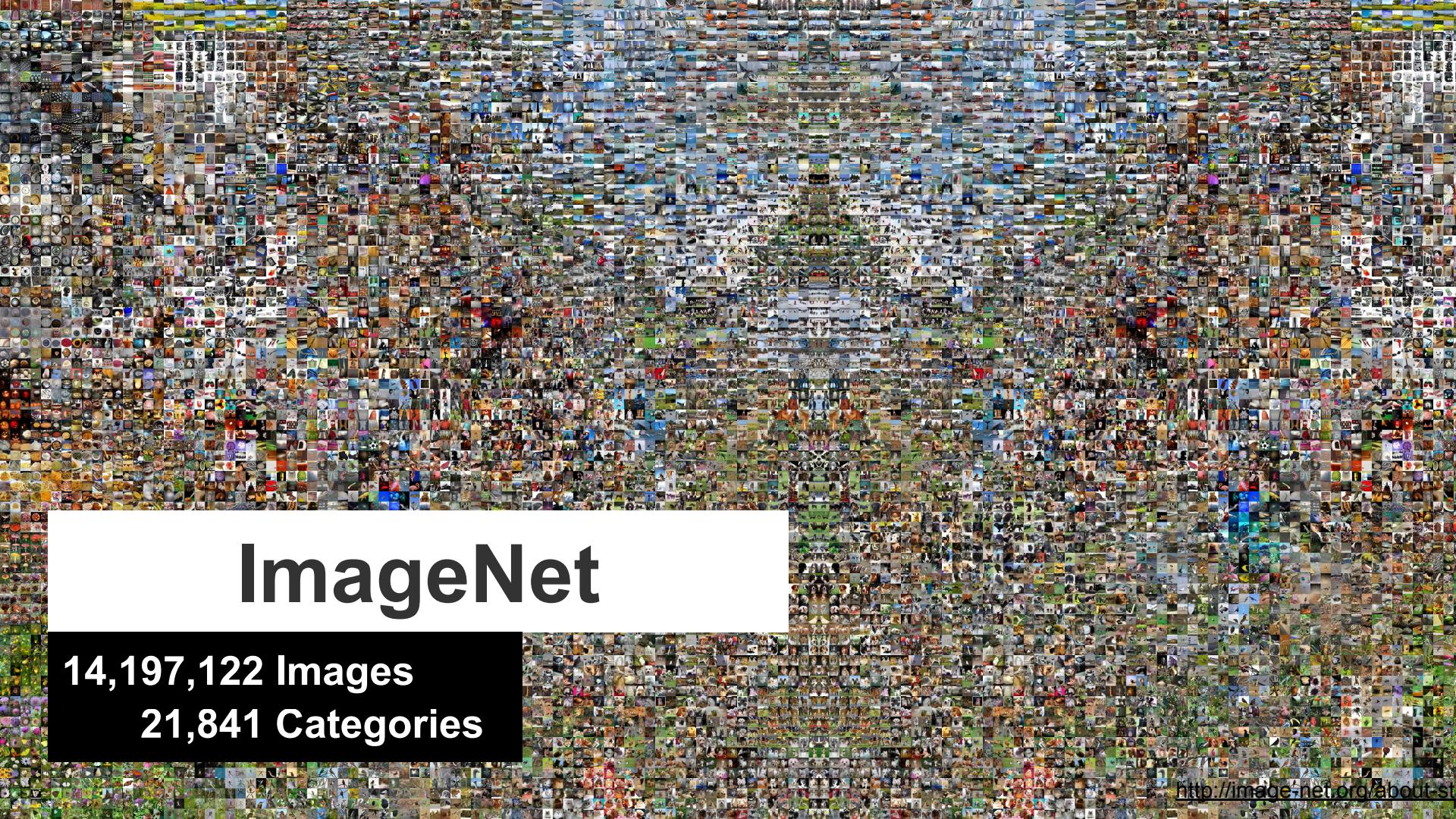


Training ImageNet using TensorFlow and R

@javierluraschi

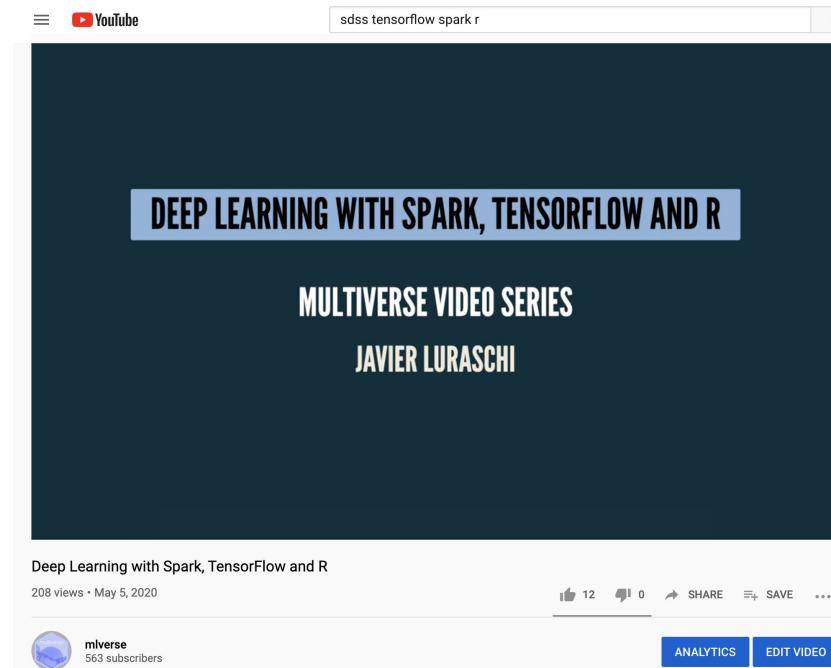
DSSV 2020 / RStudio PBC



ImageNet

14,197,122 Images
21,841 Categories

Need to catch up?



[YouTube mlverse -- Deep Learning with Spark, TensorFlow and R](#)



Making neural nets
uncool again

[Home](#)

[About](#)

[Our MOOC](#)

Now anyone can train Imagenet in 18 minutes

Written: 10 Aug 2018 by *Jeremy Howard*

This post extends the work described in a previous post, [Training Imagenet in 3 hours for 25; and CIFAR10 for 0.26.](#)

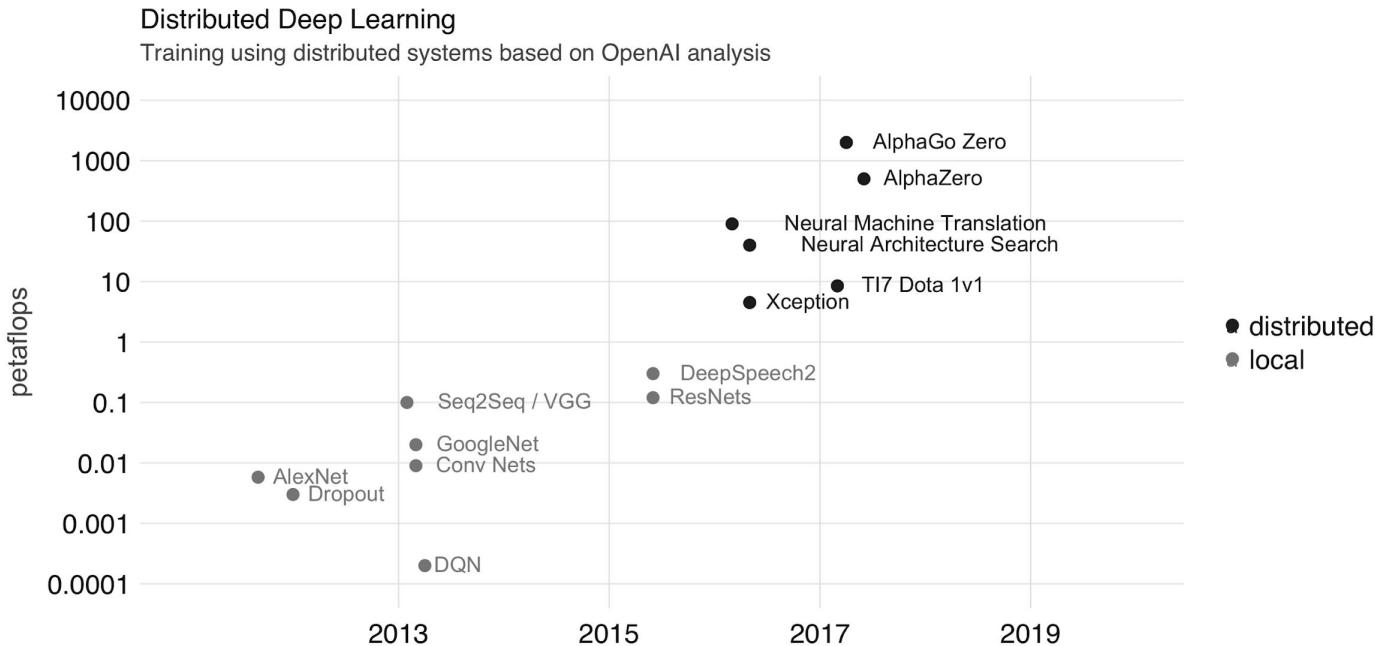
A team of fast.ai alum Andrew Shaw, [DIU](#) researcher Yaroslav Bulatov, and I have managed to train [Imagenet](#) to 93% accuracy in just 18 minutes, using 16 public [AWS](#) cloud instances, each with 8 [NVIDIA V100](#) GPUs, running the [fastai](#) and [PyTorch](#) libraries. This is a new speed record for training Imagenet to this accuracy on publicly available infrastructure, and is 40% faster than Google's [DAWNBench](#) record on their proprietary [TPU Pod](#) cluster. Our approach uses the same number of processing units as Google's benchmark (128) and costs around \$40 to run.

DIU and fast.ai will be releasing software to allow anyone to easily train and monitor their own distributed models on AWS, using the best practices developed in this project. The main training methods we used (details below) are: fast.ai's progressive resizing for classification, and rectangular image validation; NVIDIA's NCCL with PyTorch's all-reduce; Tencent's weight decay tuning; a variant of Google Brain's dynamic batch sizes, gradual learning rate warm-up (Goyal et al 2018, and Leslie Smith 2018). We used the classic ResNet-50 architecture, and SGD with momentum.

<https://www.fast.ai/2018/08/10/fastai-diu-imagenet/>

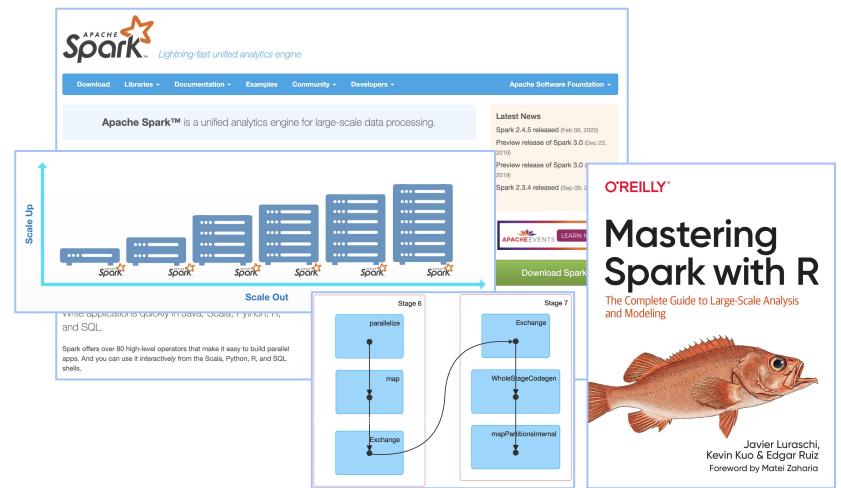
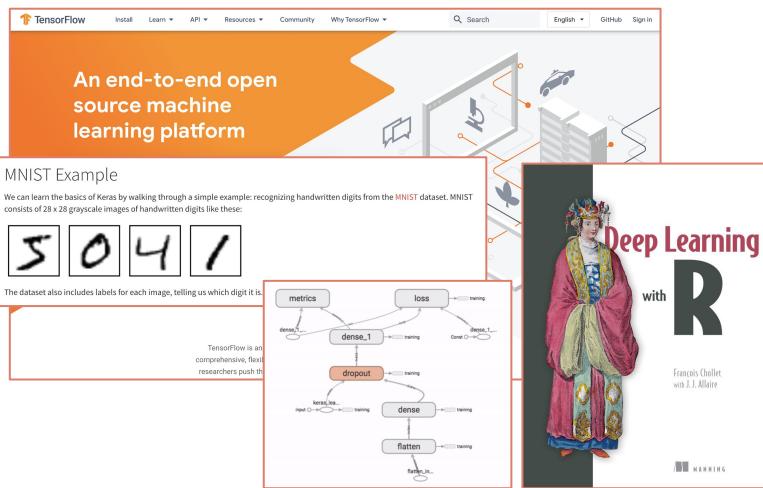
Distributed Deep Learning

Recent state-of-the-art models require distributed training.



Overview

This talk assumes you are somewhat familiar with TensorFlow and Spark:



Talk Outline: **Distributed Deep Learning, TensorFlow and Spark.**

Distributed Deep Learning

Deep Learning Recap: AlexNet

ImageNet Classification with Deep Convolutional Neural Networks

Alex Krizhevsky
University of Toronto
kriz@cs.utoronto.ca Ilya Sutskever
University of Toronto
ilya@cs.utoronto.ca Geoffrey E. Hinton
University of Toronto
hinton@cs.utoronto.ca

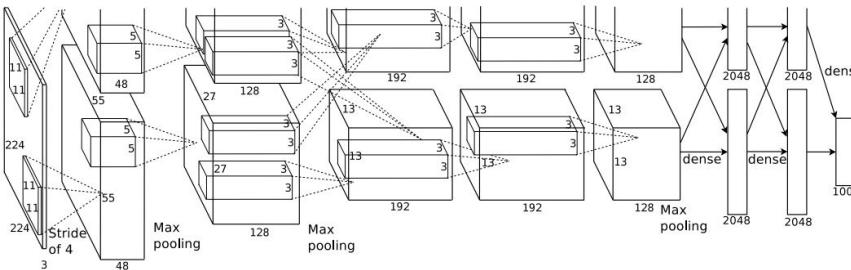
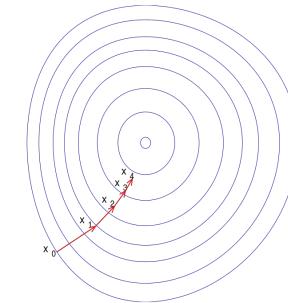


Figure 2: An illustration of the architecture of our CNN, explicitly showing the delineation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom. The GPUs communicate only at certain layers. The network's input is 150,528-dimensional, and the number of neurons in the network's remaining layers is given by 253,440–186,624–64,896–64,896–43,264–4096–4096–1000.

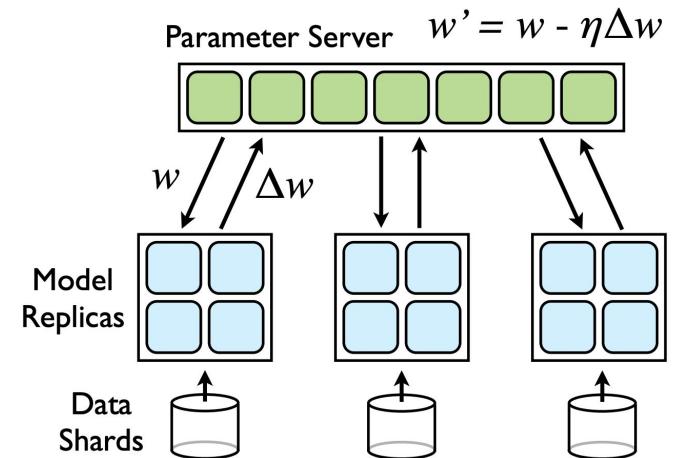


$$\mathbf{a}_{n+1} = \mathbf{a}_n - \gamma \nabla F(\mathbf{a}_n)$$

Parameter Server @Google

Large Scale Distributed Deep Networks

Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen,
Mathieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato,
Andrew Senior, Paul Tucker, Ke Yang, Andrew Y. Ng
`{jeff, gcorrado}@google.com`
Google Inc., Mountain View, CA

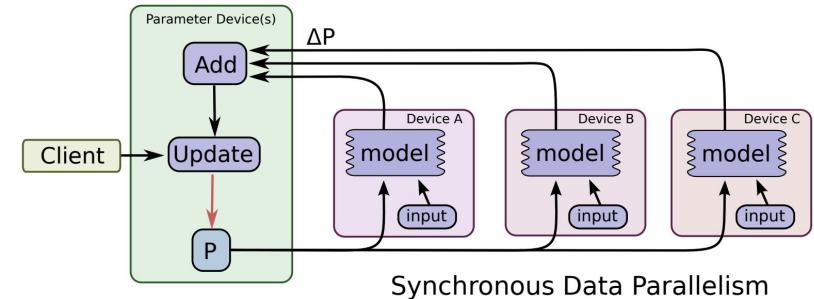


the same order. Moreover, because the model replicas are permitted to fetch parameters and push gradients in separate threads, there may be additional subtle inconsistencies in the timestamps of parameters. There is little theoretical grounding for the safety of these operations for nonconvex problems, but in practice we found relaxing consistency requirements to be remarkably effective.

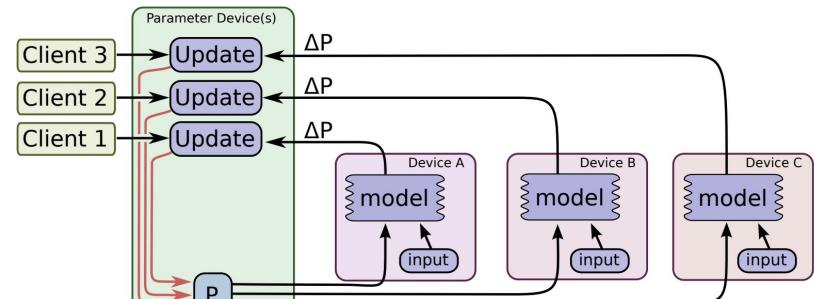
Parameter Server @TensorFlow

TensorFlow:
Large-Scale Machine Learning on Heterogeneous Distributed Systems
(Preliminary White Paper, November 9, 2015)

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dan Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng
Google Research*



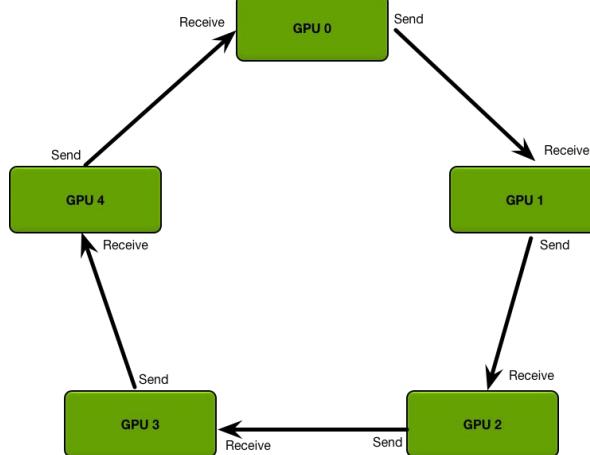
Synchronous Data Parallelism



Asynchronous Data Parallelism

Ring All-Reduce @Baidu

Two steps: First, a scatter-reduce, and then, an allgather.

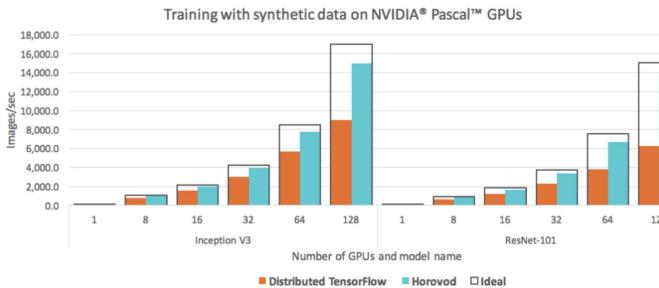


Ring All-Reduce @Horovod

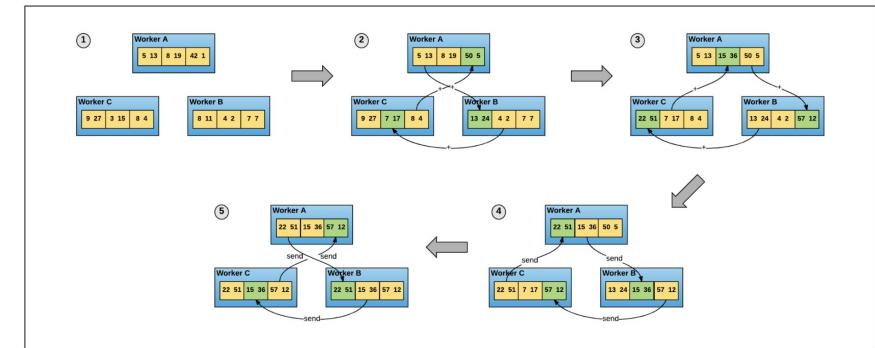
Horovod: fast and easy distributed deep learning in TensorFlow

Alexander Sergeev
Uber Technologies, Inc.
asergeev@uber.com

Mike Del Balso
Uber Technologies, Inc.
mdb@uber.com



In early 2017 Baidu published an article [8] evangelizing a different algorithm for averaging gradients and communicating those gradients to all nodes (Steps 2 and 3 above), called ring-allreduce, as well as a fork of TensorFlow through which they demonstrated a draft implementation of this algorithm. The algorithm was based on the approach introduced in the 2009 paper by Patarasuk and Yuan [9].





TensorFlow

Distributed Training with TensorFlow

The screenshot shows the TensorFlow Core documentation page for 'Distributed training with TensorFlow'. The page includes a sidebar with various TensorFlow API categories like Ragged tensor, Create an op, Random number generation, Data input pipelines, Save a model, Checkpoint, SavedModel, Concrete functions, Accelerators, and more. The main content area has a title 'Distributed training with TensorFlow' with a 'Run in Google Colab' button, a 'View source on GitHub' button, and a 'Download notebook' button. Below this is an 'Overview' section with a paragraph about the `tf.distribute.Strategy` API and a bulleted list of key goals. To the right is a sidebar with links to 'Contents' (Overview, Types of Strategies, MirroredStrategy, CentralStorageStrategy, MultiWorkerMirroredStrategy, TPUStrategy, ParameterServerStrategy, OneDeviceStrategy), 'Using' (tf.distribute.Strategy with Keras, What's supported now), 'Examples and Tutorials' (Using tf.distribute.Strategy with custom training loops, What's supported now), and 'API Reference' (tf.distribute.Strategy with Estimator (Limited support), What's supported now).

Training API	MirroredStrategy	TPUStrategy	MultiWorkerMirroredStrategy	CentralStorageStrategy	ParameterServerStrategy	OneDeviceStrategy
Keras API	Supported	Experimental support	Experimental support	Experimental support	Supported planned post 2.0	Supported
Custom training loop	Experimental support	Experimental support	Support planned post 2.0	Support planned post 2.0	No support yet	Supported
Estimator API	Limited Support	Not supported	Limited Support	Limited Support	Limited Support	Limited Support

Distributed Training with TensorFlow

The screenshot shows the TensorFlow Core Guide page for 'Distributed training with TensorFlow'. The page has a sidebar with various links like Overview, Tutorials, and Performance. The main content area has a title 'Distributed training with TensorFlow' with a star rating of 4.5. It includes sections for 'Overview', 'tf.distribute.Strategy', and 'What's supported now?'. A sidebar on the right lists 'Contents' such as Overview, Types of Strategies, MirroredStrategy, MultiWorkerMirroredStrategy, TPUStrategy, ParameterServerStrategy, OneDeviceStrategy, Using tf.distribute.Strategy with Keras, What's supported now!, Examples and Tutorials, Using tf.distribute.Strategy with custom training loops, and What's supported now! again.

Training API	MirroredStrategy	TPUStrategy	MultiWorkerMirroredStrategy	CentralStorageStrategy	ParameterServerStrategy	OneDeviceStrategy
Keras API	Supported	Experimental support	Experimental support	Experimental support	Supported (planned post 2.0)	Supported
Custom training loop	Experimental support	Experimental support	Support planned post 2.0	Support planned post 2.0	No support yet	Supported
Estimator API	Limited Support	Not supported	Limited Support	Limited Support	Limited Support	Limited Support

Configuration File



```
# run in main worker
Sys.setenv(TF_CONFIG = jsonlite::toJSON(list(
  cluster = list(
    worker = c("172.31.11.122:10090", "172.31.10.143:10088", "172.31.4.119:10087", "172.31.4.116:10089")
  ),
  task = list(type = 'worker', index = 0)
), auto_unbox = TRUE))

# run in worker(1)
Sys.setenv(TF_CONFIG = jsonlite::toJSON(list(
  cluster = list(
    worker = c("172.31.11.122:10090", "172.31.10.143:10088", "172.31.4.119:10087", "172.31.4.116:10089")
  ),
  task = list(type = 'worker', index = 1)
), auto_unbox = TRUE))

# run in worker(2)
Sys.setenv(TF_CONFIG = jsonlite::toJSON(list(
  cluster = list(
    worker = c("172.31.11.122:10090", "172.31.10.143:10088", "172.31.4.119:10087", "172.31.4.116:10089")
  ),
  task = list(type = 'worker', index = 2)
), auto_unbox = TRUE))

# run in worker(3)
Sys.setenv(TF_CONFIG = jsonlite::toJSON(list(
  cluster = list(
    worker = c("172.31.11.122:10090", "172.31.10.143:10088", "172.31.4.119:10087", "172.31.4.116:10089")
  ),
  task = list(type = 'worker', index = 3)
), auto_unbox = TRUE))
```

Distributed Pseudocode

Local Model

```
library(tensorflow)
library(keras)

# create model
model <- keras_model_sequential() # %>% ...

# compile model
model %>% compile()

# fit model
model %>% fit()
```

Distributed Model

```
# define configuration
Sys.setenv(TF_CONFIG = "")

library(tensorflow)
library(keras)

# define strategy
strategy <- tf$distribute$experimental$MultiWorkerMirroredStrategy()
with (strategy$scope(), {
  # create model
  model <- keras_model_sequential() # %>% ...

  # compile model
  model %>% compile()
})

# fit model
model %>% fit()
```

Distributed Code

Local Model

```
library(tensorflow)
library(keras)

batch_size <- 64L

mnist <- dataset_mnist()
x_train <- mnist$train$x
y_train <- mnist$train$y

x_train <- array_reshape(x_train, c(nrow(x_train), 28, 28, 1))
x_train <- x_train / 255

model <- keras_model_sequential() %>%
  layer_conv_2d(
    filters = 32,
    kernel_size = 3,
    activation = 'relu',
    input_shape = c(28, 28, 1)
) %>%
  layer_max_pooling_2d() %>%
  layer_flatten() %>%
  layer_dense(units = 64, activation = 'relu') %>%
  layer_dense(units = 10)

model %>% compile(
  loss = tf$keras$layers$SparseCategoricalCrossentropy(from_logits = TRUE),
  optimizer = tf$keras$optimizers$SGD(learning_rate = 0.001),
  metrics = 'accuracy')

model %>% fit(x_train, y_train, batch_size = batch_size, epochs = 3, steps_per_epoch = 5)
```

Distributed Model

```
# run in main worker
Sys.setenv(TF_CONFIG = jsonlite::toJSON(list(
  cluster = list(
    worker = c("172.31.11.122:10090", "172.31.10.143:10088", "172.31.4.119:10087", "172.31.4.116:10089")
  ),
  task = list(type = 'worker', index = 0)
), auto_unbox = TRUE))
```



```
library(tensorflow)
library(keras)

strategy <- tf$distribute$experimental$MultiWorkerMirroredStrategy()

num_workers <- 4L
batch_size <- 64L * num_workers

mnist <- dataset_mnist()
x_train <- mnist$train$x
y_train <- mnist$train$y

x_train <- array_reshape(x_train, c(nrow(x_train), 28, 28, 1))
x_train <- x_train / 255

with (strategy$scope(), {
  model <- keras_model_sequential() %>%
    layer_conv_2d(
      filters = 32,
      kernel_size = 3,
      activation = 'relu',
      input_shape = c(28, 28, 1)
) %>%
    layer_max_pooling_2d() %>%
    layer_flatten() %>%
    layer_dense(units = 64, activation = 'relu') %>%
    layer_dense(units = 10)

  model %>% compile(
    loss = tf$keras$layers$SparseCategoricalCrossentropy(from_logits = TRUE),
    optimizer = tf$keras$optimizers$SGD(learning_rate = 0.001),
    metrics = 'accuracy')
})

model %>% fit(x_train, y_train, batch_size = batch_size, epochs = 3, steps_per_epoch = 5)
```

Distributed Training

SSH to each machine, transfer **data subset**, run the code in each machine.

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

The terminal shows a single machine's login and command execution. It starts with a standard Mac OS X login message, followed by a prompt to update the shell to zsh. Finally, the user runs the command 'ssh' to start a session.

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

The terminal shows a single machine's login and command execution. It starts with a standard Mac OS X login message, followed by a prompt to update the shell to zsh. Finally, the user runs the command 'ssh' to start a session.

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

The terminal shows a single machine's login and command execution. It starts with a standard Mac OS X login message, followed by a prompt to update the shell to zsh. Finally, the user runs the command 'ssh' to start a session.

What if you need dozens of machines?

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

The terminal shows five separate sessions of a user logging in to different machines and running the 'ssh' command. Each session follows the same pattern: a standard Mac OS X login message, a shell update prompt, and the execution of the 'ssh' command.

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

The terminal shows five separate sessions of a user logging in to different machines and running the 'ssh' command. Each session follows the same pattern: a standard Mac OS X login message, a shell update prompt, and the execution of the 'ssh' command.

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

```
Last login: Mon May 11 22:37:47 on ttys000
The default interactive shell is now zsh.
To update your account to use zsh, please run 'chsh -s /bin/zsh'.
For more details, please visit https://support.apple.com/kb/HT208050.
Javiers-MacBook-Pro:~ javierluraschi$ ssh
```

The terminal shows five separate sessions of a user logging in to different machines and running the 'ssh' command. Each session follows the same pattern: a standard Mac OS X login message, a shell update prompt, and the execution of the 'ssh' command.

A vertical strip on the left side of the slide is composed of numerous small, square images arranged in a grid. These images are highly varied in content, including what appears to be a collection of logos, icons, and other abstract or specific symbols. The colors range from dark to light across the different squares.

Spark

Deep Learning Frameworks and Spark

Explosion of ML Frameworks

Azure



Prep data



Train models



Execution Models

Spark

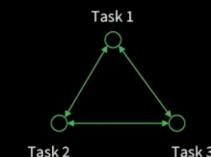
Tasks are independent of each other



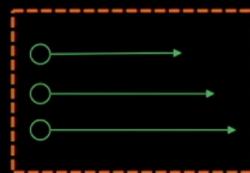
Distributed ML Frameworks

Complete coordination among tasks

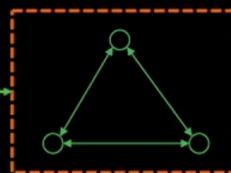
Optimized for communication



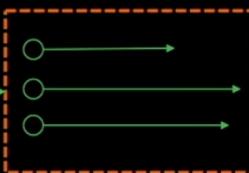
Stage 1
data prep
embarrassingly parallel



Stage 2
distributed ML training
gang scheduled

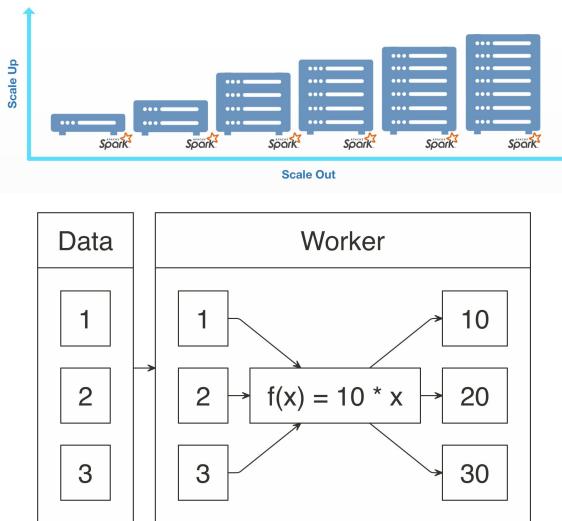


Stage 3
data sink
embarrassingly parallel



Spark Apply

You can use **spark_apply()** to apply an arbitrary transformation in R:



```
sdf_len(sc, 3) %>%
  spark_apply(~ 10 * .x)
```

```
# Source: spark<?> [?? x 1]
  id
  * <dbl>
  1    10
  2    20
  3    30
```

Can't we just use **spark_apply()** to run TensorFlow code?

Barrier Execution

Barrier Execution supported since sparklyr 1.1 and Spark 2.4. Used to schedule all-or-nothing resources, retrieve IP addresses and process data.

Barrier Execution

Barrier execution is a new feature introduced in [Spark 2.4](#) which enables Deep Learning on Apache Spark by implementing an all-or-nothing scheduler into Apache Spark. This allows Spark to not only process analytic workflows, but also to use Spark as a high-performance computing cluster where other framework, like [OpenMP](#) or [TensorFlow Distributed](#), can reuse cluster machines and have them directly communicate with each other for a given task.

In general, we don't expect most users to use this feature directly; instead, this is a feature relevant to advanced users interested in creating extensions that support additional modeling frameworks. You can learn more about barrier execution in Reynold Xin's [keynote](#).

To use barrier execution from R, set the `barrier = TRUE` parameter in `spark_apply()` and then make use of a new parameter in the R closure to retrieve the network address of the additional nodes available for this task. A simple example follows:

```
library(sparklyr)
sc <- spark_connect(master = "local", version = "2.4")

sdf_len(sc, 1, repartition = 1) %>%
  spark_apply(~ .y$address, barrier = TRUE, columns = c(address = "character")) %>%
  collect()
```

```
# A tibble: 1 x 1
  address
  <chr>
1 localhost:50693
```

Spark and TensorFlow Pseudocode

```
library(sparklyr)
sc <- spark_connect(master = "local|yarn|etc")

# partition dataset
sdf_len(sc, 3, repartition = 3) %>%
  spark_apply(function(df, barrier) {
    library(tensorflow)
    library(keras)

    # define configuration from barrier
    Sys.setenv(TF_CONFIG = "")

    # define strategy and model
    strategy <- MultiWorkerMirroredStrategy()
    with (strategy$scope(), {
      model <- keras_model_sequential() # %>% ...
      model %>% compile()
    })

    # fit and retrieve model
    model %>% fit()
  }, barrier = TRUE) %>% collect()
```

- Connect
- Partition Data
- Apply Transform
- Load Libraries
- Set Configuration
- Define Strategy
- Define Model
- Compile Model
- Fit Model
- Retrieve Model

Example (1/2)

```
library(sparklyr)
sc <- spark_connect(master = "yarn", spark_home = "/usr/lib/spark/", config =
list(spark.dynamicAllocation.enabled= FALSE, `sparklyr.shell.executorcores` = 8,
`sparklyr.shell.numexecutors` = 3, sparklyr.apply.env.WORKON_HOME = "/tmp/.virtualenvs"))

sdf_len(sc, 3, repartition = 3) %>%
  spark_apply(function(df, barrier) {
    tryCatch({
      library(tensorflow)
      library(keras)

      Sys.setenv(TF_CONFIG = jsonlite:::toJSON(list(
        cluster = list(worker = paste(gsub(":[0-9]+$", "", barrier$address), 8000 +
seq_along(barrier$address), sep = ":")),
        task = list(type = 'worker', index = barrier$partition)
      ), auto_unbox = TRUE))

      if (is.null(tf_version())) install_tensorflow()
      // << tensorflow model >>
    }, error = function(e) e$message)
  }, barrier = TRUE, columns = c(address = "character"), ) %>% collect()
```

Example (1/2)

```
strategy <- tf$distribute$experimental$MultiWorkerMirroredStrategy()
mnist <- dataset_mnist()

x_train <- mnist$train$x
y_train <- mnist$train$y
x_train <- array_reshape(x_train, c(nrow(x_train), 28, 28, 1))
x_train <- x_train / 255

with (strategy$scope(), {
  model <- keras_model_sequential() %>%
    layer_conv_2d(filters = 32, kernel_size = 3, activation = 'relu', input_shape = c(28, 28, 1)) %>%
    layer_max_pooling_2d() %>%
    layer_flatten() %>%
    layer_dense(units = 64, activation = 'relu') %>%
    layer_dense(units = 10)

  model %>% compile(
    loss = tf$keras$losses$SparseCategoricalCrossentropy(from_logits = TRUE),
    optimizer = tf$keras$optimizers$SGD(learning_rate = 0.001),
    metrics = 'accuracy') }

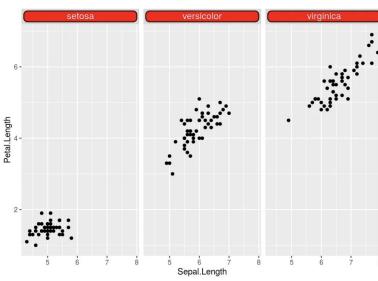
model %>% fit(x_train, y_train, batch_size = 64L * 3L, epochs = 3, steps_per_epoch = 5)
```



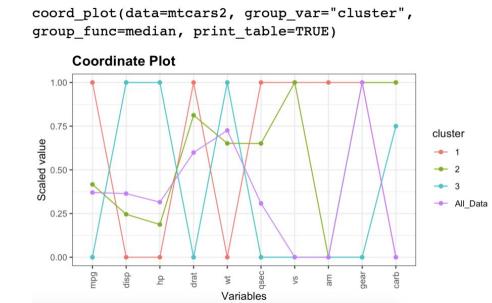
Pins

Data Workflows - Today

June 16th posts in R-bloggers use iris, mtcars, have missing data paths and require manually downloading datasets.



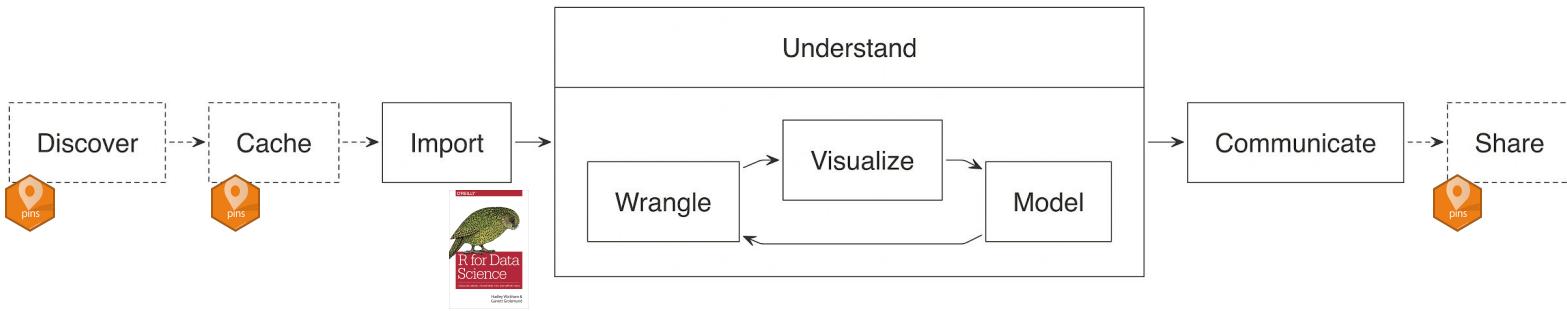
```
#####
### IMPORT RAW DATA
log_info("Loading data")
mydat = fread(data_path)
```



```
# https://www.data.gouv.fr/fr/datasets/donnees-hospitalieres-relatives-a-lepidemie-de-covid-19/fichier_covid <- "donnees/covid.csv"
```

Data Workflows - In Data Science

We know from R for Data Science that in a typical project we usually need to import, tidy, understand and communicate knowledge.



However, it is often also required for us to discover which dataset to use, cache it locally, and share our datasets with colleagues.

Data Workflows - Reproducibility

But more importantly, properly caching and sharing our datasets allows us to make Data Science more reproducible in tools like R Markdown and Jupyter.

The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Try it in your browser Install the Notebook

Language of choice
Jupyter supports over 40 programming languages, including Python, R, Julia, and Scala.

Share notebooks
Notebooks can be shared with others using email, Dropbox, GitHub and the Jupyter Notebook Viewer.

Interactive output
Your code can produce rich, interactive output: HTML, images, videos, LaTeX, and custom MIME types.

Big data integration
Leverage big data tools, such as Apache Spark, from Python, R and Scala. Explore that same data with pandas, scikit-learn, ggplot2, TensorFlow,

R Markdown documents are fully reproducible. Use a productive notebook interface to weave together narrative text and code to produce elegantly formatted output. Use multiple languages including R, Python, and SQL.

Analyze. Share. Reproduce.
Your data tells a story. Tell it with R Markdown. Turn your analyses into high quality documents, reports, presentations and dashboards.

Data Workflows - Ideally



- Learn a **single tool** regardless of where data lives.
- Easy **switch storage** and share it anywhere.
- Easy to build **automated workflows**.
- Can share datasets of **any size**.
- Can easily use **interesting datasets!**
- Data loads **fast** and works **offline**.

What is the pins package?



- **Pin** remote resources locally with `pin()`, work offline and cache results.
- **Discover** new resources across different boards using `pin_find()`.
- **Share resources** in local folders, GitHub, Kaggle, and RStudio Connect by registering new boards with `board_register()`.

Links

Download from CRAN at
<https://cloud.r-project.org/package=pins>

Browse source code at
<https://github.com/rstudio/pins>

Report a bug at
<https://github.com/rstudio/pins/issues>

License

Apache License 2.0

Developers

Javier Luraschi
Author, maintainer

[All authors...](#)

Dev status

build passing

CRAN 0.4.1

codecov 77%

downloads 3954/month

lifecycle maturing

chat on gitter

stars 107

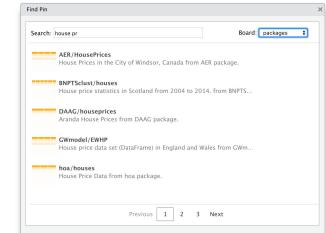
How can I use Pins?

Use `pin()` to save objects, `pin_get()` to retrieve them and `pin_find()` to search.

```
pin(1:10, name = "numbers")
pin_get("numbers")

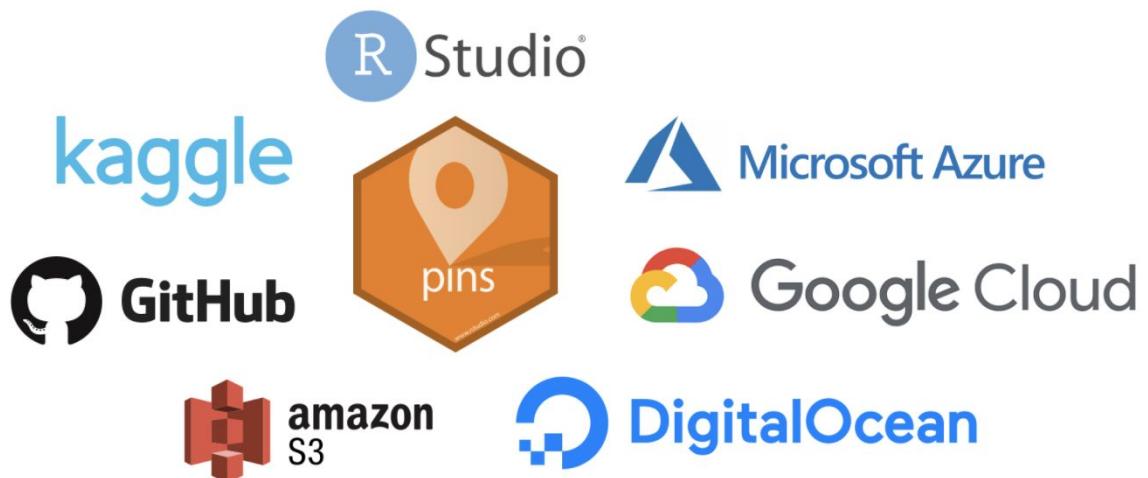
pin("http://cs231n.stanford.edu/tiny-imagenet-200.zip ", name = "tinyimagenet")
pin_get("tinyimagenet")

pin_find()
```



What are Boards?

A board is a storage location, like your local hard drive, but there are also many additional places where you can store your datasets like RStudio Connect, Microsoft Azure, Google Cloud, Digital Ocean, Amazon S3, GitHub and Kaggle.



What are Boards?

To use pins with boards, first register the board, then specify the board in `pin()`.

```
board_register("_____")  
  
pin(1:10, name = "numbers", board = "_____")  
pin_get("numbers")  
  
pin("cs231n.stanford.edu/tiny-imagenet-200.zip ", "tinyimagenet", board = "_____")  
pin_get("tinyimagenet")  
  
pin_find()
```



ImageNet

Retrieving ImageNet

Use pins and the Kaggle board to fetch ImageNet's 250GB dataset.

```
library(pins)
board_register("kaggle", "path/to/kaggle.json")

imagenet <- pin_get("c/imagenet-object-localization-challenge" , board = "kaggle")
untar(imagenet[1], exdir = "imagenet/")
```

Which takes 60 minutes to download and 60 minutes to uncompress...

Using Local SSDs

n1-standard-8 (8 vCPUs, 30 GB memory) and 4 local SSDs

```
> system.time(pins::pin_get("imagenet", board = "https://storage.googleapis.com/imagenet-pins/"))
  user  system elapsed
211.937 512.572 1847.234
.
.
.
> system.time(untar(imagenet[1], exdir = "/localssd/imagenet/"))
  user  system elapsed
1273.879 443.763 1360.702
```

Down to 30 minutes to download and 22 to uncompress...

More and Faster CPUs

c2-standard-16 (16 vCPUs, 64 GB memory)

```
> system.time(pins::pin_get("imagenet", board = "https://storage.googleapis.com/imagenet-pins/"))
  user  system elapsed
147.031 387.707 1522.945

> system.time(untar(imagenet[1], exdir = "/localssd/imagenet/"))
  user  system elapsed
887.538 284.525 961.746
```

Down to 25 minutes to download and 16 minutes to uncompress...

CPU Bottleneck? (top)

Use the **top** command to display CPU usage to diagnose bottlenecks.

```
top - 22:48:19 up 9 min, 0 users, load average: 0.23, 0.63, 0.46
Tasks: 9 total, 1 running, 8 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.7 us, 0.9 sy, 0.0 ni, 98.0 id, 0.1 wa, 0.0 hi, 0.3 si, 0.0 st
KiB Mem : 65972836 total, 50130140 free, 1029696 used, 14813000 buff/cache
KiB Swap: 0 total, 0 free, 0 used. 64238656 avail Mem
```

PID	USER	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
140	rstudio	20	0	1544424	228464	59368	S	28.7	0.3	0:25.61	rsession
3479	root	20	0	250608	5716	5292	S	0.3	0.0	0:00.22	iftop
1	root	20	0	20048	3436	3180	S	0.0	0.0	0:00.17	bash
7	root	20	0	6220	852	788	S	0.0	0.0	0:00.00	tail
16	rstudio+	20	0	268368	52276	8712	S	0.0	0.1	0:00.57	rserver
176	rstudio	20	0	21364	5076	3436	S	0.0	0.0	0:00.01	bash
3478	root	20	0	53676	3948	3512	S	0.0	0.0	0:00.00	sudo
3516	rstudio	20	0	21364	4924	3380	S	0.0	0.0	0:00.01	bash
3519	rstudio	20	0	38264	3596	3168	R	0.0	0.0	0:00.00	top

Network Bottleneck? (iftop)

Or iftop to troubleshoot network bottlenecks:

	373Mb	745Mb	1.09Gb	1.46Gb
imagenet-1.c.rstudio-cloudml.internal	<=	= il-in-f128.1e100.net	904Kb	1.29Mb
imagenet-1.c.rstudio-cloudml.internal	<=	=> metadata.google.internal	285Mb	393Mb
imagenet-1.c.rstudio-cloudml.internal	<=	=> c-66-235-7-252.sea.wa.customer.broadstripe	6.57Kb	6.60Kb
imagenet-1.c.rstudio-cloudml.internal	<=	=> 35.235.241.18	32.6Kb	32.7Kb
imagenet-1.c.rstudio-cloudml.internal	<=	=> 193.142.146.204	1.30Kb	2.17Kb
imagenet-1.c.rstudio-cloudml.internal	<=	=> 122.155.212.244	208b	1.26Kb
				552b
			0b	42b
			0b	42b
			0b	8b
			0b	0b
			0b	16b
			0b	8b
			0b	8b

TX:	cum:	71.6MB	peak:	2.62Mb			
RX:	2.11GB	890Mb	25.8GB	845Mb			
TOTAL:				847Mb	rates:	912Kb	1.30Mb

Disk Bottleneck? (dstat)

Or dstat to diagnose disk performance:

--total-cpu-usage-- -dsk/total- -net/total- ---paging-- --system--												
usr	sys	idl	wai	stl	read	writl	recv	sendl	in	out	int	csw
1	1	98	0	0	0	32M	141M	715kl	0	0	14k	3477
1	2	97	0	0	0	34M	190M	1004kl	0	0	20k	5884
1	1	98	0	0	0	32M	148M	778kl	0	0	15k	3734
1	1	98	0	0	0	32M	110M	583kl	0	0	12k	3135
1	1	98	0	0	0	34M	137M	720kl	0	0	14k	3560
1	1	98	0	0	0	32M	105M	567kl	0	0	11k	3174
1	1	98	0	0	0	34M	145M	780kl	0	0	16k	4376
1	1	98	0	0	0	32M	129M	696kl	0	0	14k	3775
1	3	92	5	0	0	1171M	143M	745kl	0	0	16k	3111
1	2	89	8	0	4096B	1542M	139M	718kl	0	0	15k	2048
1	2	92	5	0	4096B	1519M	80M	446kl	0	0	11k	3267
1	1	97	0	0	0	220M	142M	738kl	0	0	14k	2376
1	2	97	0	0	0	34M	190M	993kl	0	0	19k	3915
1	1	98	0	0	0	32M	121M	621kl	0	0	12k	2530
1	1	98	0	0	0	34M	144M	748kl	0	0	15k	2938
1	1	98	0	0	0	32M	123M	641kl	0	0	13k	2715
1	1	98	0	0	0	32M	153M	800kl	0	0	15k	3373
1	1	97	0	0	0	34M	160M	816kl	0	0	16k	3114
1	1	98	0	0	0	32M	123M	619kl	0	0	12k	1915
1	1	99	0	0	0	34M	89M	455kl	0	0	18891	1474
0	1	99	0	0	0	32M	81M	412kl	0	0	18171	1678

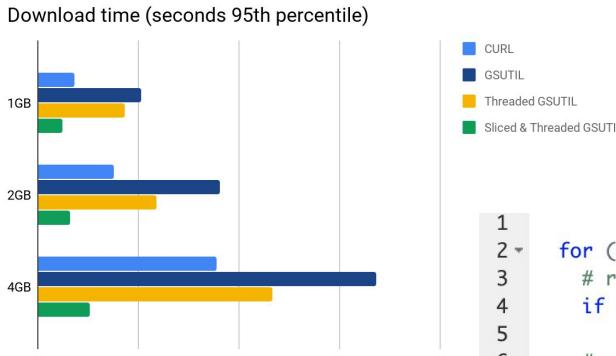
No Bottleneck?

Not the CPU, not the disk, not the network...



So now what?

- a. Use faster tools (gsutil parallel, [lbzip2](#)).
- b. Partition datasets.



```
1  for (path in dir("/localssd/imagenet/ILSVRC/Data/CLS-LOC/train/", full.names = TRUE)) {  
2    # re-register board every 10 uploads to refresh authentication headers  
3    if (runif(1) > 0.9) board_register("gcloud", name = "imagenet", bucket = "r-imagenet")  
4  
5    # upload imagenet partition  
6    dir(path, full.names = TRUE) %>% pin(name = basename(path), board = "imagenet", zip = TRUE)  
7  }  
8  
9
```

Partitioned Dataset

Assuming we train with 16 instances, we can test download/extraction with 1/16th of the data:

```
> pins::board_register_datatxt("https://storage.googleapis.com/r-imagenet/", "imagenet")
> categories <- pins::pin_get("categories", board = "imagenet")
> categories <- categories$id[1:(length(categories$id) / 16)]
> system.time{
+   procs <- lapply(categories, function(cat)
+     callr::r_bg(function(cat) {
+       pins::board_register_datatxt("https://storage.googleapis.com/r-imagenet/", "imagenet")
+       pins::pin_get(cat, board = "imagenet", extract = TRUE)
+     }, args = list(cat))
+   )
+   +
+   while (any(sapply(procs, function(p) p$is_alive())))) Sys.sleep(1)
+ }
user  system elapsed
309.800 86.256 58.729
```

Nice improvement from 41 minutes down to 58 seconds.

Dry Run

We can train a subset using the [r-tensorflow/alexnet](#) package

```
pins::board_register_datatxt("https://storage.googleapis.com/r-imagenet/", "imagenet")
data <- list(
  image = unlist(lapply(categories, function(cat) {
    pins::pin_get(cat, board = "imagenet", download = FALSE)
  })),
  category = unlist(lapply(categories, function(cat) {
    rep(cat, length(pins::pin_get(cat, board = "imagenet", download = FALSE)))
  })),
  categories = categories
)
alexnet::alexnet_train(data = data)

      user   system  elapsed
1267.204 360.276 696.639
```

Although, still takes 11.6 minutes to train 1/16th epoch in a Tesla K80 GPU.

Dry Run (Multi GPU)

We can try instead with 4 GPUs,

```
> strategy <- tf$distribute$MirroredStrategy()
> alexnet::alexnet_train(data = data, strategy = strategy)
2020-07-30 03:06:39.958696: W tensorflow/core/framework/op_kernel.cc:1753] OP_REQUIRES failed at nccl_op
s.cc:104 : Internal: NCCL: unhandled system error. Set NCCL_DEBUG=WARN for detail.
2020-07-30 03:06:39.958863: W tensorflow/core/framework/op_kernel.cc:1753] OP_REQUIRES failed at nccl_op
s.cc:104 : Internal: NCCL: unhandled system error. Set NCCL_DEBUG=WARN for detail.
2020-07-30 03:06:39.958921: W tensorflow/core/framework/op_kernel.cc:1753] OP_REQUIRES failed at nccl_op
s.cc:104 : Internal: NCCL: unhandled system error. Set NCCL_DEBUG=WARN for detail.
2020-07-30 03:06:39.958986: W tensorflow/core/framework/op_kernel.cc:1753] OP_REQUIRES failed at nccl_op
s.cc:104 : Internal: NCCL: unhandled system error. Set NCCL_DEBUG=WARN for detail.
```

```
Error in py_call_impl(callable, dots$args, dots$keywords) :
  InternalError: 2 root error(s) found.
  (0) Internal: NCCL: unhandled system error. Set NCCL_DEBUG=WARN for detail.
    [[node NcclAllReduce (defined at /keras/engine/training.py:66) ]]
  (1) Internal: NCCL: unhandled system error. Set NCCL_DEBUG=WARN for detail.
    [[node NcclAllReduce (defined at /keras/engine/training.py:66) ]]
    [[GroupCrossDeviceControlEdges_1/SGD/SGD/update_0/Const/_135]]
0 successful operations.
2 derived errors ignored. [Op:__inference_train_function_10923]
```

[Show Traceback](#)
[Rerun with Debug](#)

Function call stack:
train_function -> train_function

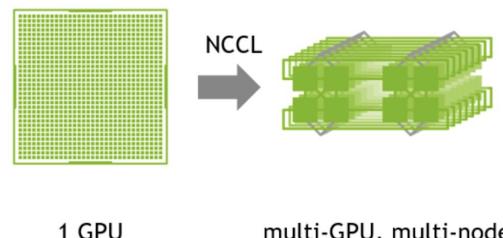
Ughh, NCCL errors, NCCL?

NVIDIA NCCL

NVIDIA NCCL

The NVIDIA Collective Communications Library (NCCL) implements multi-GPU and multi-node collective communication primitives that are performance optimized for NVIDIA GPUs. NCCL provides routines such as all-gather, all-reduce, broadcast, reduce, reduce-scatter, that are optimized to achieve high bandwidth and low latency over PCIe and NVLink high-speed interconnect.

Get Started



Performance

NCCL conveniently removes the need for developers to optimize their applications for specific machines. NCCL provides fast collectives over multiple GPUs both within and across nodes.

Ease of Programming

NCCL uses a simple C API, which can be easily accessed from a variety of programming languages. NCCL closely follows the popular collectives API defined by MPI (Message Passing Interface).

Compatibility

NCCL is compatible with virtually any multi-GPU parallelization model, such as: single-threaded, multi-threaded [using one thread per GPU] and multi-process (MPI combined with multi-threaded operation on GPUs).

Dry Run (Multi GPU)

Or we can just disable NCCL by reducing in one device.

```
> system.time(alexnet::alexnet_train(data = data, strategy = strategy))
2020-07-30 03:44:39.084119: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully
opened dynamic library libcublas.so.10
2020-07-30 03:44:40.058825: I tensorflow/stream_executor/platform/default/dso_loader.cc:44] Successfully
opened dynamic library libcudnn.so.7
Epoch 1/2
2067/2067 [=====] - 375s 181ms/step - accuracy: 0.0383 - loss: nan
Epoch 2/2
2067/2067 [=====] - 373s 181ms/step - accuracy: 0.0197 - loss: nan
      user    system   elapsed
1711.044  334.644  764.518
```

```
+-----+
| NVIDIA-SMI 418.152.00  Driver Version: 418.152.00  CUDA Version: 10.1 |
+-----+
| GPU Name Persistence-MI Bus-Id Disp.A | Volatile Uncorr. ECC |
| Fan Temp Perf Pwr:Usage/Cap| Memory-Usage | GPU-Util Compute M. |
|=====+=====+=====+=====+=====+=====+=====+=====+=====+=====+=====+=====|
| 0 Tesla K80          OFF | 00000000:00:04.0 OFF | 0 |
| N/A 48C P0 79W / 149W | 10936MB / 11441MB | 47% Default |
+-----+
| 1 Tesla K80          OFF | 00000000:00:05.0 OFF | 0 |
| N/A 73C P0 80W / 149W | 10936MB / 11441MB | 27% Default |
+-----+
| 2 Tesla K80          OFF | 00000000:00:06.0 OFF | 0 |
| N/A 74C P0 84W / 149W | 10936MB / 11441MB | 30% Default |
+-----+
| 3 Tesla K80          OFF | 00000000:00:07.0 OFF | 0 |
| N/A 74C P0 86W / 149W | 10936MB / 11441MB | 28% Default |
+-----+
+-----+
| Processes:           GPU Memory |
| GPU     PID  Type Process name        Usage |
|=====+=====+=====+=====+=====+=====+=====+=====+=====+=====+=====+=====|

```

Multi Worker - Multi GPU

After configuring Spark, we can then run something like:

```
1 library(sparklyr)
2 sc <- spark_connect(master = "<master>")
3
4 sdf_len(sc, 3, repartition = 3) %>%
5   spark_apply(function(df, barrier) {
6     tryCatch({
7       library(tensorflow)
8
9       Sys.setenv(TF_CONFIG = jsonlite:::toJSON(list(
10         cluster = list(worker = paste(gsub(":[0-9]+$", "", barrier$address), 8000 + seq_along(barrier$address), sep = ":")),
11         task = list(type = 'worker', index = barrier$partition)
12       ), auto_unbox = TRUE))
13
14       strategy <- tf$distribute$experimental$MultiWorkerMirroredStrategy()
15
16       data <- alexnet::alexnet_imagenet(partition = barrier$partition, partitions = 16)
17       alexnet::alexnet_train(data = data, strategy = strategy, parallel = 6)
18     }, error = function(e) e$message)
19   }, barrier = TRUE, columns = c(address = "character"), ) %>%
20   collect()
```

Thanks!

The screenshot shows the RStudio AI Blog homepage. It features a header with the site name and navigation links for Home, Gallery, About, Contributing, and a search bar. Below the header, there are two blog post cards.

Post 1: Towards privacy: Encrypted deep learning with Syft and Keras
April 28, 2020
A new blog post about encrypted deep learning using Syft and Keras. It includes a small image of a winter landscape and a sidebar for subscribing to the blog.

Post 2: sparklyr 1.2: Foreach, Spark 3.0 and Databricks Connect
April 21, 2020
A new sparklyr release is now available. This sparklyr 1.2 release features new functionalities such as support for DataFrames and DataBricks backends for the foreach package, inter-op improvements for working with Spark 3.0 preview, as well as a number of bug fixes and improvements addressing user-visible pain points.

blogs.rstudio.com/ai

The screenshot shows a YouTube channel page for 'mlverse'. The channel has 495 subscribers. The main navigation bar includes Home, Videos, Playlists (selected), Channels, Discussion, and About. The 'PLAYLISTS' tab is active, showing various playlists like TensorFlow, Spark, Datasets, MLflow, and Visualization, each with a thumbnail and a 'VIEW FULL PLAYLIST' link.

youtube.com/c/mlverse



@zkajdan



@dfalbel



@javierluraschi



@yl790