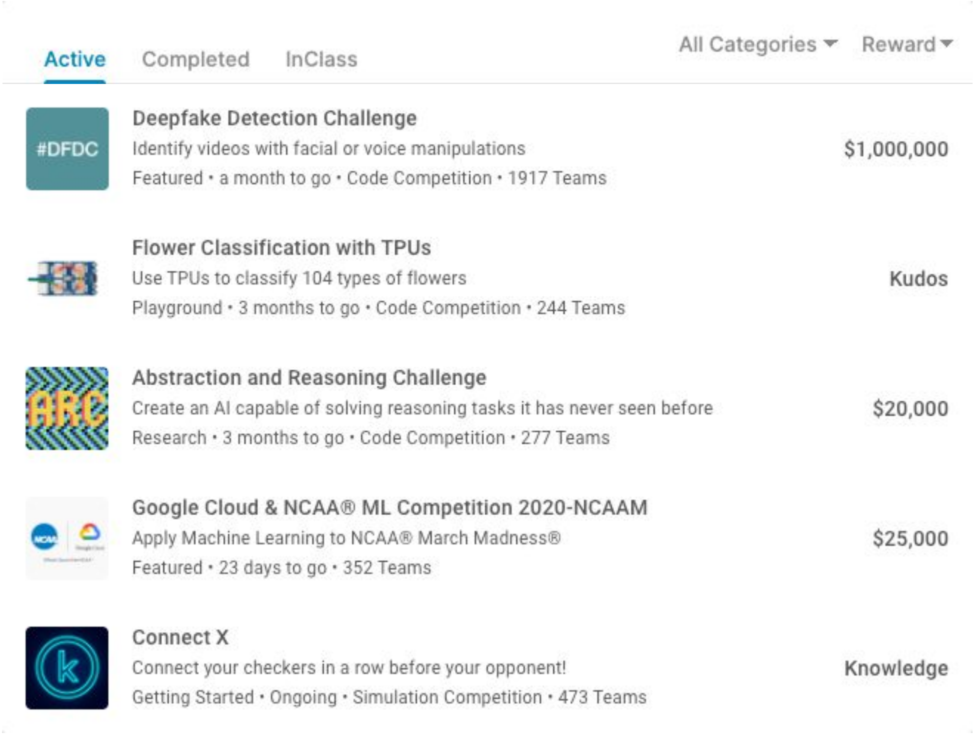
# What is Kaggle?

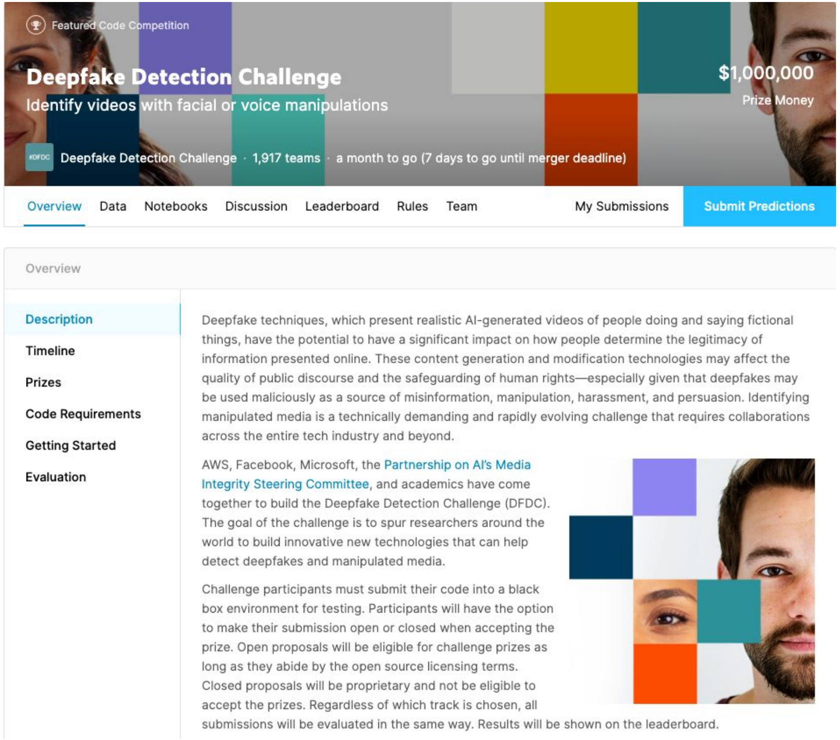


- Community
- Competitions
- Datasets
- Notebooks
- Courses

> 4 million registered users

# Kaggle hosts machine learning competitions



# TF 2.1 + Kaggle == easy acceleration

- No setup
- No provisioning
- Datasets in optimal formats
- TPUs and GPUs provided at no cost to users


iris [NO→petunia]
rose [NO→hibiscus]
common dandelion [NO→blanket flower]
wallflower [OK]
petunia [NO→hibiscus]
spear thistle [OK]
morning glory [NO→watercress]
thorn apple [OK]
iris [OK]
daisy [NO→orange dahlia]
rose [NO→geranium]
iris [NO→fritillary]
daisy [OK]
common dandelion [NO→tree poppy]
daisy [OK]
yellow iris [NO→king protea]
iris [NO→yellow iris]
rose [NO→camellia]
iris [NO→wild rose]
wild geranium [OK]

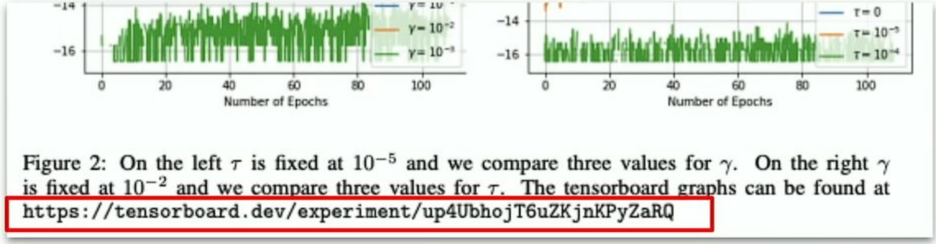
# Check out Tensorflow on Kaggle!

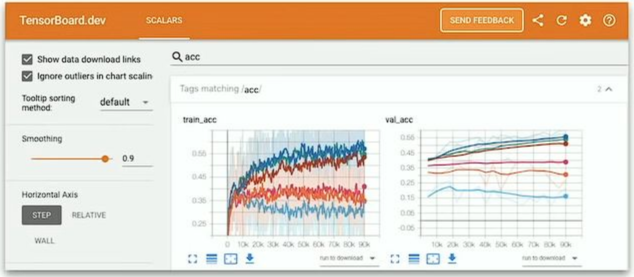
- Check out the code above
  - https://www.kaggle.com/philculliton/a-simple-tf-2-1-notebook
- Compete in the Flower Classification competition
  - https://www.kaggle.com/c/flower-classification-with-tpus
- Try TPUs with TF 2.1 in Notebooks
  - https://www.kaggle.com/notebooks

# Better experimentation with TensorFlow

## TensorBoard.dev

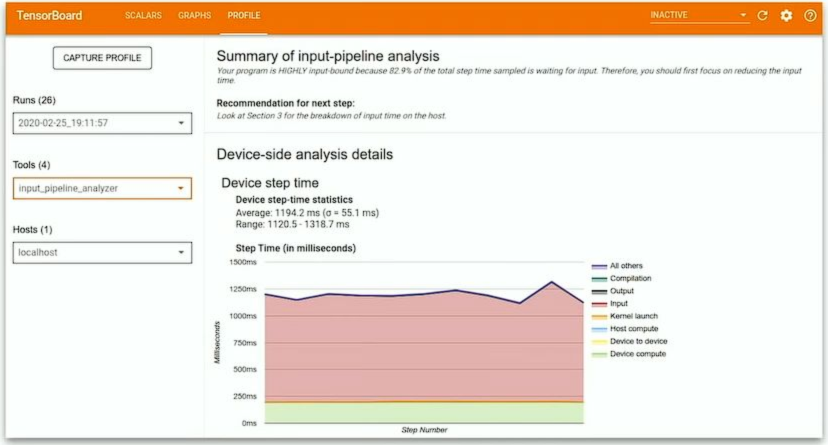
Upload and share your ML experiments with anyone



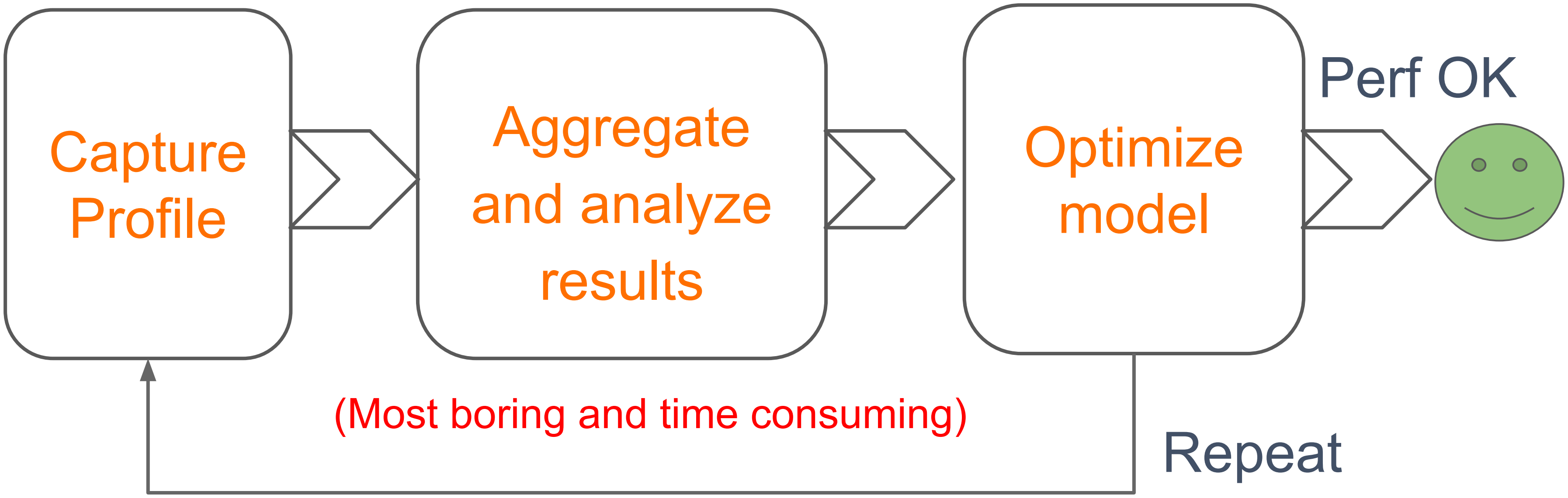

## Performance Profiler

Available in TensorBoard, Profiler provides overview of model performance and better debugging guidance

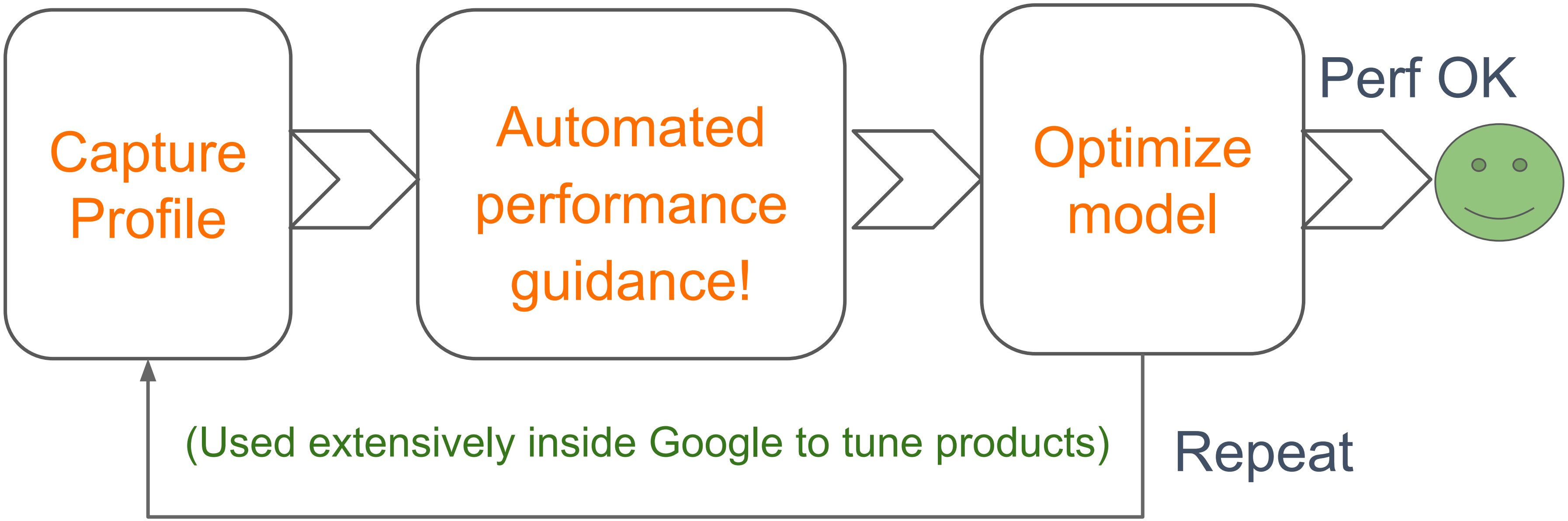


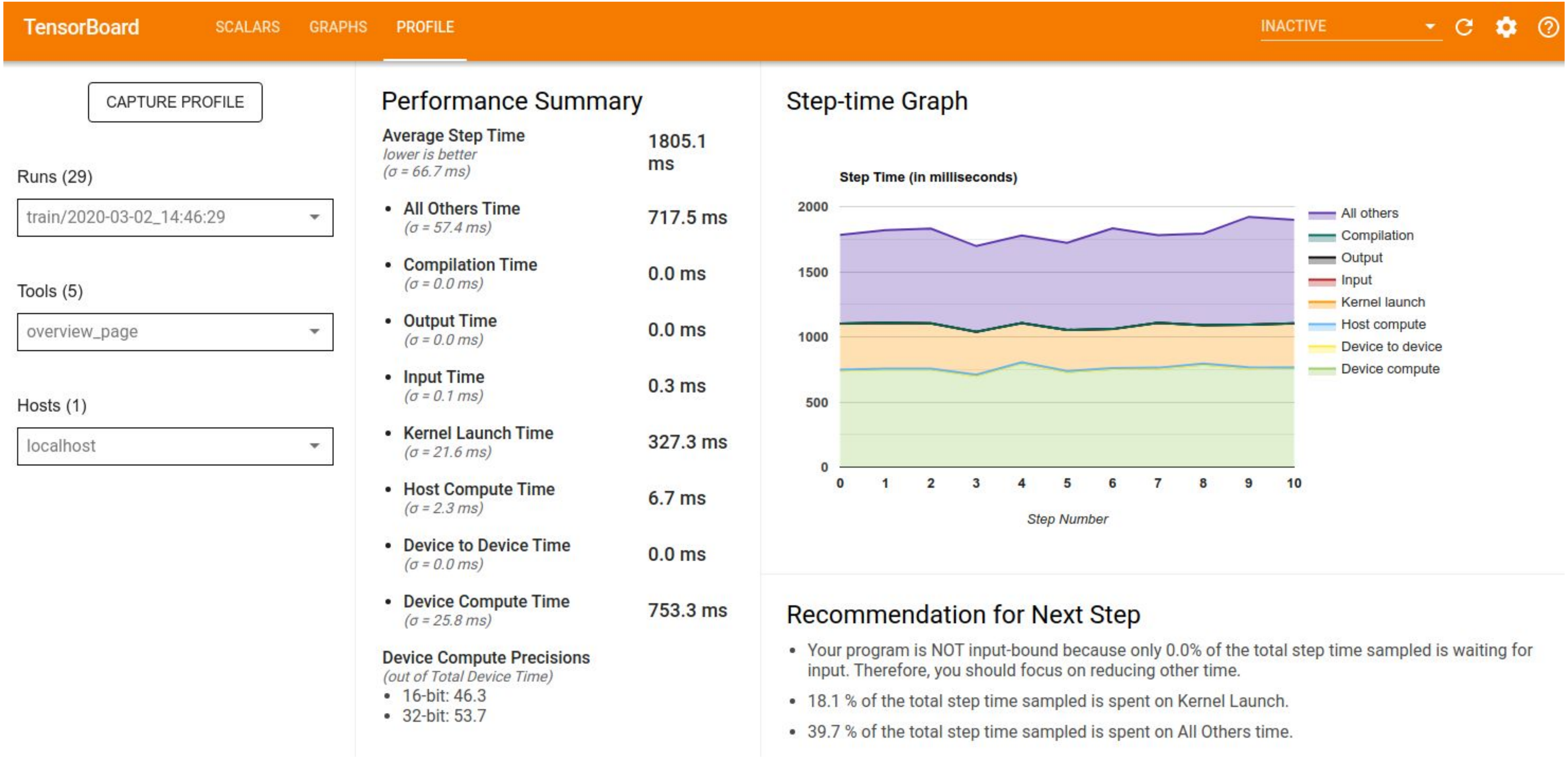

# TF is speed!

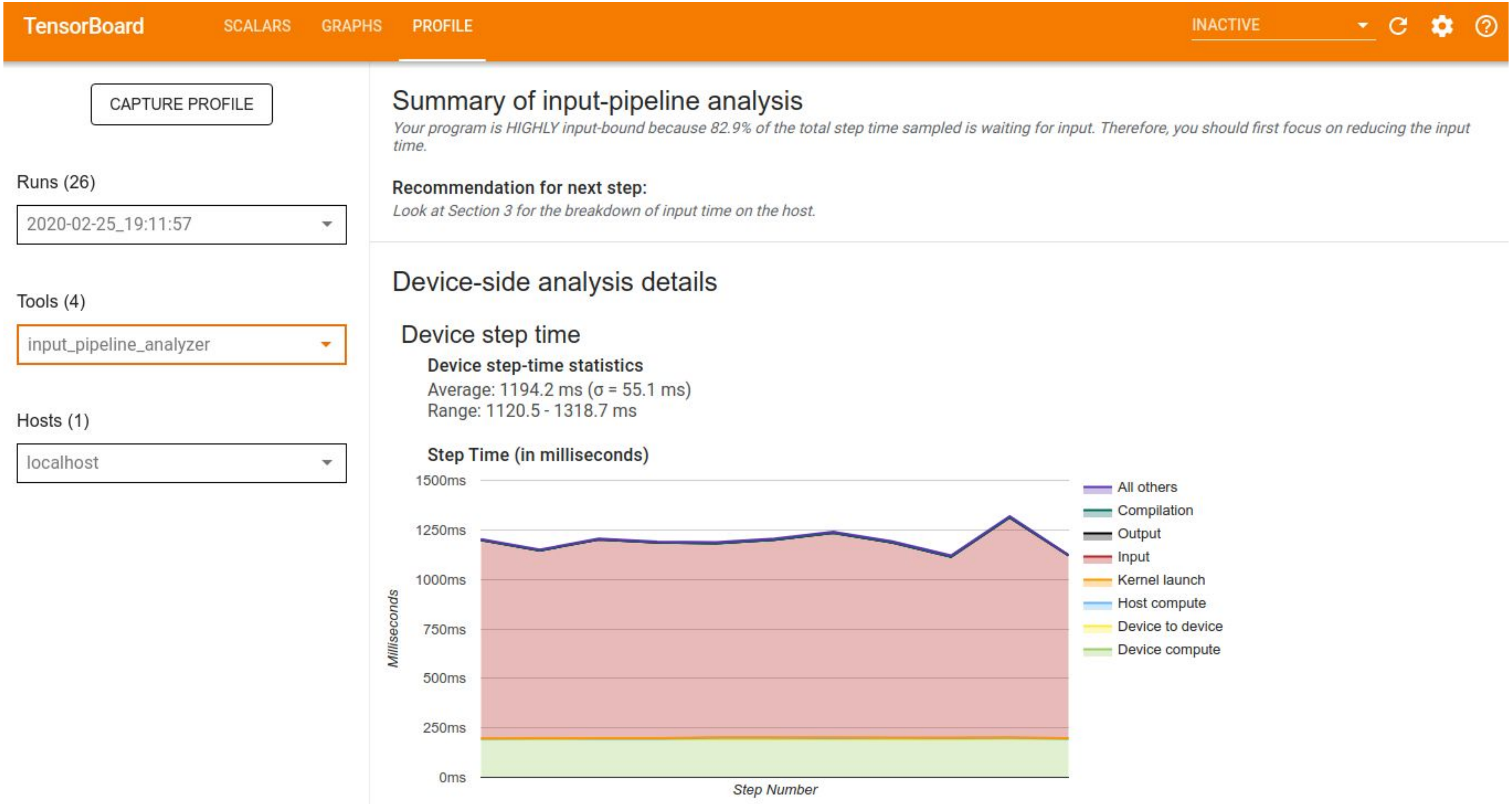
# Life of a performance engineer

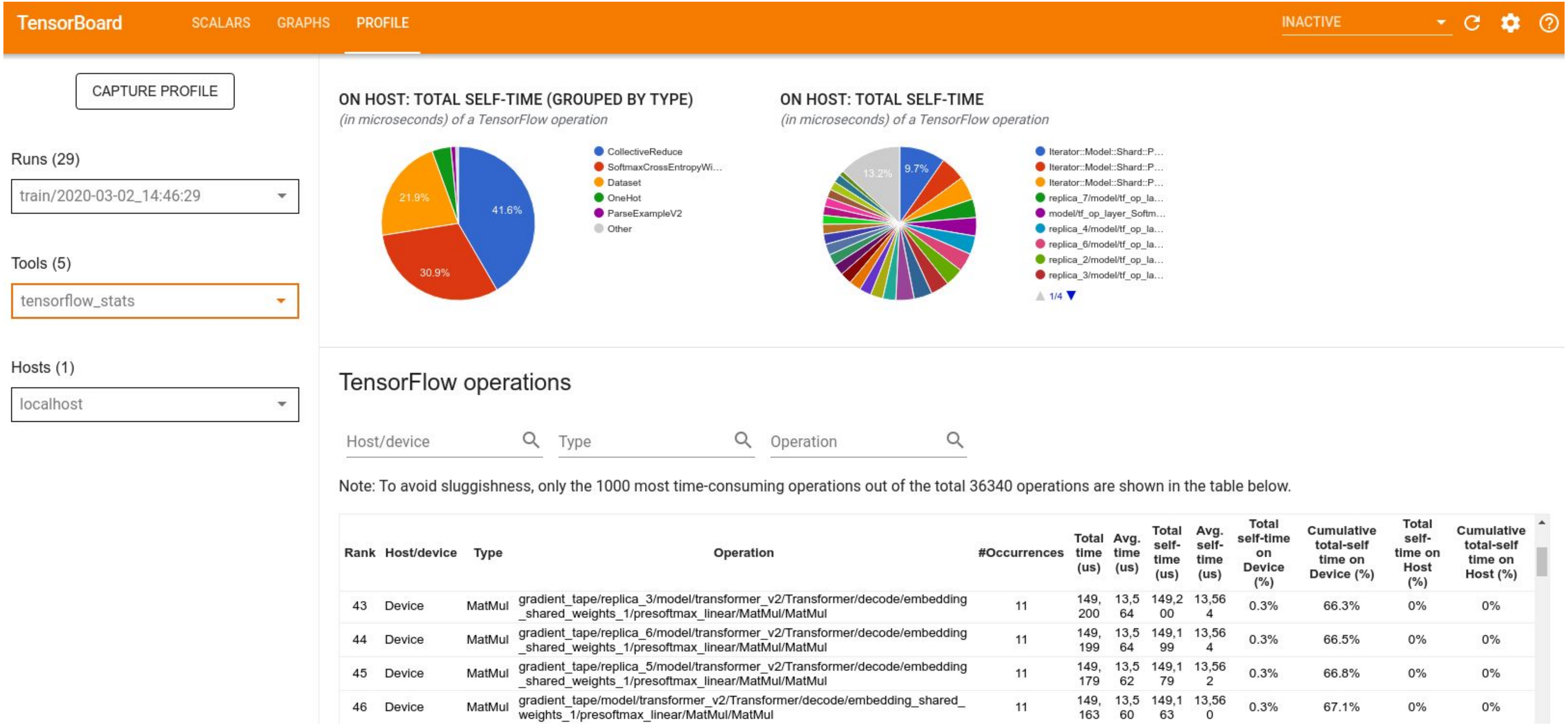


# ML productivity

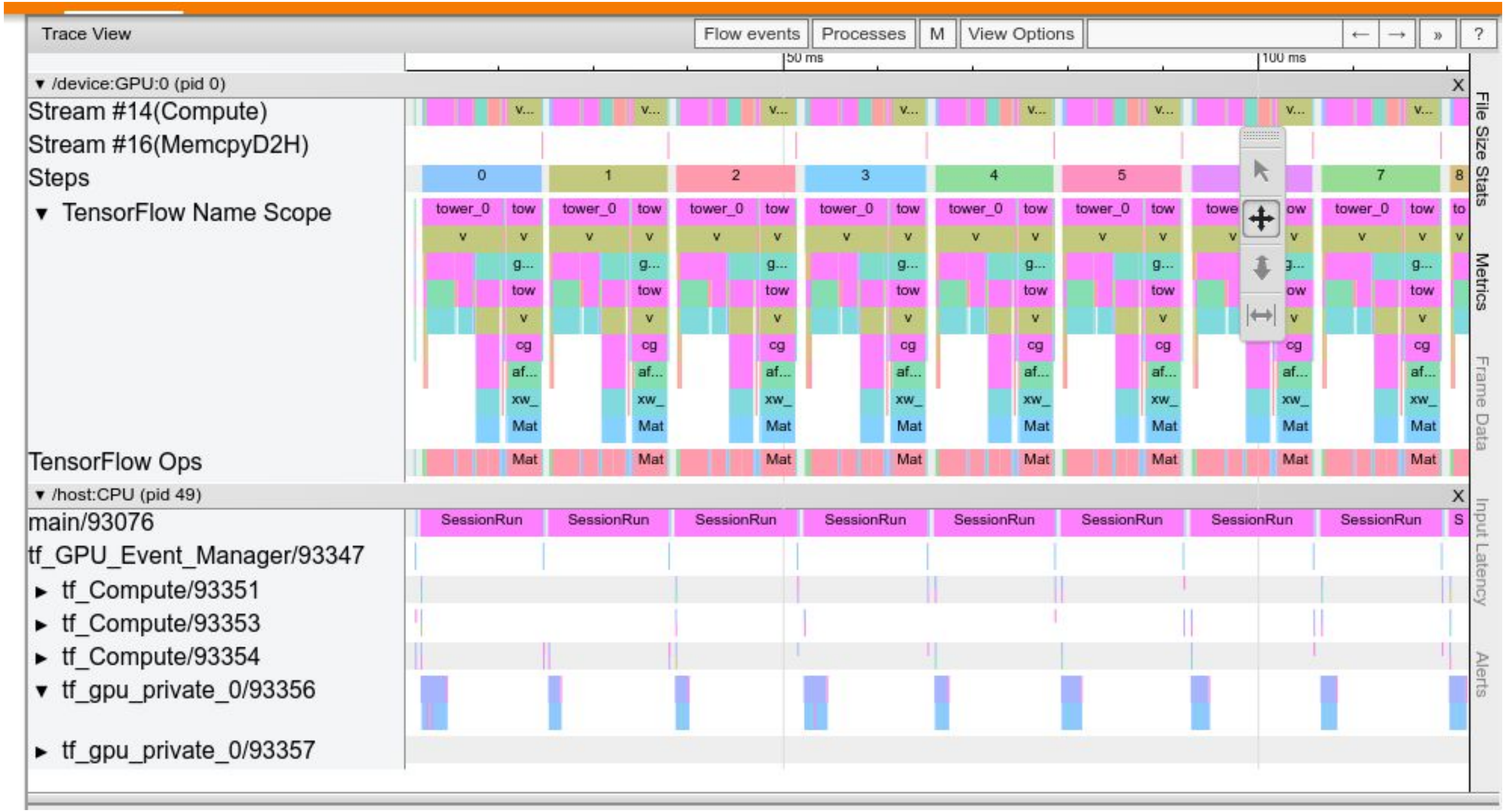



TensorBoard
SCALARS
GRAPHS
PROFILE
INACTIVE
CAPTURE PROFILE
Runs (29)
train/2020-03-02_14:46:29
Tools (5)
overview_page
Hosts (1)
localhost
Performance Summary
Average Step Time
lower is better
(σ = 66.7 ms)
1805.1 ms
All Others Time
(σ = 57.4 ms)
717.5 ms
Compilation Time
(σ = 0.0 ms)
0.0 ms
Output Time
(σ = 0.0 ms)
0.0 ms
Input Time
(σ = 0.1 ms)
0.3 ms
Kernel Launch Time
(σ = 21.6 ms)
327.3 ms
Host Compute Time
(σ = 2.3 ms)
6.7 ms
Device to Device Time
(σ = 0.0 ms)
0.0 ms
Device Compute Time
(σ = 25.8 ms)
753.3 ms
Device Compute Precisions
(out of Total Device Time)
16-bit: 46.3
32-bit: 53.7
Step-time Graph
Step Time (in milliseconds)
2000
1500
1000
500
0
0 1 2 3 4 5 6 7 8 9 10
Step Number
All others
Compilation
Output
Input
Kernel launch
Host compute
Device to device
Device compute
Recommendation for Next Step
Your program is NOT input-bound because only 0.0% of the total step time sampled is waiting for input. Therefore, you should focus on reducing other time.
18.1 % of the total step time sampled is spent on Kernel Launch.
39.7 % of the total step time sampled is spent on All Others time.


TensorBoard
SCALARS
GRAPHS
PROFILE
INACTIVE
CAPTURE PROFILE
Runs (26)
2020-02-25_19:11:57
Tools (4)
input_pipeline_analyzer
Hosts (1)
localhost
Summary of input-pipeline analysis
Your program is HIGHLY input-bound because 82.9% of the total step time sampled is waiting for input. Therefore, you should first focus on reducing the input time.
Recommendation for next step:
Look at Section 3 for the breakdown of input time on the host.
Device-side analysis details
Device step time
Device step-time statistics
Average: 1194.2 ms (σ = 55.1 ms)
Range: 1120.5 - 1318.7 ms
Step Time (in milliseconds)
1500ms
1250ms
1000ms
750ms
500ms
250ms
0ms
Milliseconds
Step Number
All others
Compilation
Output
Input
Kernel launch
Host compute
Device to device
Device compute




| Rank | Host/device | Type | Operation | #Occurrences | Total time (us) | Avg. time (us) | Total self-time (us) | Avg. self-time (us) | Total self-time on Device (%) | Cumulative total-self time on Device (%) | Total self-time on Host (%) | Cumulative total-self time on Host (%) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 43 | Device | MatMul | gradient_tape/replica_3/model/transformer_v2/Transformer/decode/embedding_shared_weights_1/presoftmax_linear/MatMul/MatMul | 11 | 149,200 | 13,564 | 149,200 | 13,564 | 0.3% | 66.3% | 0% | 0% |
| 44 | Device | MatMul | gradient_tape/replica_6/model/transformer_v2/Transformer/decode/embedding_shared_weights_1/presoftmax_linear/MatMul/MatMul | 11 | 149,199 | 13,564 | 149,199 | 13,564 | 0.3% | 66.5% | 0% | 0% |
| 45 | Device | MatMul | gradient_tape/replica_5/model/transformer_v2/Transformer/decode/embedding_shared_weights_1/presoftmax_linear/MatMul/MatMul | 11 | 149,179 | 13,562 | 149,179 | 13,562 | 0.3% | 66.8% | 0% | 0% |
| 46 | Device | MatMul | gradient_tape/model/transformer_v2/Transformer/decode/embedding_shared_weights_1/presoftmax_linear/MatMul/MatMul | 11 | 149,163 | 13,560 | 149,163 | 13,560 | 0.3% | 67.1% | 0% | 0% |



Trace View
Flow events
Processes
M
View Options
50 ms
100 ms
/device:GPU:0 (pid 0)
Stream #14(Compute)
Stream #16(MemcpyD2H)
Steps
TensorFlow Name Scope
TensorFlow Ops
/host:CPU (pid 49)
main/93076
tf_GPU_Event_Manager/93347
tf_Compute/93351
tf_Compute/93353
tf_Compute/93354
tf_gpu_private_0/93356
tf_gpu_private_0/93357
SessionRun
File Size Stats
Metrics
Frame Data
Input Latency
Alerts

# TF 2 Profiler Tool Set

Overview Page

Input Pipeline Analyzer

TensorFlow Stats

Trace Viewer

\+ 4 GPU/TPU expert tools

**Available Today on TensorBoard**

# Next Steps

- Tutorial: https://www.tensorflow.org/tensorboard/tensorboard_profiling_keras
- Guide: https://tensorflow.org/guide/profiler
- Github: https://github.com/tensorflow/profiler
- Two related talks in this afternoon:
    - “Scaling TensorFlow data processing with tf.data”
    - “Scaling TensorFlow 2 models to multi-worker GPUs”

# Thank you!

TensorFlow

DEV SUMMIT 2020