



TensorFlow



76,000,000

downloads

80,000+

commits

13,000+

pull requests

2,400+

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

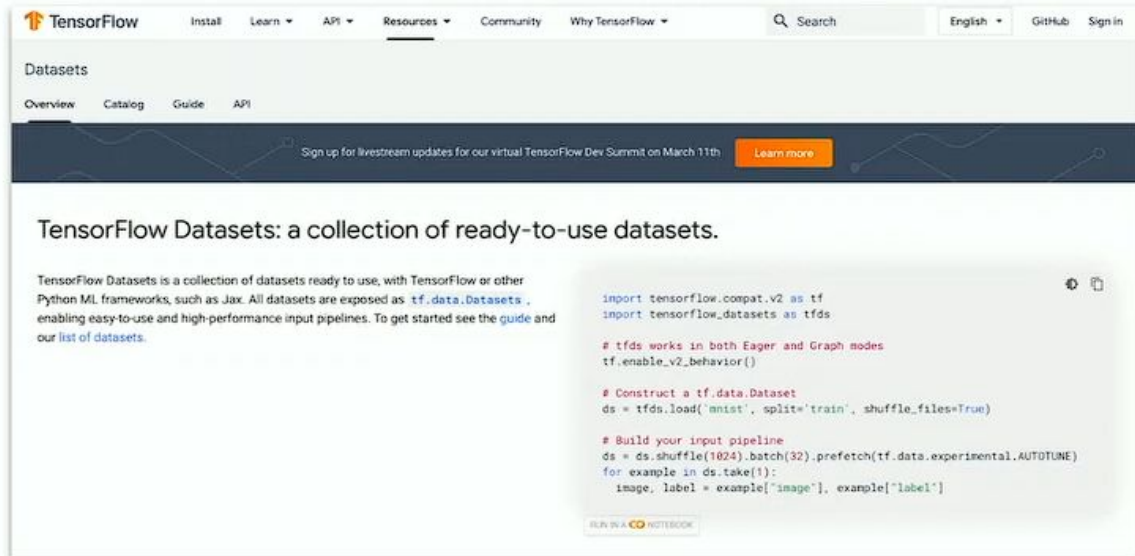


Eager at the core and simple, performant data input pipelines in 2.x

eager execution,
supporting numpy arrays

tf.data

TensorFlow Datasets



The screenshot shows the TensorFlow Datasets website. At the top is the TensorFlow logo and navigation links: Install, Learn, API, Resources, Community, Why TensorFlow. A search bar and language selector (English) are on the right. Below the navigation is a 'Datasets' section with tabs for Overview, Catalog, Guide, and API. A banner for a TensorFlow Dev Summit livestream is present. The main heading is 'TensorFlow Datasets: a collection of ready-to-use datasets.' Below this, a paragraph describes the datasets as ready-to-use with TensorFlow or other Python ML frameworks, exposing them as `tf.data.Datasets`. To the right, a code block shows a Python snippet for loading and using the MNIST dataset in eager mode. At the bottom right of the code block is a 'RUN IN A Jupyter NOTEBOOK' button.

TensorFlow

Install Learn API Resources Community Why TensorFlow

Search English GitHub Sign in

Datasets

Overview Catalog Guide API

Sign up for livestream updates for our virtual TensorFlow Dev Summit on March 11th [Learn more](#)

TensorFlow Datasets: a collection of ready-to-use datasets.

TensorFlow Datasets is a collection of datasets ready to use, with TensorFlow or other Python ML frameworks, such as Jax. All datasets are exposed as `tf.data.Datasets`, enabling easy-to-use and high-performance input pipelines. To get started see the [guide](#) and our [list of datasets](#).

```
import tensorflow.compat.v2 as tf
import tensorflow_datasets as tfds

# tfds works in both Eager and Graph modes
tf.enable_v2_behavior()

# Construct a tf.data.Dataset
ds = tfds.load('mnist', split='train', shuffle_files=True)

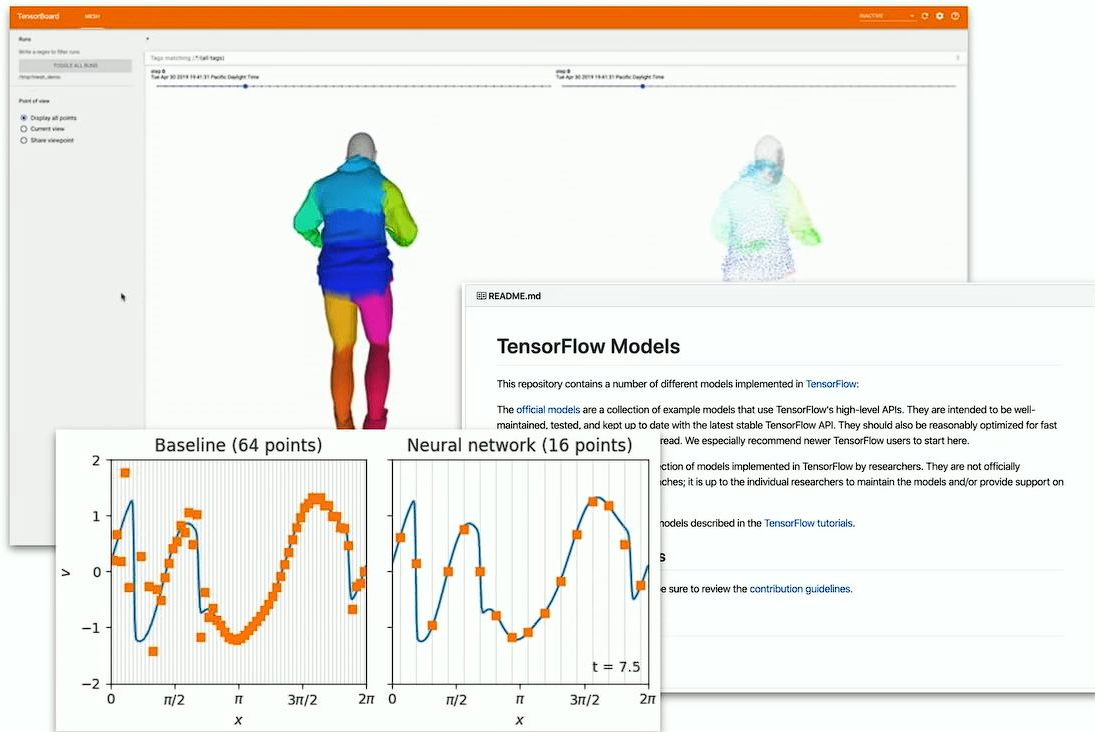
# Build your input pipeline
ds = ds.shuffle(1024).batch(32).prefetch(tf.data.experimental.AUTOTUNE)
for example in ds.take(1):
    image, label = example['image'], example['label']
```

RUN IN A Jupyter NOTEBOOK



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](https://www.tensorflow.org/)!





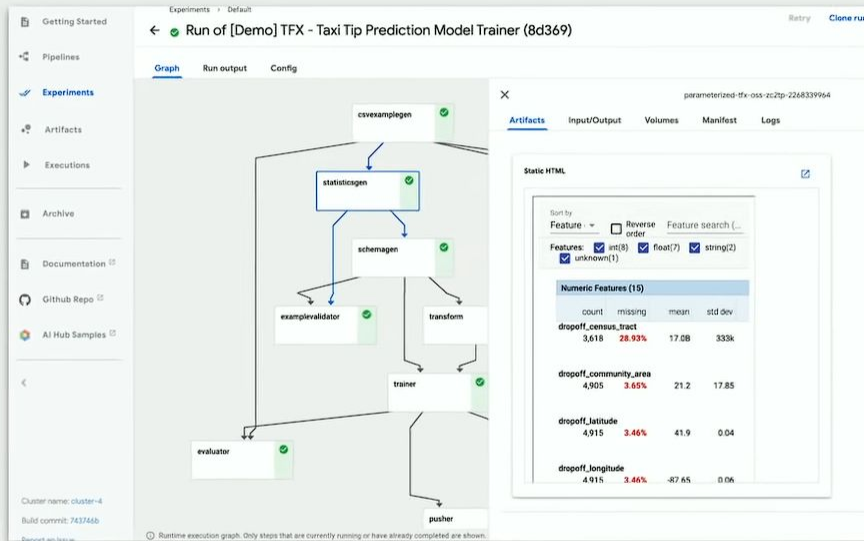
TensorFlow Extended



Google Cloud AI Platform Pipelines



TensorFlow
Enterprise

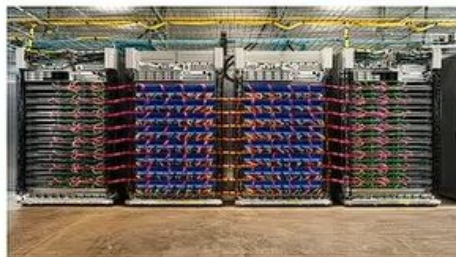
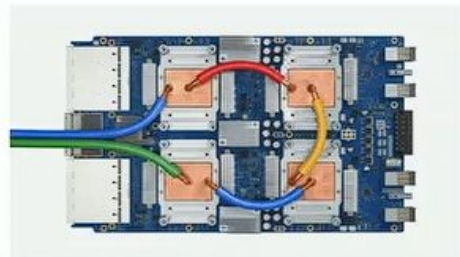




TensorFlow 2.1 supports Cloud TPUs



+





Challenge

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 Bidirectional Encoder Representations from Transformers. Unlike most language representation models (Peters et al., 2018a; Radford et al., 2018), BERT is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right contexts in all layers. As a result, the pre-trained BERT model can be finetuned 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% (7.9% point absolute improvement), MNLI accuracy to 86.7% (4.0% absolute improvement), SQuAD v1.1 question answering F1 to 91.2 (1.5 point absolute improvement) and SQuAD v2 F1 to 83.1 (3.1 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; Brown and Ruder, 2018). These include sentence-level tasks such as natural language inference (Bowman et al., 2015; Williams et al., 2018) and paraphrasing (Olan and Brockett, 2005), which aids in predicting the relationship 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 (Yang, Kim, Song, and De Meester, 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 unidirectional 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: Bidirectional Encoder Representations from Transformers. BERT addresses the previously mentioned unidirectionality constraint by using a “masked language model” (MLM) pre-training objective, inspired by the Cloze task (Lytic, 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



Search or jump to...

🔍

Pull requests

Issues

Marketplace

Explore

tensorflow / models

Used by 7

Unwatch 3k

★ Unstar 62.1k

🔗 Fork 39.7k

Code

Issues 1,694

Pull requests 239

Actions

Projects 2

Wiki

Security

Insights

Branch master

models / official /

Create new file

Upload files

Find file

History

longmize and tensorflow-gardner

Internal change (36)

Latest commit c781596 13 hours ago

..

benchmark

Internal change

20 hours ago

modeling

Internal change

13 hours ago

nlp

Internal change

13 hours ago

pip_package

Add setup.py for official models.

4 months ago

rl

Convert numpy arrays to floats using float constructor, not .item().

9 days ago

recommendation

Use unittest.mock as we are py3 now

10 days ago

staging

Internal change

13 hours ago

utils

Add TimeHistory callback to BERT.

3 days ago

vision

Internal change

13 hours ago

LICENSE

Merge Transformer V2 to Github (f6646)

10 months ago

README-TPU.md

Internal change

3 days ago

README.md

Readme updated

10 days ago



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

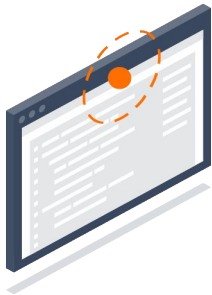


Ready to use

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



TensorFlow
Extended



TensorFlow
.JS



TensorFlow
Lite



Coral

[← 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.

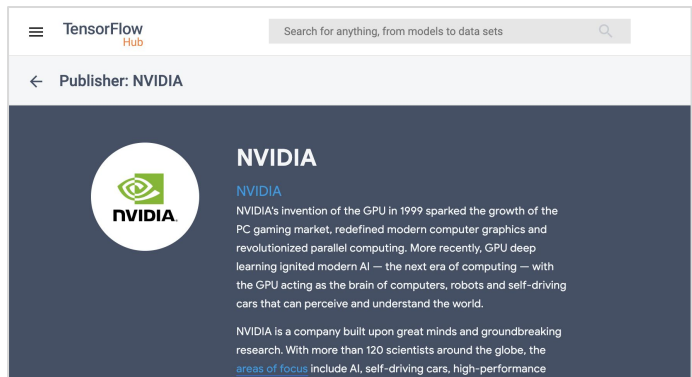
Mobilenets come in various sizes controlled by a multiplier for the depth (number of features) in the convolutional layers. They can also be trained for various sizes of input images to control inference speed.



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...





Kaggle competition now supports 2.x

No setup

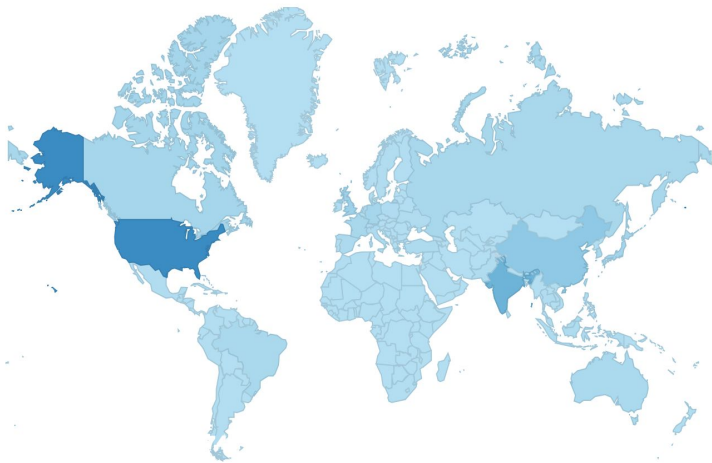
No provisioning

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

Active

Completed

InClass

All Categories ▾

Reward ▾



Deepfake Detection Challenge

Identify videos with facial or voice manipulations

Featured • a month to go • Code Competition • 1917 Teams

\$1,000,000



Flower Classification with TPUs

Use TPUs to classify 104 types of flowers

Playground • 3 months to go • Code Competition • 244 Teams

Kudos



Abstraction and Reasoning Challenge

Create an AI capable of solving reasoning tasks it has never seen before

Research • 3 months to go • Code Competition • 277 Teams

\$20,000



Google Cloud & NCAA® ML Competition 2020-NCAAM

Apply Machine Learning to NCAA® March Madness®

Featured • 23 days to go • 352 Teams

\$25,000



Connect X

Connect your checkers in a row before your opponent!

Getting Started • Ongoing • Simulation Competition • 473 Teams

Knowledge

Featured Code Competition

Deepfake Detection Challenge

Identify videos with facial or voice manipulations

\$1,000,000
Prize Money

Deepfake Detection Challenge • 1,917 teams • a month to go (7 days to go until merger deadline)

[Overview](#) [Data](#) [Notebooks](#) [Discussion](#) [Leaderboard](#) [Rules](#) [Team](#) [My Submissions](#) [Submit Predictions](#)

Overview

Description

Deepfake techniques, which present realistic AI-generated videos of people doing and saying fictional things, have the potential to have a significant impact on how people determine the legitimacy of information presented online. These content generation and modification technologies may affect the quality of public discourse and the safeguarding of human rights—especially given that deepfakes may be used maliciously as a source of misinformation, manipulation, harassment, and persuasion. Identifying manipulated media is a technically demanding and rapidly evolving challenge that requires collaborations across the entire tech industry and beyond.

Timeline

Prizes

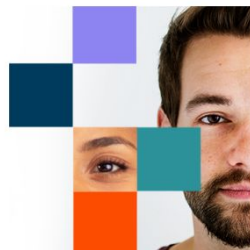
Code Requirements

Getting Started

Evaluation

AWS, Facebook, Microsoft, the [Partnership on AI's Media Integrity Steering Committee](#), and academics have come together to build the Deepfake Detection Challenge (DFDC). The goal of the challenge is to spur researchers around the world to build innovative new technologies that can help detect deepfakes and manipulated media.

Challenge participants must submit their code into a black box environment for testing. Participants will have the option to make their submission open or closed when accepting the prize. Open proposals will be eligible for challenge prizes as long as they abide by the open source licensing terms. Closed proposals will be proprietary and not be eligible to accept the prizes. Regardless of which track is chosen, all submissions will be evaluated in the same way. Results will be shown on the leaderboard.





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]



spear thistle [OK]



rose [NO→geranium]



yellow iris [NO→king protea]



rose [NO→hibiscus]



morning glory [NO→watercress]



iris [NO→fritillary]



iris [NO→yellow iris]



common dandelion [NO→blanket flower]



thorn apple [OK]



daisy [OK]



rose [NO→camellia]



wallflower [OK]



iris [OK]



common dandelion [NO→tree poppy]



iris [NO→wild rose]



petunia [NO→hibiscus]



daisy [NO→orange dahlia]



daisy [OK]



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

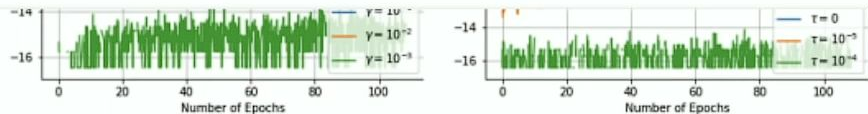
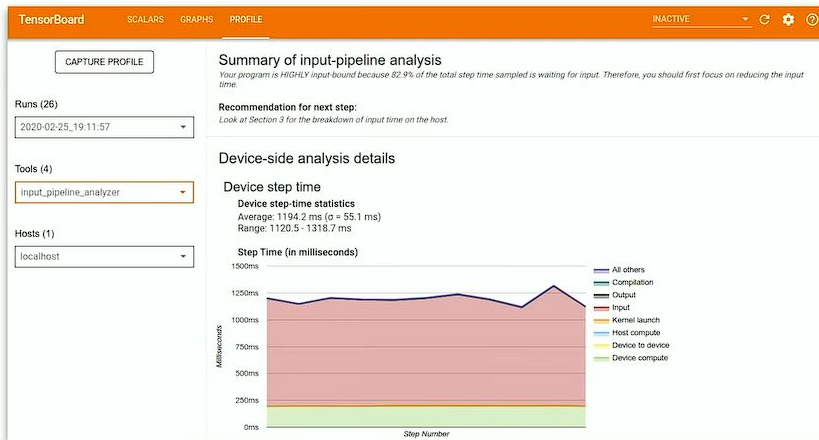


Figure 2: On the left τ is fixed at 10^{-5} and we compare three values for γ . On the right γ is fixed at 10^{-2} and we compare three values for τ . The tensorboard graphs can be found at <https://tensorboard.dev/experiment/up4UbhojT6uZKjnkPyZaRQ>



Performance Profiler

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

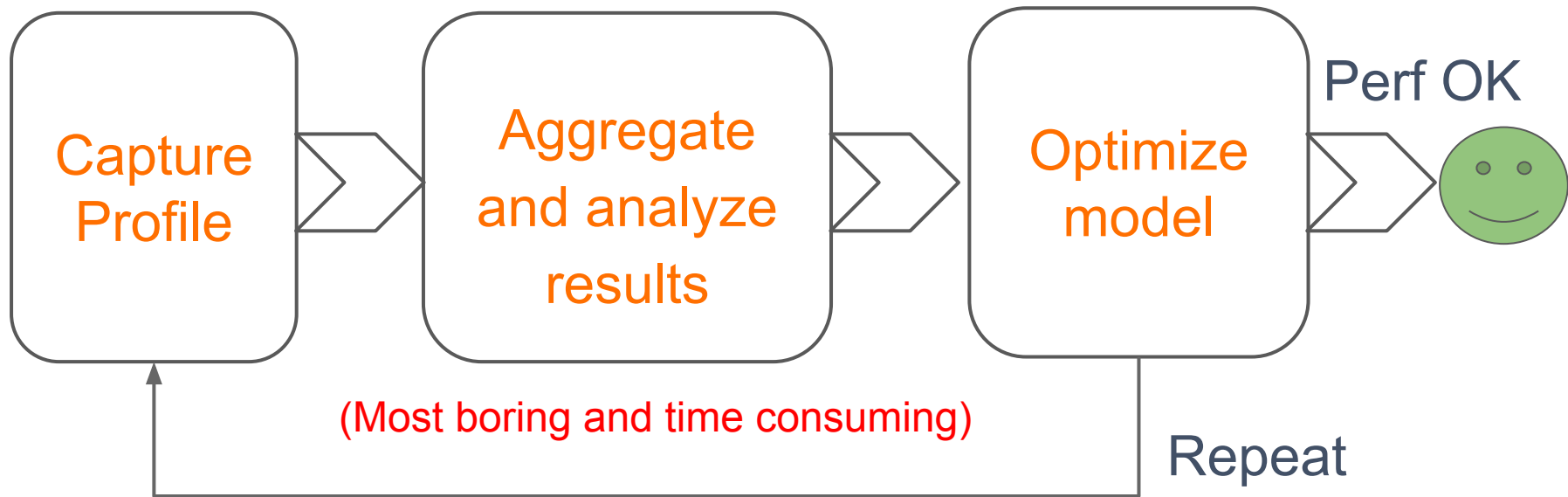




TF is speed!

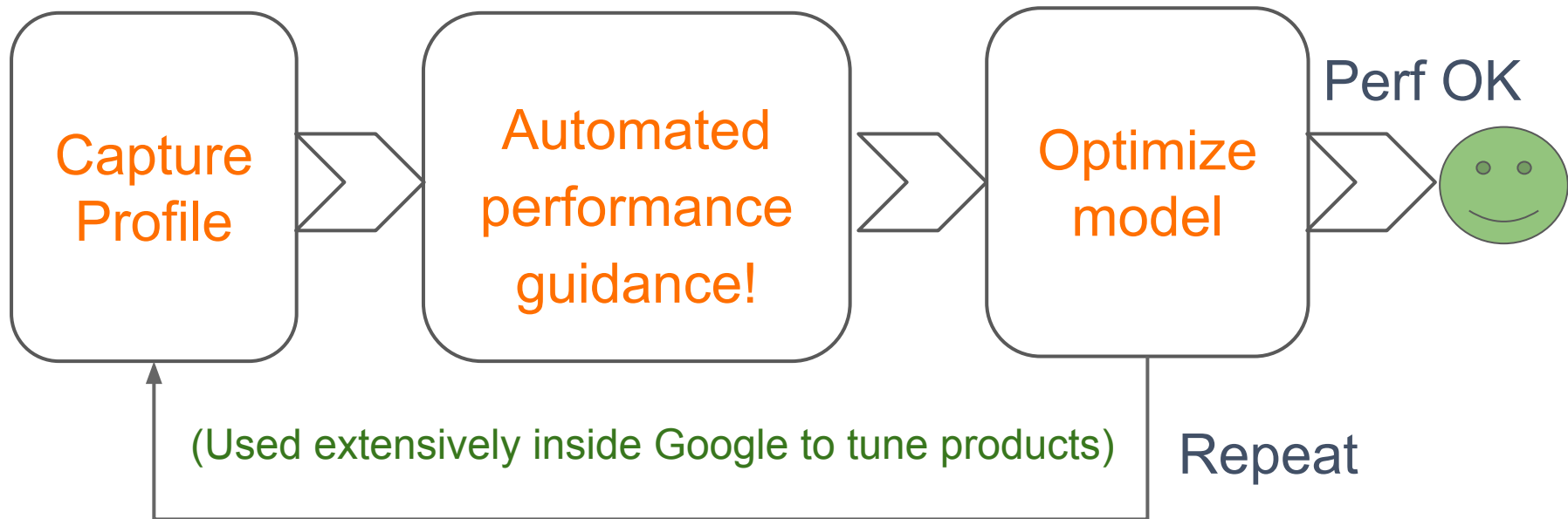


Life of a performance engineer





ML productivity





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
($\sigma = 66.7$ ms)1805.1
ms

• All Others Time

($\sigma = 57.4$ ms)

717.5 ms

• Compilation Time

($\sigma = 0.0$ ms)

0.0 ms

• Output Time

($\sigma = 0.0$ ms)

0.0 ms

• Input Time

($\sigma = 0.1$ ms)

0.3 ms

• Kernel Launch Time

($\sigma = 21.6$ ms)

327.3 ms

• Host Compute Time

($\sigma = 2.3$ ms)

6.7 ms

• Device to Device Time

($\sigma = 0.0$ ms)

0.0 ms

• Device Compute Time

($\sigma = 25.8$ ms)

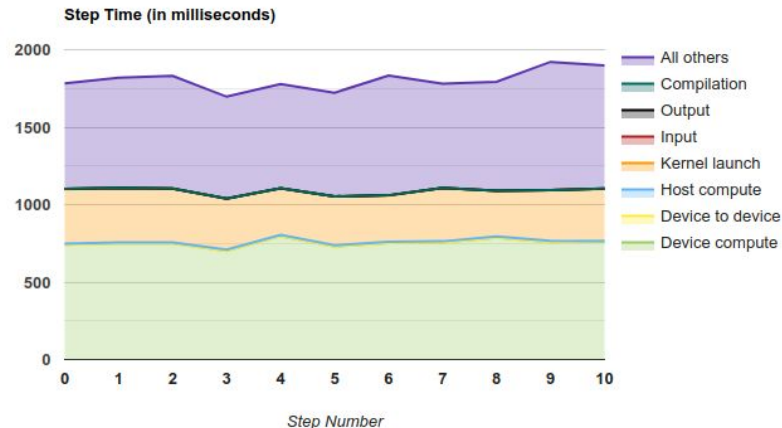
753.3 ms

Device Compute Precisions

(out of Total Device Time)

- 16-bit: 46.3
- 32-bit: 53.7

Step-time Graph



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.



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

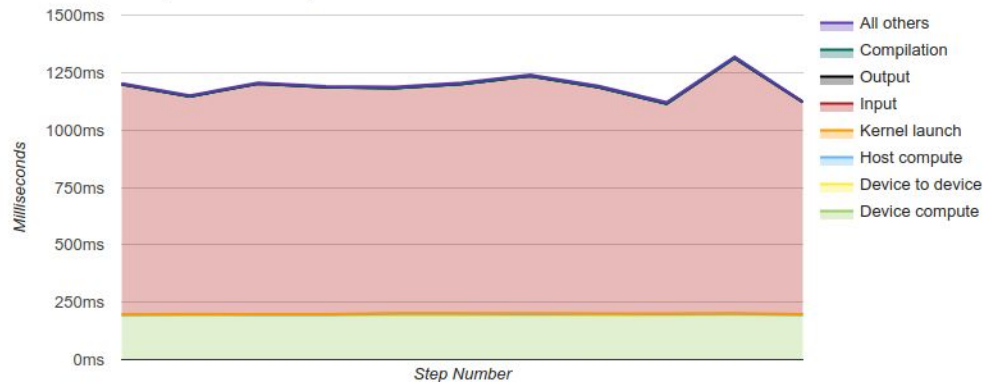
Device step time

Device step-time statistics

Average: 1194.2 ms ($\sigma = 55.1$ ms)

Range: 1120.5 - 1318.7 ms

Step Time (in milliseconds)





TensorBoard

SCALARS

GRAPHS

PROFILE

INACTIVE



CAPTURE PROFILE

Runs (29)

train/2020-03-02_14:46:29

Tools (5)

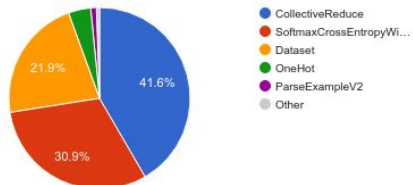
tensorflow_stats

Hosts (1)

localhost

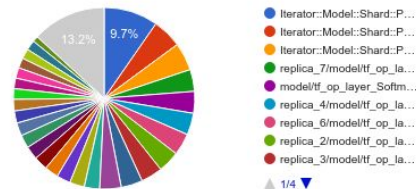
ON HOST: TOTAL SELF-TIME (GROUPED BY TYPE)

(in microseconds) of a TensorFlow operation



ON HOST: TOTAL SELF-TIME

(in microseconds) of a TensorFlow operation



TensorFlow operations

Host/device



Type

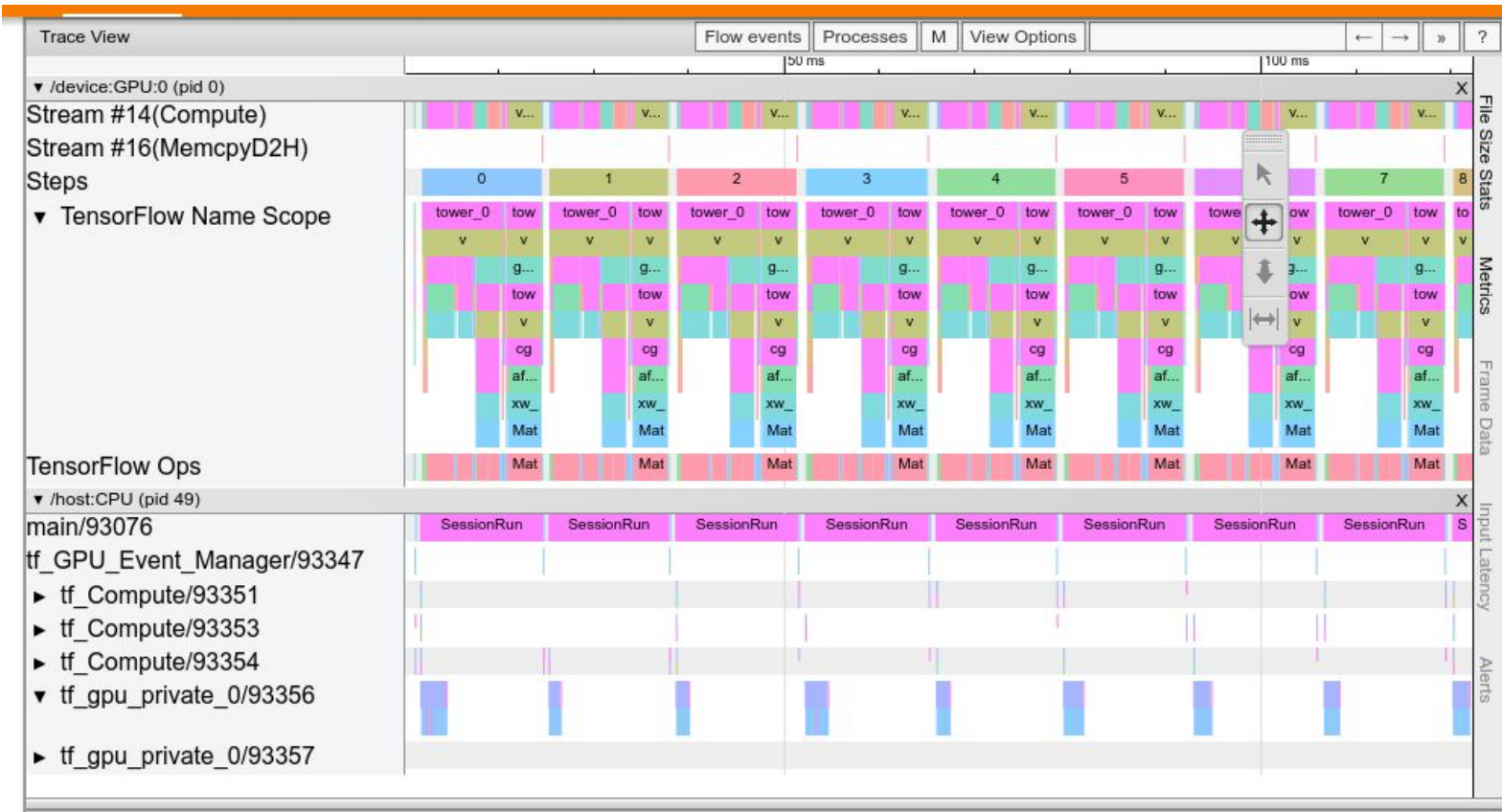


Operation



Note: To avoid sluggishness, only the 1000 most time-consuming operations out of the total 36340 operations are shown in the table below.

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%





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