**TensorFlow Dev Summit 2017**

**@tensorflow #tfdevsummit**

**Notes By: Erica Lee**\* (means it’s an “Erica” favorite)

**These Notes are at** [**http://bit.ly/tfdevsummit**](http://bit.ly/tfdevsummit)

**Schedule**

7:30 am Breakfast Opens

7:30 am Registration Opens — please bring a government-issued ID

9:20 am Doors open for Keynote

9:30 am Keynote

1:00 pm Lunch

5:00 pm Registration Ends

5:30 pm Afterparty with great food & drinks

**Featured Talks**

* Keynote — Jeff Dean, Rajat Monga, and Megan Kacholia
* XLA: TensorFlow, Compiled! — Todd Wang and Chris Leary
* TensorFlow High-Level API — Martin Wicke
* Keras and TensorFlow — François Chollet
* Distributed TensorFlow — Derek Murray
* Creating Art & Music with Project Magenta — Doug Eck
* TensorFlow Ecosystem: Hadoop and More — Jonathan Hseu
* Serving Models in Production with TensorFlow Serving — Noah Fidel
* Memorization + Generalization with TensorFlow Wide and Deep — Heng-Tze Cheng
* Hands-on TensorBoard — Dandelion Mane
* Mobile and Embedded TensorFlow — Pete Warden
* ML Toolkit — Ashish Agarwal
* Sequence Models and the RNN API — Eugene Brevdo
* TensorFlow in Health — Lily Peng

More information about the agenda and speakers [on our website](https://www.google.com/appserve/mkt/p/1m2kSAoyheGbJALC9S6TzHIBxoTb07wCk3F_RBcvVPmR0AOyymLqIU4OSBb0puAkaCTKku6kid60zLVF5_9Zac2knBCwc1wOcYcelOPSN57nrg==)!

**Talk Notes**

Keynote — Jeff Dean, Rajat Monga, and Megan Kacholia

* DistBelief - pre-TF days, built for CPUs though
* New Product Launch - TensorFlow v1.0\*
  + 58x speedup on 64 GPUs for Inception v3
  + OS = Android, iOS, CPU GPU, ....
  + TF Distributed Execution Engine
  + Frontend = python, C++
  + Added = Keras support, canned estimators,
  + Broad ML support
    - k-means (cluster)
    - SVM (classification)
    - random forest (classification and regression)
  + Broad hardware support
    - Intel PowerAI
    - Movidius Myraid 2 accelerator
    - Qualcomm Hexagon DSP
  + XLA = experimental TF Compiler
* Neural Machine Translation
* TF Use Cases
  + Cucumber sorting
  + Detecting hemorrhages in eyes before early blindness onsets

**XLA: TensorFlow, Compiled! — Todd Wang and Chris Leary**

* **TF New Feature: XLA compiler**
  + How is it done?
  + Flexible + expressive + blackbox modular
  + Just in time compilation = XLA = Accelerated Linear Algebra compiler \*
  + Need = for developers to start using it!
  + JIT “just in time” compilation = get optimization benefits and specialization
  + Model-shaped benchmarks
    - XLA’s just in time compillation
    - Ex. SyntaxNet latency reductions: 200 microsec ---> 5 micro sec
    - Mobile footprint reductions
      * XLA’s ahead of time compilation
      * Turn models to executables
      * Ex. LSTM Model for mobile: 2.6 Mib ---> <600 Kib
  + Whole-Program analysis made easy
    - XLA’s high-level optimizer
  + Documentation: [tensorflow.org/performance/xla](http://tensorflow.org/performance/xla)
  + 4 libraries:
    - tensorflow/compiler/xla
    - tensorflow/compiler/tf2xla
    - tensorflow/compiler/---
    - tensorflow/compiler/---
  + XLA graph → LLVM IR (toolkit framework, intermediate representation) → code generation for machine platforms (x86 binary, arm binary, PTX binary)
  + Turning on JIT compilation
  + tensorflow.tfcompile.Config

[Code form TF GitHub](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/compiler/aot/test_graph_tfadd.config.pbtxt)

# Text form of tensorflow.tfcompile.Config proto.

feed {

id { node\_name: "x\_const" }

shape {

dim { size: 1 }

}

}

feed {

id { node\_name: "y\_const" }

shape {

dim { size: 1 }

}

}

fetch {

id { node\_name: "x\_y\_sum" }

* + Compile your graph using tf\_library bazel build macro

[Code form TF GitHub](https://github.com/tensorflow/tensorflow/blob/master/tensorflow/compiler/aot/tfcompile.bzl)

"""Build macro that compiles a TensorFlow graph into a cc\_library.

To use from your BUILD file, add the following line to load the macro:

load("//tensorflow/compiler/aot:tfcompile.bzl", "tf\_library")

Then call the macro like this:

tf\_library(

name = "test\_graph\_tfmatmul",

config = "test\_graph\_tfmatmul.config.pbtxt",

cpp\_class = "MatMulComp",

graph = ":test\_graph\_tfmatmul.pb",

)

"""

**Hands-on TensorBoard — Dandelion Mane**

* MNIST architecture
* Conv 1
* Pooling 1
* Conv 2
* Pooling 2
* Fully-connected 1
* Fully-connected 1
* Train the model
  + Loss & training function
    - Cross entropy
* Cleaning the graph
  + Set up graph, pass names, name=”blah”
* Collect some summaries
  + Ex:
    - tf.summary.scalar
    - tf.summary.image
    - tf.summary.tensor (under development)
* TensorBoard - visual graphs of dataflow, like seeing a histogram
  + Easy display change, toggle tools to visual time
* Hyperparameter Search\*
  + Compare different learning rates and model architectures?
* Embedding Visualizer
  + Turn high dimensionality into lower dimensionality

>>> tensorboard --logdir /tmp/mnist\_tutorial

[Slides at goo.gl/San2uR](https://gist.github.com/dandelionmane/4f02ab8f1451e276fea1f165a20336f1#file-mnist-py)

**TensorFlow High-Level API — Martin Wicke**

* Estimator
  + Inputs, labels -- > model\_fn ---> 3 paths = fit(), evaluate(), predict (), export\_savedmodel()\*
  + Estimators combine the model\_fn with sessions, graphs, loops to let you focus on what’s important

**Keras and TensorFlow — François Chollet**

* Keras is an API spec for building DL models across many platforms
* [**Keras**](https://keras.io/)
* Keras isa high-level neural networks library, written in Python and capable of running on top of either [TensorFlow](https://github.com/tensorflow/tensorflow) or [Theano](https://github.com/Theano/Theano). It was developed with a focus on enabling fast experimentation. With over 60,000 users and over 200 open-source contributors, Keras is used by top startups, large companies and research labs, including CERN, Microsoft Research, OpenAI, Netflix, Yelp, Square, Google, and many more.

tf.keras

* Easily mix and match pure TF & Keras functionality
  + Distributed training
  + Hyperparameter tuning
  + Adopt only parts of Keras code, drop into code base, and go from there
  + Re-use any existing Keras codebase with TF platform
* Audio improvements using DL
  + Comparing WaveNet vs. other models - human speech was improved
  + Music Genertion - trained on corpus of classical music, results were
  + [Learning to learn by gradient descent by gradient descent](https://github.com/deepmind/learning-to-learn)
* More info:
  + <https://github.com/deepmind/learning-to-learn>
  + <https://deepmind.com/blog/deepmin-round-up-2016/>

**Mobile and Embedded TensorFlow — Pete Warden**

* **Building TF for Android**
  + Android examples
    - TF Detect - identify if person or now
    - TF Stylize - neural style transfer like prisma-ai.com
    - Also: YOLO - open source project for image tracking
  + Android Interference Library
  + <https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android>
* **Building TF for iOS**
  + Need XCode8, pass in “-OS” flag, use apple Accelerate framework internally for fast matrix multiplies

Brew install automake

Brew install libtool

* **Building TF for Raspberry Pi**
  + Very similar to normal Linux
  + Don’t recommend cross-compilation
  + Easy to use custom trained model from TensorFlow for Poets
* Error Msg: “No session factory registered for the given session options”
  + This means use “ --whole archive or --40”
  + TF registers providers like sessions using global C\_\_ constructors
* Inception-v3 is trained for the ImageNet Large Visual Recognition Challenge using the data from 2012. \*
  + <https://www.tensorflow.org/tutorials/image_recognition>

**<lunch-break>**

**Distributed TensorFlow — Derek Murray**

* Distributed power - train bigger models on more GPUs
* Migration tools - move random forest or other ML models onto the TF platform
* TF treats local and remote devices the same - split up graphs and transfers tensors
* Distributed device placement\*
  + PS tasks vs. Worker tasks (different serves you send data to)
  + PS tasks
    - Variables
    - Update ops
  + Worker tasks
    - Pre-processing
    - Loss calculation
    - Backpropagation
  + In graph replication - don’t replicate data on local, but send to all other servers
  + Between graph replication - run smaller client program on graph for single replica, in batch
  + Loading balancing variables
    - GreedyLoadBalancingStrategy - unpack load one by one,

Greeedy = if.contrib.training.GreedyLoadBalancingStrategy(...)

With tf.device(tf.train.replica\_device\_setter()

* Device Placement

tf.train.replica\_device\_setter()

* Sessions and Servers
  + Distributed TF runs on a cluster of servers

ClusterSpec = maps name of jobs to network address, cluster defines the set of processes,

Server = server represents a particular task in cluster,

tf.session(server.target) = sess can run code on any device in cluster

**TensorFlow Ecosystem: Hadoop and More — Jonathan Hseu**

* Data treatment
  + Using: Apache Spark, Hadoop MapReduce
  + Preprocessing
    - MySQL + Hive → Hadoop
  + Input Formats (fastest to slowest)
    - if.Example and tf.SequenceExample protocol buggers in TFRecords files
    - Native TF ops reads .CSV and .JSON
    - Feed data directly from Python (common & most flexible, but slowest)
      * Py → numpy arrays ---> tensors
* Training
  + Local - on your machine or VM, debugging and smaller data sets
  + Distributed - finish training faster, but needs appropriate infrastructure
  + Cluster Manager
    - Kubernetes, mesos, hadoop
    - Also Spark is easiest way to get started as data processer instead of CM (Spark on top of Y
  + Distributed storage
    - Google cloud storage, hadoop HDFS
  + Container Engine
    - Rkt, docker
  + Distributed training refresher
    - Between graph replication - run smaller client program on graph for single replica, in batch

**Serving Models in Production with TensorFlow Serving — Noah Fidel**

* **Serving**
  + After you’ve cleaned, trained, ready to deploy and apply ML models (serve)
* **Goals**
  + Online, low latency
  + Multiple models in single process - streamlined
* **What is TensorFlow serving?**
  + C++ libraries = used by Google to make binaries
    - Basic functions like save, export, etc.
  + Binaries - help scale
    - Like docker containers
* Libraries
  + Core platform is all generic - any C++
  + Batcher for inference performance

**ML Toolkit — Ashish Agarwal**\*

* SVM
  + Non linear kernals coming soon
* SDCA (stochastic dual coordinate ascent)
* Random forests
  + Gradient boosting trees coming soon
  + Example: estimator API
    - # create the model

Model = KMeansCluster(num\_clusters=1000)

* + - # fit the model

Model.fit

* + Example: co-train KMeans and DNN
* Distributed Implementations

**Sequence Models and the RNN API — Eugene Brevdo**

* Sequence-to-sequence models
  + Encoder ---(thought vector)---> decoder
* Reading and batching sequence data
* Feeding Sequence data
  + SequenceExample to store seq. Data + context
* The RNN API\*
  + An RNN is a unit of computation you repeat over and over again
  + Forward calculation
  + Get inputs from below X at every time step t, and get errors from previous step, calculates and gets h and updated step
* RNNCell\*
  + Base class, provides properties for RNN architecture
  + Represent a time step as a layer (c.f. Keras layers)
* Calculation
  + Handle sequences of unknown length
    - dynamic calculation

tf.while\_loop

* + - At every time step

tf.TensorArray

* Train with XLA and test against intended benchmark
* Manually-Fused RNN Loops
  + Train with CUDNN, inference on CPU/mobile
* Dynamic Decoding - forthcoming API
  + Sampling method
  + Define RNN Architecture
  + Build decoder
  + Run decover

**Memorization + Generalization with TensorFlow Wide and Deep — Heng-Tze Cheng**

* Combine mem with gen on one unified ML platform for everyone
* Memorization - “pigeons can fly”
  + Very precise, first step in learning → “wide”
* Generalization - “animals with wings can fly”
  + One compact representation of knowledge → “deep”
* Generalization + memorizing exceptions - “animals with wings can fly, except for penguins”

**Creating Art & Music with Project Magenta — Doug Eck**

* Magenta: Music and Art Generation with Machine Intelligence
* <https://github.com/tensorflow/magenta>
* Critical to have feedback loop between computer generated content and human consumption/feedback on it

**TensorFlow in Health — Lily Peng**

* Former MD, now PM @ Google
* Adapt NN to read fundus images
  + Started with 130k images