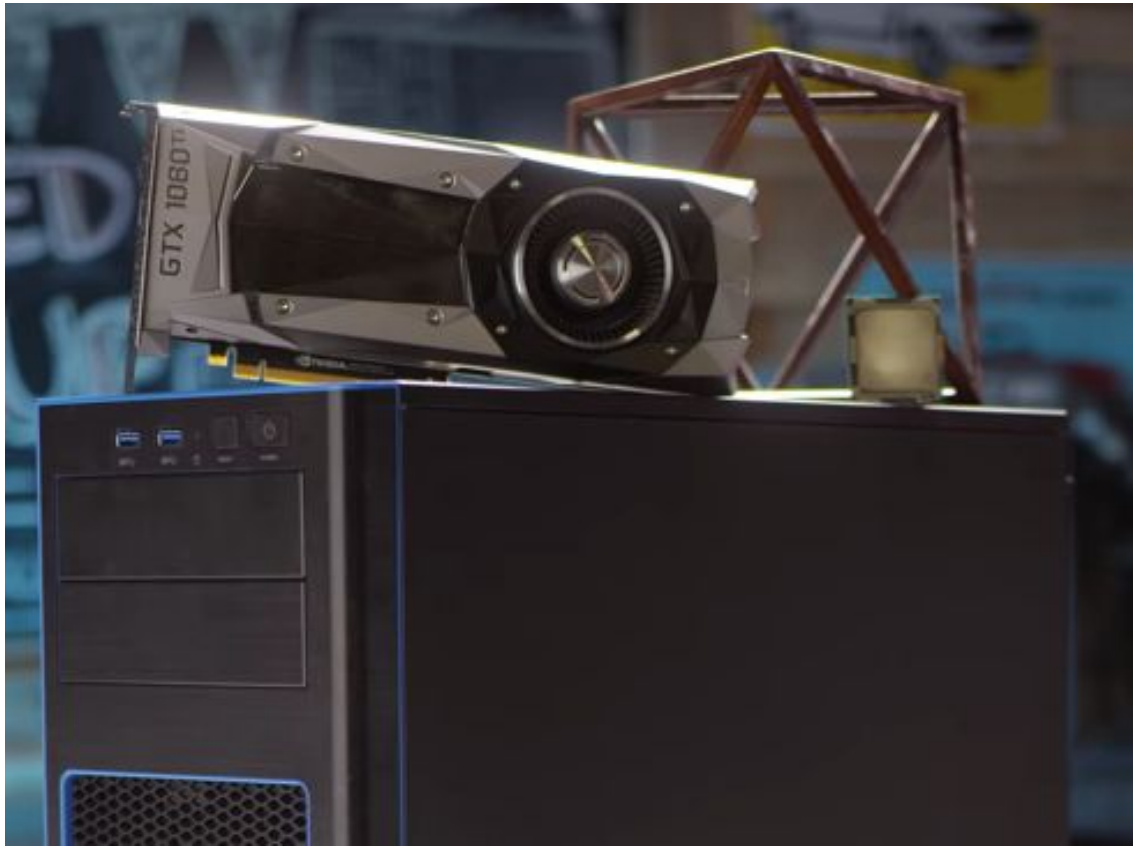
github.com/ernestyalumni/advanced-tensorflow

# Advanced TensorFlow

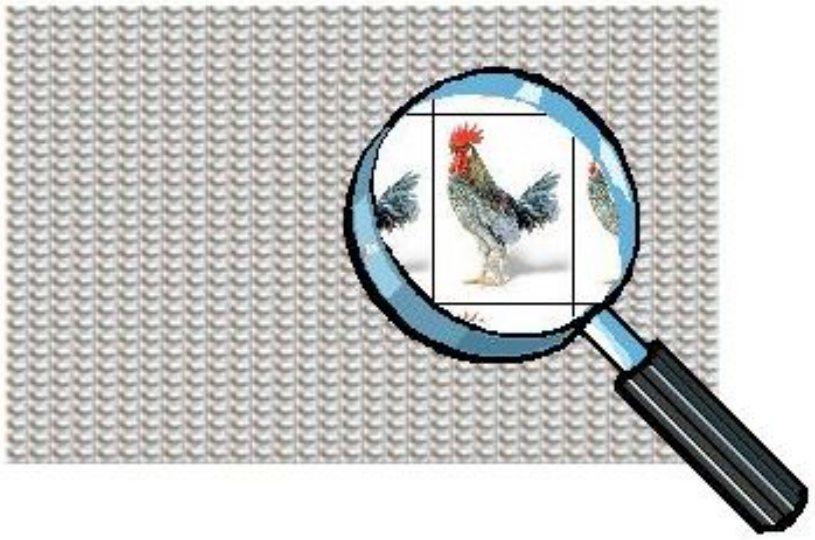
## The Datasets API (and on the GPU(s))

Ernest Yeung
<ernestyalumni@gmail.com>
Twitter, IG, github: @ernestyalumni

# GPU vs. CPU



*If you were plowing a field, which would you rather use: Two strong oxen or 1024 chickens?*
--Attributed to Seymour Cray



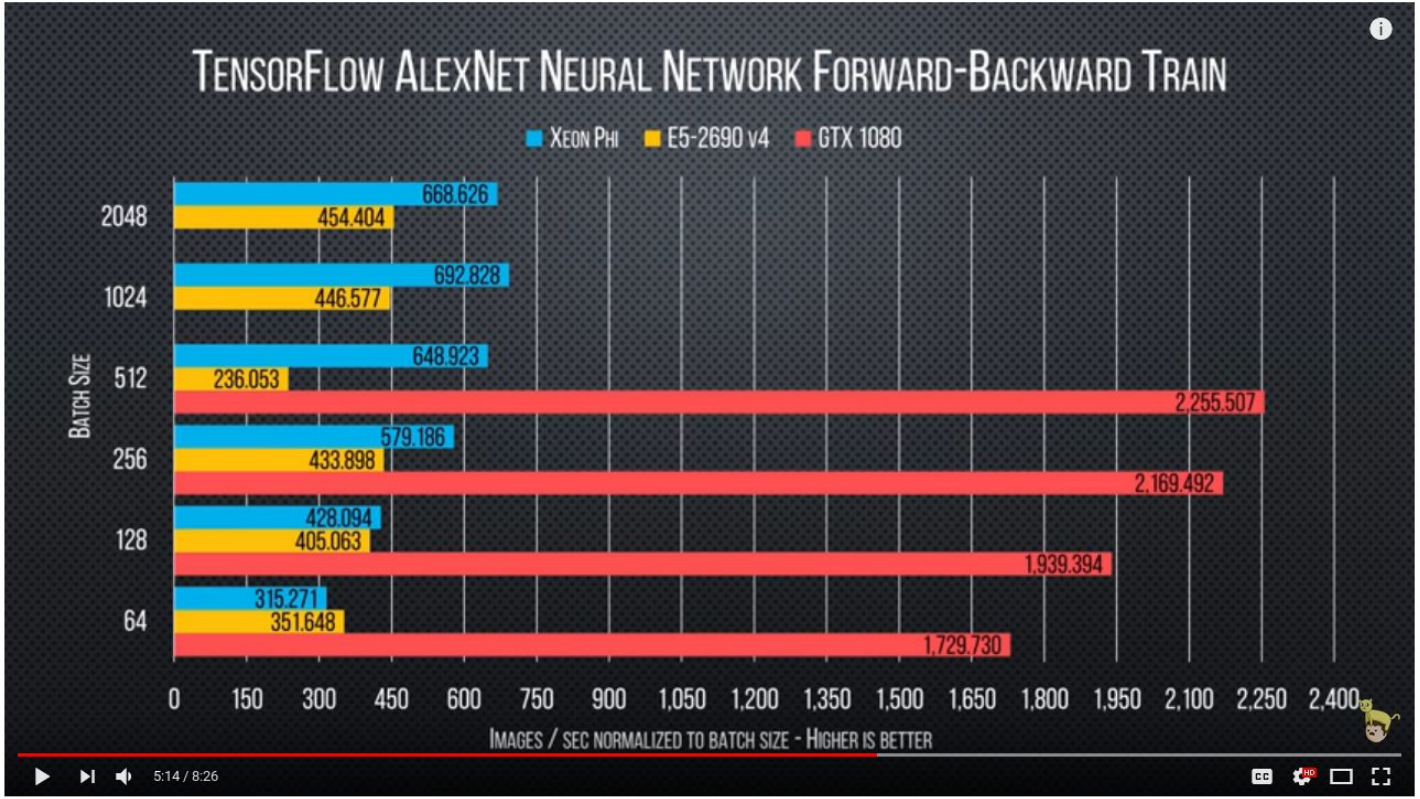
Intel Xeon Phi 7210

- 64 cores
- 215 W
- $1881.00


NVIDIA GeForce GTX 1080Ti

- 3584 CUDA cores
- 250 W
- $769


Linus of Linus Tech Tips
POLO




A REAL 64 Core CPU - For SCIENCE!

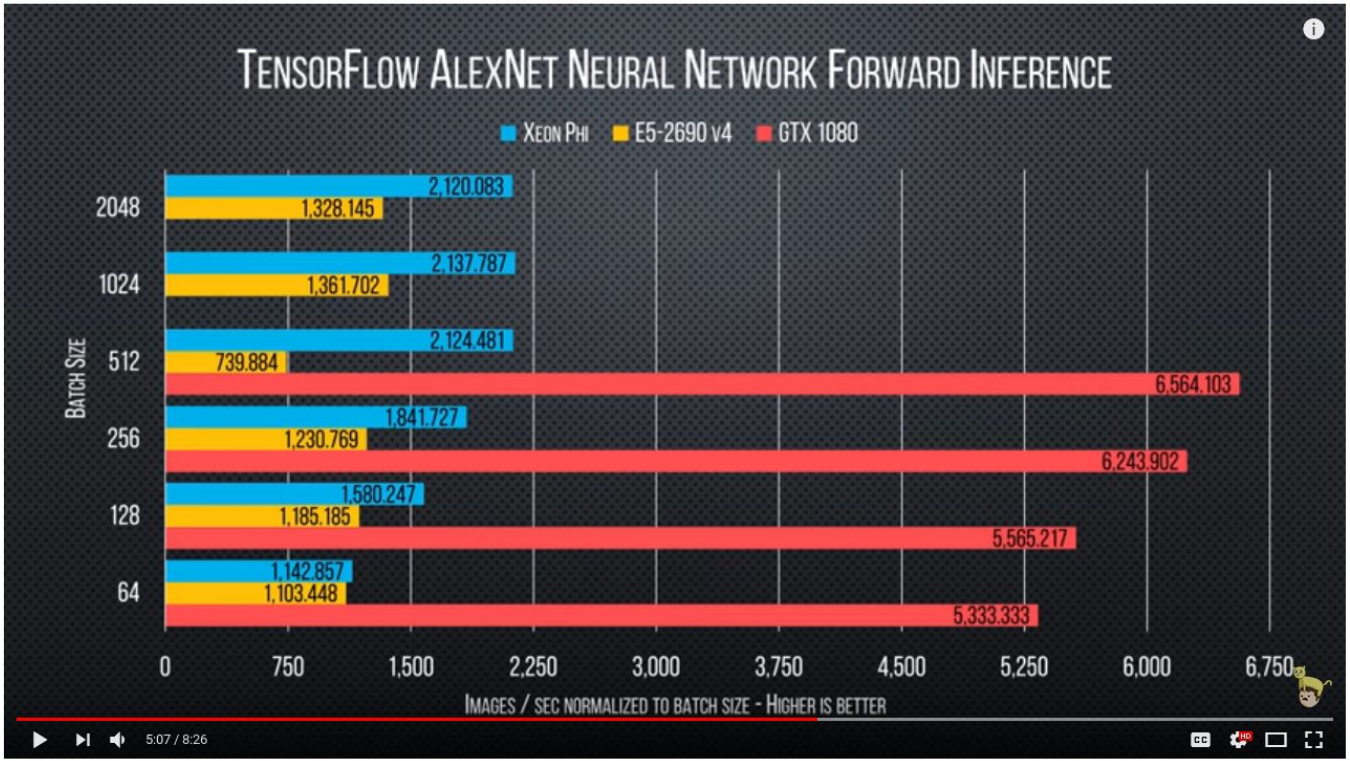
793,648 views

27K 975 SHARE

Linus Tech Tips

Published on Oct 31, 2017

SUBSCRIBE 4.6M




A REAL 64 Core CPU - For SCIENCE!

793,648 views

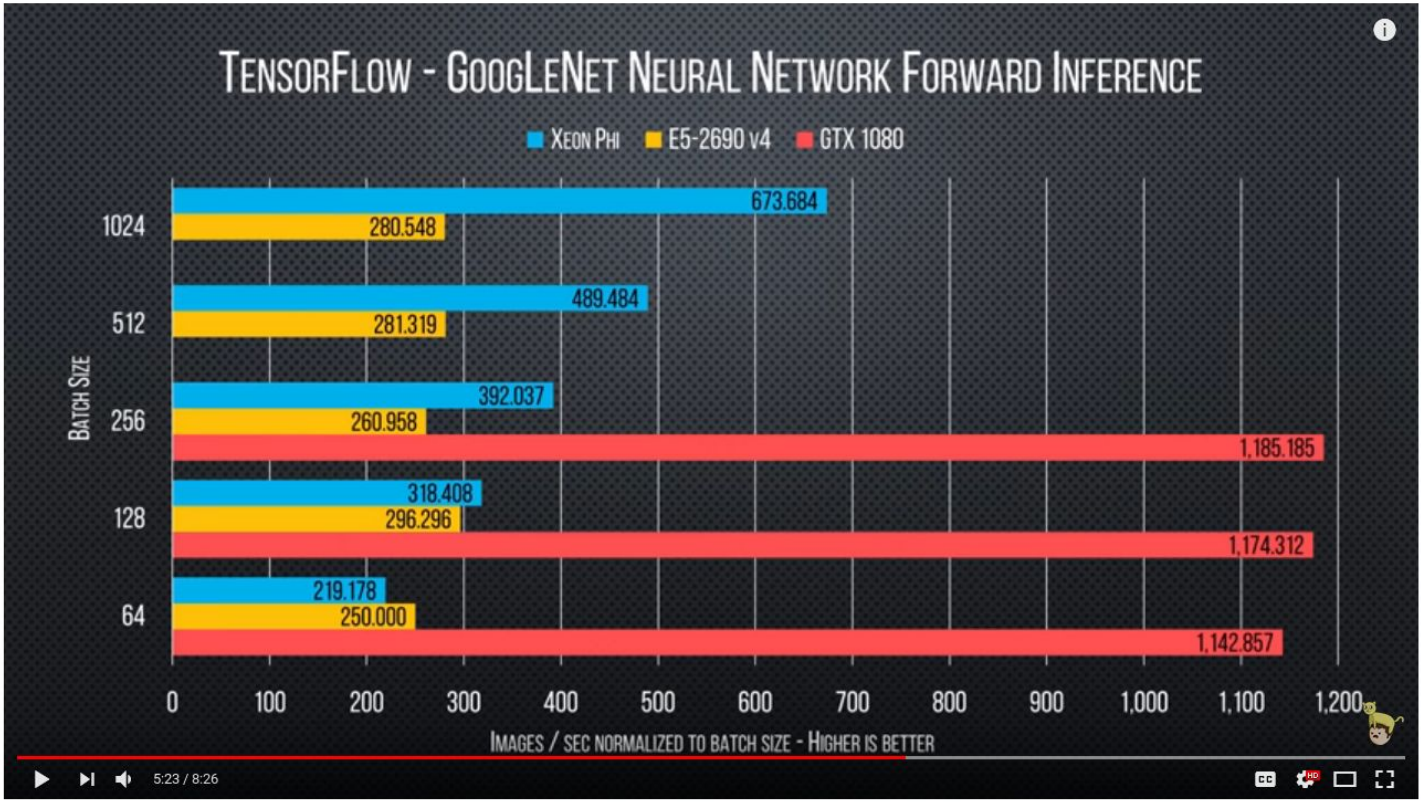
27K 975 SHARE

Linus Tech Tips

Published on Oct 31, 2017

SUBSCRIBE 4.6M

Forget Threadripper and Core i9 - Xeon Phi is the go-to for x86 core count. But how does it actually perform?




A REAL 64 Core CPU - For SCIENCE!

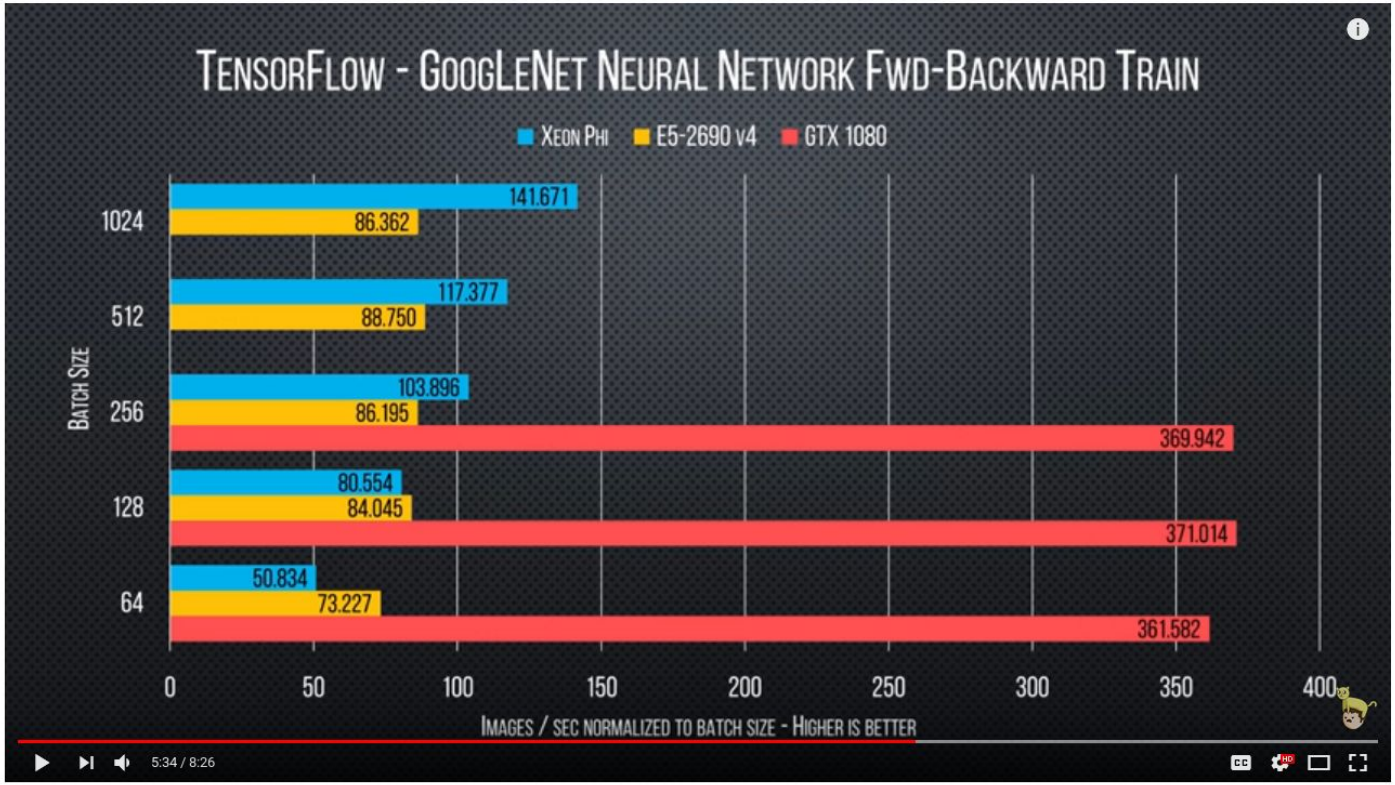
793,648 views

27K 975 SHARE

Linus Tech Tips

Published on Oct 31, 2017

SUBSCRIBE 4.6M




A REAL 64 Core CPU - For SCIENCE!

793,648 views

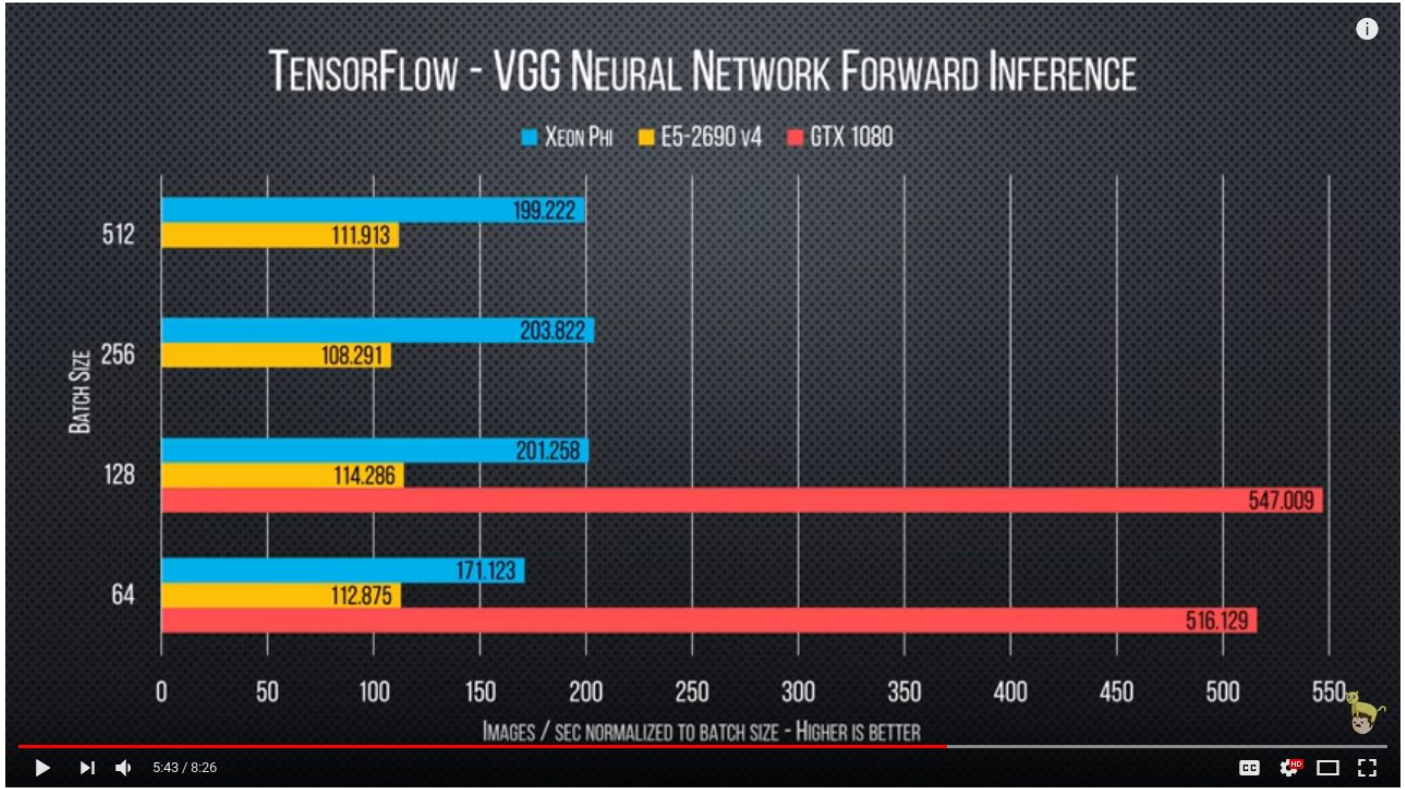
27K 975 SHARE

Linus Tech Tips

Published on Oct 31, 2017

SUBSCRIBE 4.6M

Forget Threadripper and Core i9 - Xeon Phi is the go-to for x86 core count. But how does it actually




A REAL 64 Core CPU - For SCIENCE!

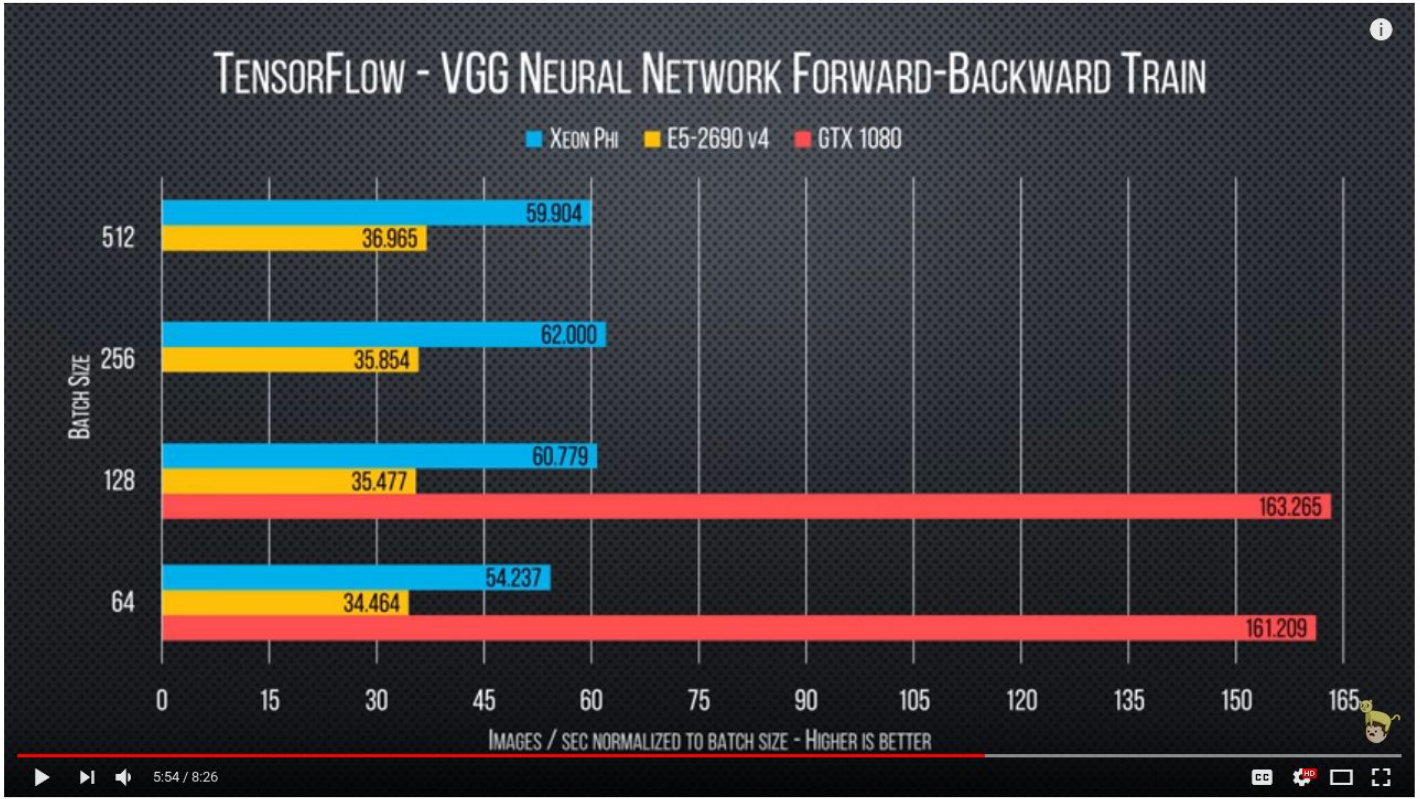
793,648 views

27K 975 SHARE

Linus Tech Tips

Published on Oct 31, 2017

SUBSCRIBE 4.6M




A REAL 64 Core CPU - For SCIENCE!

793,648 views

27K 975 SHARE

Linus Tech Tips

Published on Oct 31, 2017

SUBSCRIBE 4.6M

Data
(X,y)

# Data
# (X,y)

So that we can train models to predict y from X, i.e.

X -> y

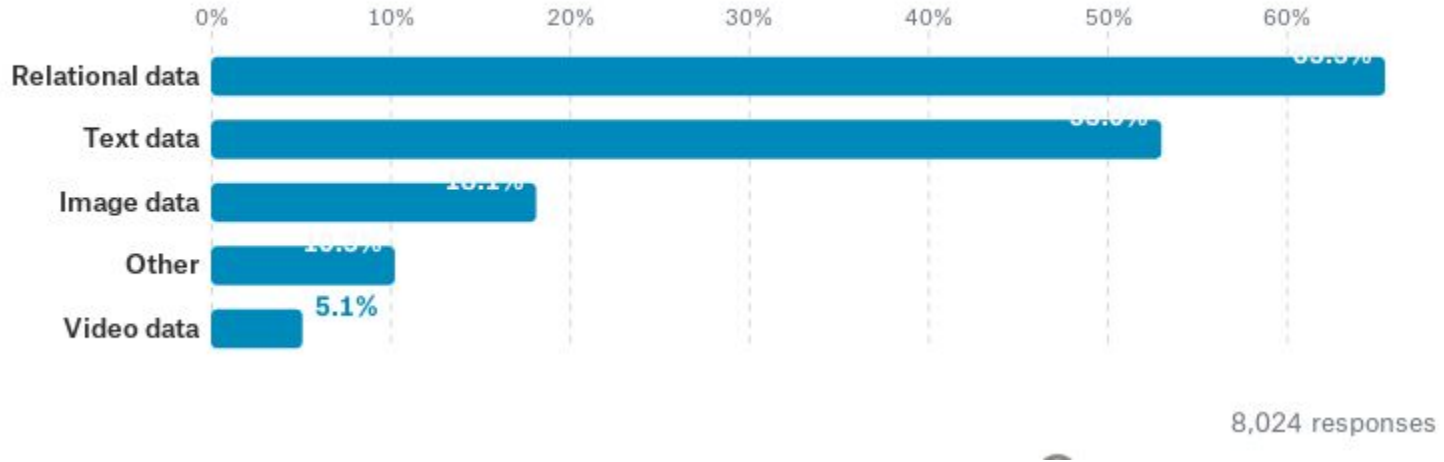
kaggle

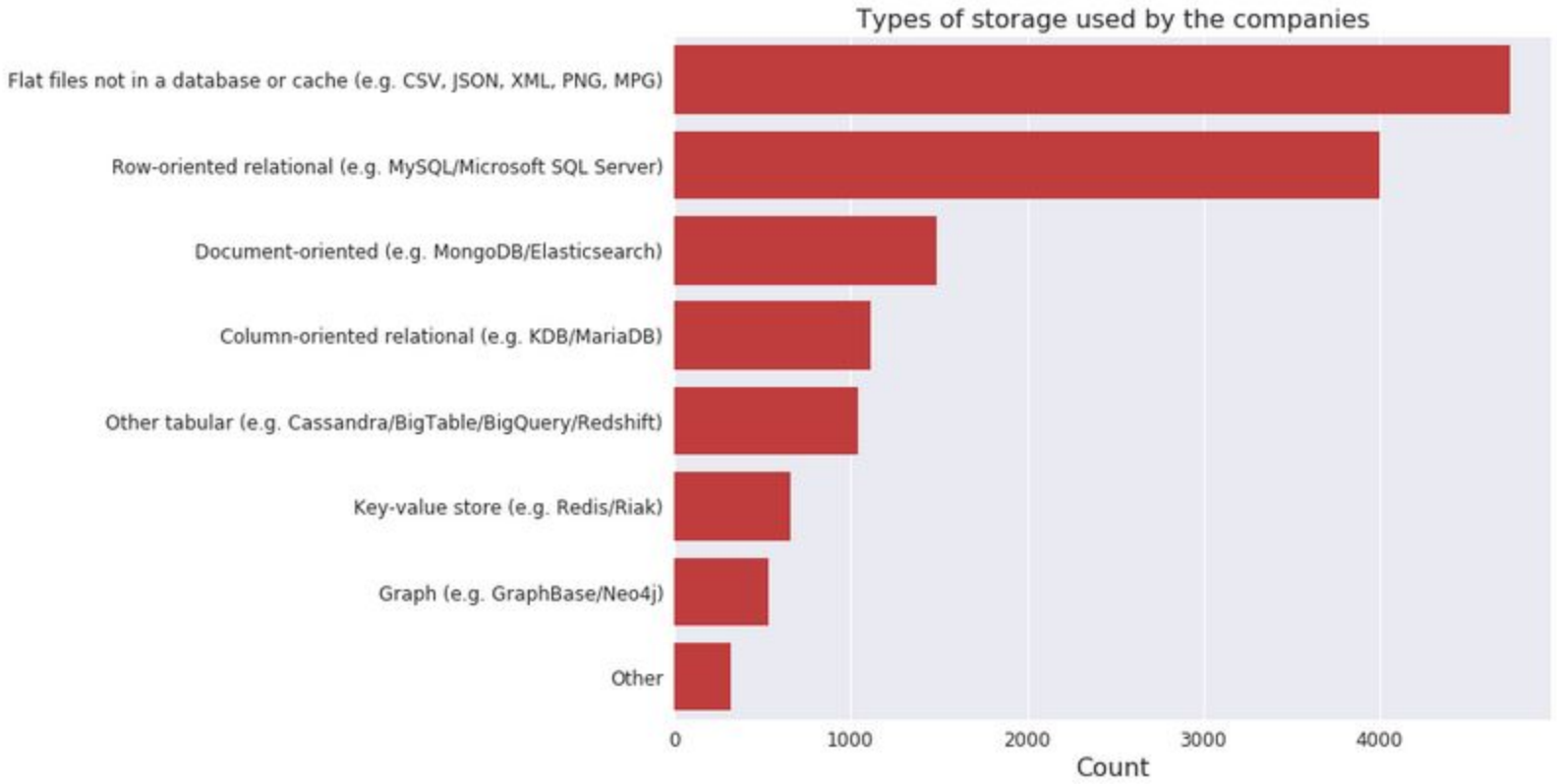
16000 responses

2017

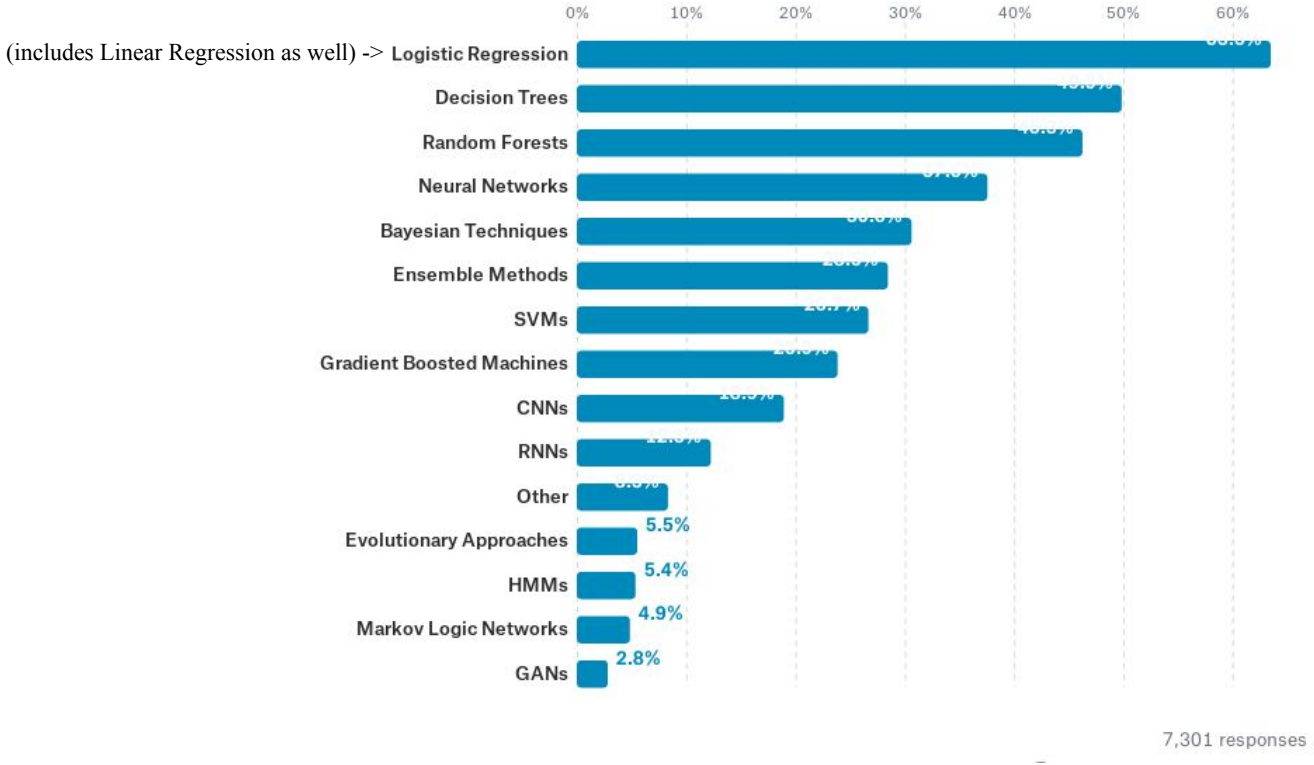
# The State of Data Science & Machine Learning

“What type of data is used at work?”




Types of storage used by the companies
Flat files not in a database or cache (e.g. CSV, JSON, XML, PNG, MPG)
Row-oriented relational (e.g. MySQL/Microsoft SQL Server)
Document-oriented (e.g. MongoDB/Elasticsearch)
Column-oriented relational (e.g. KDB/MariaDB)
Other tabular (e.g. Cassandra/BigTable/BigQuery/Redshift)
Key-value store (e.g. Redis/Riak)
Graph (e.g. GraphBase/Neo4j)
Other
0
1000
2000
3000
4000
Count

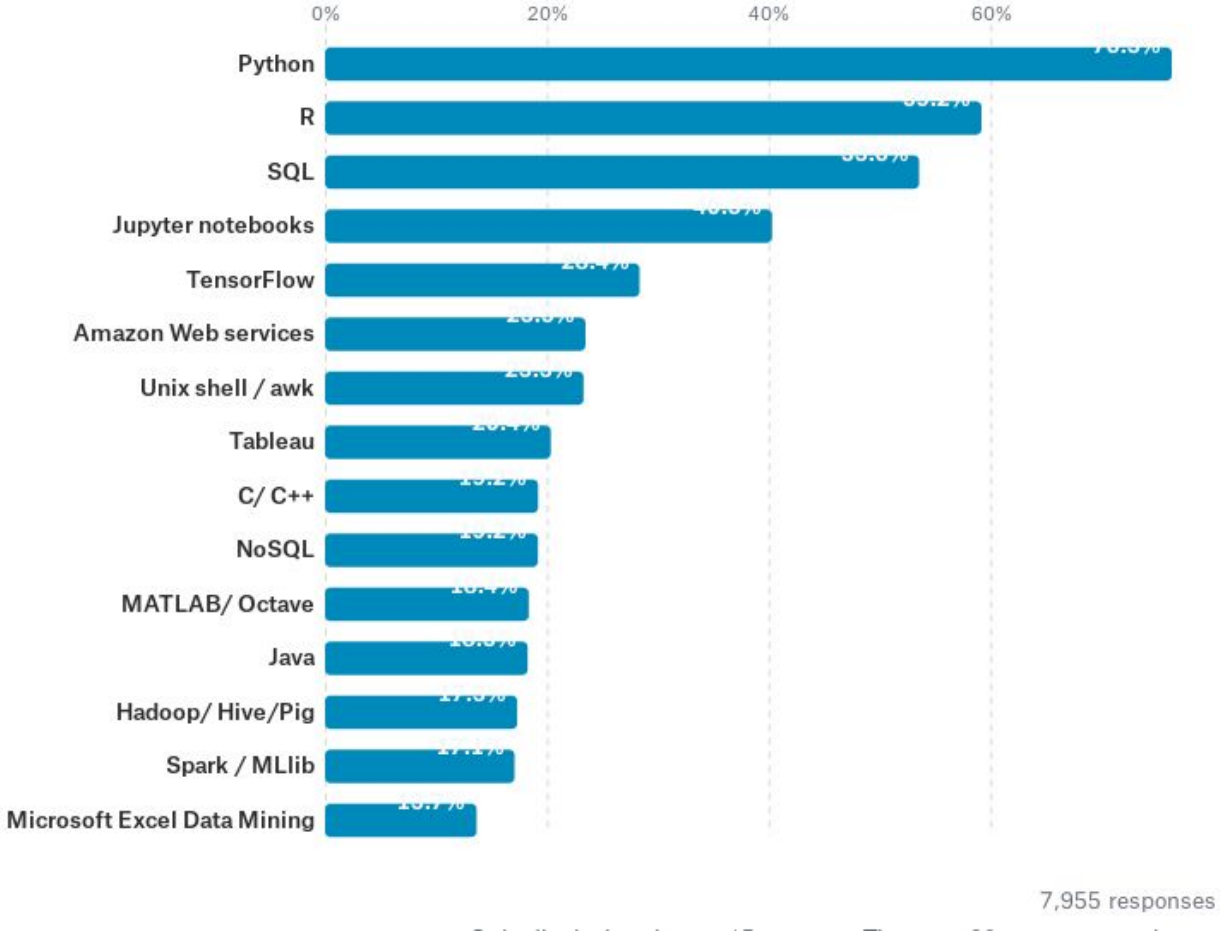
*What people do at work* - Algorithm used at work, selected from multiple choice by responders



*Machine Learning (ML) methods people want to learn* -
"Which ML, Data Science (DS) method are you most excited about learning in the next year?"

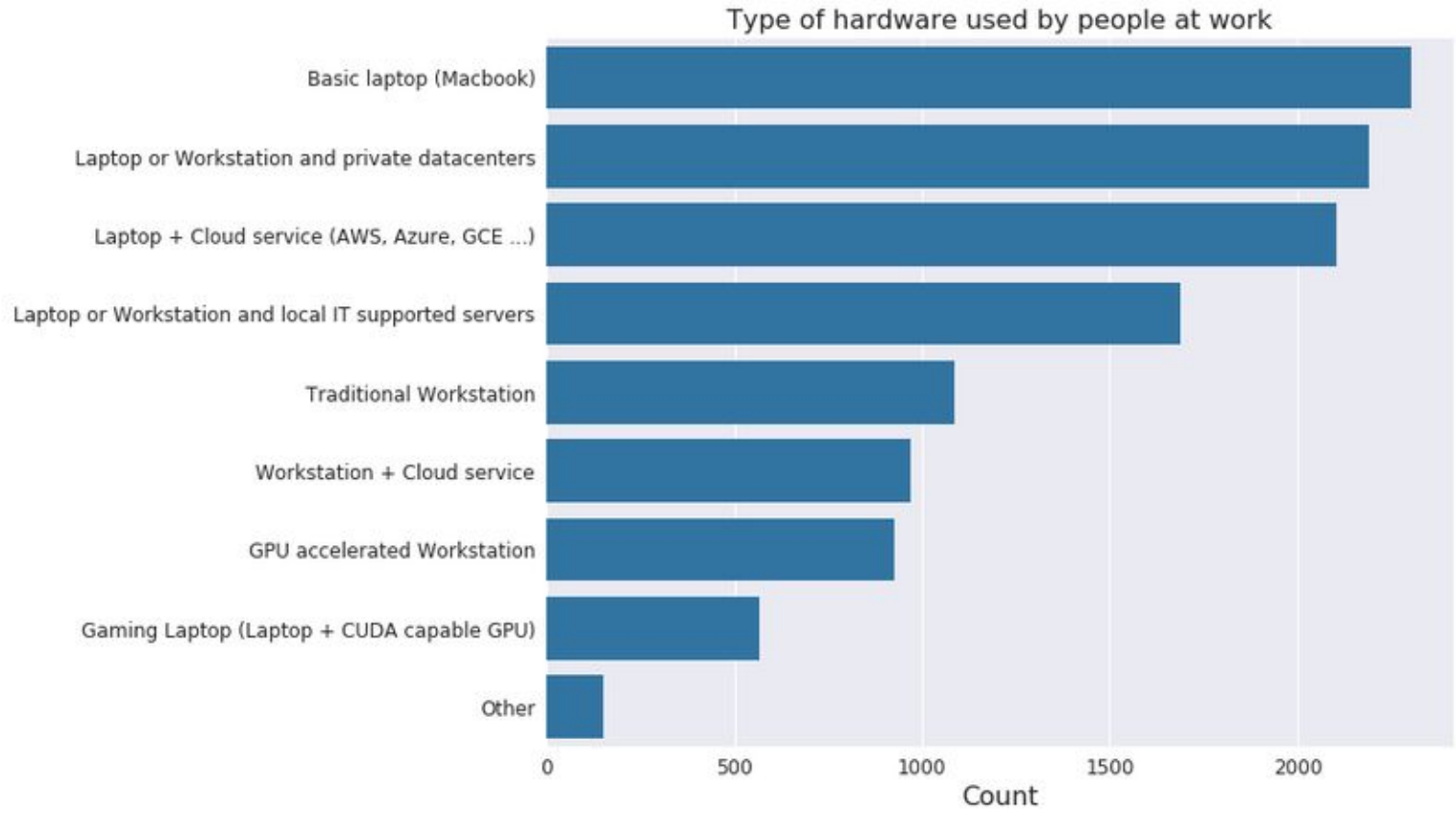
```
Deep learning                                   : 4362 --> approx. 40.27%
Neural Nets                                     : 1386 --> approx. 12.79%
Time Series Analysis                            : 680 --> approx. 6.28%
Bayesian Methods                                : 511 --> approx. 4.72%
Text Mining                                     : 493 --> approx. 4.55%
Genetic & Evolutionary Algorithms               : 425 --> approx. 3.92%
Social Network Analysis                         : 364 --> approx. 3.36%
Anomaly Detection                               : 307 --> approx. 2.83%
Ensemble Methods (e.g. boosting, bagging)       : 269 --> approx. 2.48%
Other                                           : 258 --> approx. 2.38%
```

“What tools are used at work?”



Tools for Next year -
“Which tool or technology are you most excited about learning in the next year?”

```
TensorFlow                                : 2621 --> approx. 23.83%
Python                                    : 1713 --> approx. 15.58%
R                                         : 910 --> approx. 8.27%
Spark / MLlib                             : 755 --> approx. 6.86%
Hadoop/Hive/Pig                           : 417 --> approx. 3.79%
```


Type of hardware used by people at work
Basic laptop (Macbook)
Laptop or Workstation and private datacenters
Laptop + Cloud service (AWS, Azure, GCE ...)
Laptop or Workstation and local IT supported servers
Traditional Workstation
Workstation + Cloud service
GPU accelerated Workstation
Gaming Laptop (Laptop + CUDA capable GPU)
Other
0
500
1000
1500
2000
Count

- GPU(s)
- Deep Learning/Neural Networks (with Linear/Logistic Regression)
- TensorFlow (tf)

# So what is TensorFlow doing now for data pipelines?

Given that Data elements have same type,

*Problem* : Dataset might be too large to materialize all at once … or infinite

# Functional programming to the rescue!

(Python) Functions on functions, composing functions, functionals

Compose functions like `.map()` and `.filter()` to preprocess

Apply functions such as `.shuffle()` and `.batch()` to transform data to feed into model

# Input pipelines = lazy lists (i.e. lazy evaluation)

```
Python 2.7.13 (default, May 10 2017, 20:04:28)
[GCC 6.3.1 20161221 (Red Hat 6.3.1-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> numbers = range(10)
>>> iterator = iter(numbers)
>>> print numbers
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> print iterator
<listiterator object at 0x7f703e5f45d0>
>>> print iterator.next()
0
>>> print iterator.next()
1
>>> print iterator.next()
2
>>> print iterator.next()
3
>>> print iterator.next()
4
>>>
```

# Introducing `tf.data`

Functional input pipelines in TensorFlow

# The Datasets API

`tf.data`

Better input pipelines for TensorFlow

Or

*“It will revolutionize the way we do data pipelines, the way we input data.”*

github.com/ernestyalumni/advanced-tensorflow

# Getting started; Install `tensorflow-gpu`, use `virtual-env`

`sudo apt-get install virtual-env` (Ubuntu)

`sudo dnf install virtual-env` (RedHat/Fedora)

`source ~/tf/bin/activate` (go to your virtual environment)

`pip install --upgrade tensorflow-gpu` # Python 2.7 and GPU(s) or

`pip3 install --upgrade tensorflow-gpu` # Python 3.x and GPU(s)

I do not recommend `pip install tensorflow`, for CPU only, as well.

# The `Dataset` interface

Data sources and functional transformations

`.csv, .txt` -> tf.data.Dataset

# The `Dataset` interface

Data sources and functional transformations

`.csv, .txt` -> tf.data.Dataset

(X,y)

```
6.1101,17.592
5.5277,9.1302
8.5186,13.662
7.0032,11.854
5.8598,6.8233
8.3829,11.886
7.4764,4.3483
8.5781,12
6.4862,6.5987
5.0546,3.8166
5.7107,3.2522
```

(X,y)

```
120,4,setosa,versicolor,virginica
6.4,2.8,5.6,2.2,2
5.0,2.3,3.3,1.0,1
4.9,2.5,4.5,1.7,2
4.9,3.1,1.5,0.1,0
5.7,3.8,1.7,0.3,0
4.4,3.2,1.3,0.2,0
5.4,3.4,1.5,0.4,0
6.9,3.1,5.1,2.3,2
6.7,3.1,4.4,1.4,1
5.1,3.7,1.5,0.4,0
5.2,2.7,3.9,1.4,1
6.9,3.1,4.9,1.5,1
5.8,4.0,1.2,0.2,0
5.4,3.9,1.7,0.4,0
7.7,3.8,6.7,2.2,2
6.3,3.3,4.7,1.6,1
6.8,3.2,5.9,2.3,2
7.6,3.0,6.6,2.1,2
6.4,3.2,5.3,2.3,2
```

# The `Dataset` interface

Data sources and functional transformations

`.csv, .txt` -> tf.data.Dataset

```
tf.data.TextLineDataset(
    filename_path)
```

(X,y)

```
6.1101,17.592
5.5277,9.1302
8.5186,13.662
7.0032,11.854
5.8598,6.8233
8.3829,11.886
7.4764,4.3483
8.5781,12
6.4862,6.5987
5.0546,3.8166
5.7107,3.2522
```

(X,y)

```
120,4,setosa,versicolor,virginica
6.4,2.8,5.6,2.2,2
5.0,2.3,3.3,1.0,1
4.9,2.5,4.5,1.7,2
4.9,3.1,1.5,0.1,0
5.7,3.8,1.7,0.3,0
4.4,3.2,1.3,0.2,0
5.4,3.4,1.5,0.4,0
6.9,3.1,5.1,2.3,2
6.7,3.1,4.4,1.4,1
5.1,3.7,1.5,0.4,0
5.2,2.7,3.9,1.4,1
6.9,3.1,4.9,1.5,1
5.8,4.0,1.2,0.2,0
5.4,3.9,1.7,0.4,0
7.7,3.8,6.7,2.2,2
6.3,3.3,4.7,1.6,1
6.8,3.2,5.9,2.3,2
7.6,3.0,6.6,2.1,2
6.4,3.2,5.3,2.3,2
```

```
tf.data.TextLineDataset(file_path)
    .skip(1) # Skip header row
    … (.map(...))...
    .batch(m_i) # e.g. m_i = 64
    .repeat(1000)  # repeat dataset this number of times, when training
```

```
tf.data.TextLineDataset(file_path)
    .map(decode_csv, num_parallel_calls=m_i) # Skip header row
```

We now have to write custom Python functions to parse data.

```
def decode_csv(line):
    Parsed_line = tf.decode_csv(line,
record_defaults=[[0.,],[0.,]], field_delim=”,”)
    …
    Xandy = parsed_line[:-1], parsed_line[-1]
    return Xandy
```

# The `Iterator` interface

## Sequential access to (batches of) `Dataset` elements

Create an `Iterator` from a `Dataset`:

```
dataset.make_one_shot_iterator()
```

```
dataset.make_initializable_iterator()
```

# The `Iterator` interface

Sequential access to (batches of) `Dataset` elements

```
next_X_i, next_y_i = iterator.get_next()

while … :

    X_i,y_i = sess.run([next_X_i, next_y_i])
```

# No more `tf.placeholder`, no more Queuing

Use `next_X_i, next_y_i = iterator.get_next()` *directly*.

e.g.

```
yhat = tf.matmul(next_X_i, w) + b
```

# No more `tf.placeholder`, no more Queuing

1 Answer

active oldest votes

2

Placeholder is not needed when using DataSet apis any more, for reading data already a part of `tf.Graph`.

We do not need to read file in python code, and feed them when training, but read data as a tensorflow op in the tf.Graph, it will be much more efficiency for tensorflow ops mainly run in cpp.

as in your case, this to lines:

```
x = tf.placeholder(tf.float32, [None, INPUT_SIZE], name='INPUT')
y_ = tf.placeholder(tf.float32, [None, OUTPUT_SIZE], name='OUTPUT')
```

change into:

```
x = next_example
y_ = next_label
```

And **remove** the `feed_dict` when calling `Session.run`

share improve this answer

edited Sep 11 at 13:40

answered Sep 11 at 13:36

Tianjin Gu

541 • 2 • 13

cf.

https://stackoverflow.com/questions/46156778/how-to-map-iterators-output-to-placeholder-in-loss-function-in-tensorflow-using

```
def input_fn():
    dataset = …

    # A one-shot iterator automatically initializes itself on first use.
    iterator = dataset.make_one_shot_iterator()

    # The return value of get_next() matches the dataset element type.
    images, labels = iterator.get_next()

    return images, labels

# The input_fn can be used as a regular Estimator input function.
estimator = tf.estimator.Estimator(...)
estimator.train(train_input_fn=input_fn, ...)
```

# Demo

# `tf.data` Summary

**`tf.data.Dataset`**

Represents input pipeline using functional transformations

**`tf.data.Iterator`**

Provides sequential access to (batches of) elements of a `Dataset`

# `tf.data` Conclusion

Getting your data into TensorFlow with `tf.data`

- Fast
  - `tf.data` is implemented in C++ to avoid Python overhead
- Flexible

*“It will revolutionize the way we do data pipelines, the way we input data.”*

```
Python 2.7.13 (default, May 10 2017, 20:04:28)
[GCC 6.3.1 20161221 (Red Hat 6.3.1-1)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/home/topolo/tf/lib/python2.7/site-packages/tensorflow/__init__.py", line 24, in <module>
    from tensorflow.python import *
  File "/home/topolo/tf/lib/python2.7/site-packages/tensorflow/python/__init__.py", line 49, in <
module>
    from tensorflow.python import pywrap_tensorflow
  File "/home/topolo/tf/lib/python2.7/site-packages/tensorflow/python/pywrap_tensorflow.py", line
 72, in <module>
    raise ImportError(msg)
ImportError: Traceback (most recent call last):
  File "/home/topolo/tf/lib/python2.7/site-packages/tensorflow/python/pywrap_tensorflow.py", line
 58, in <module>
    from tensorflow.python.pywrap_tensorflow_internal import *
  File "/home/topolo/tf/lib/python2.7/site-packages/tensorflow/python/pywrap_tensorflow_internal.
py", line 28, in <module>
    _pywrap_tensorflow_internal = swig_import_helper()
  File "/home/topolo/tf/lib/python2.7/site-packages/tensorflow/python/pywrap_tensorflow_internal.
py", line 24, in swig_import_helper
    _mod = imp.load_module('_pywrap_tensorflow_internal', fp, pathname, description)
ImportError: libcublas.so.8.0: cannot open shared object file: No such file or directory


Failed to load the native TensorFlow runtime.

See https://www.tensorflow.org/install/install_sources#common_installation_problems

for some common reasons and solutions.  Include the entire stack trace
above this error message when asking for help.
>>>
```

# Upgrade to CuDNN 7 and CUDA 9 #12052

Closed **tpankaj** opened this issue on Aug 4 · 78 comments

**tpankaj** commented on Aug 4 • edited

### System information

- **Have I written custom code (as opposed to using a stock example script provided in TensorFlow)**: No
- **OS Platform and Distribution (e.g., Linux Ubuntu 16.04)**: Windows Server 2012
- **TensorFlow installed from (source or binary)**: binary
- **TensorFlow version (use command below)**: 1.3.0-rc1
- **Python version**: 3.5.2
- **Bazel version (if compiling from source)**: N/A
- **CUDA/cuDNN version**: CUDA V8.0.44, CuDNN 6.0
- **GPU model and memory**: Nvidia GeForce GTX 1080 Ti, 11 GB
- **Exact command to reproduce**: N/A

### Describe the problem

Please upgrade TensorFlow to support CUDA 9 and CuDNN 7. Nvidia claims this will provide a 2x performance boost on Pascal GPUs.

37

**shivaniag** added stat:awaiting tensorflower type:feature labels on Aug 4

**shivaniag** commented on Aug 4

**@tfboyd** do you have any comments on this?

**tfboyd** commented on Aug 4 • edited Contributor

cuDNN 7 is still in preview mode and is being worked on. We just moved to cuDNN 6.0 with 1.3, which should go final in a couple weeks. You can download cuDNN 1.3.0rc2 if you are interested in that. I have not compiled with cuDNN 7 or CUDA 9 yet. I have heard CUDA 9 is not easy to install on all platforms and

Assign
tfbo
Labels
type:f
Project
None ye
Milesto
No mile
Notifica
You're r
from this
38 part
and othe

**smitshilu** commented 2 days ago

I am trying to build 1.4 with CUDA 9 and cuDNN 7 in mac 10.13 high sierra. I am keep getting this error

```
ERROR: /Users/smitshilu/tensorflow/tensorflow/core/kernels/BUILD:2948:1: output 'tensorflow/c
ERROR: /Users/smitshilu/tensorflow/tensorflow/core/kernels/BUILD:2948:1: not all outputs were
Target //tensorflow/tools/pip_package:build_pip_package failed to build
```

Any solution for this?

**vellamike** commented 2 days ago

**@smitshilu** possibly related #2143

**grisaitis** referenced this issue 2 days ago

**conv2d_transpose crashing, "NotFoundError: No algorithm worked!", only with batch size >=2^16** #13869 Open



**ViktorM** commented a day ago

Why 1.4 still doesn't have CUDA 9 in binaries? This version was released quit a long ago and for using with V100 building from source is required which is not so smooth and fast following to a number of issues reported.

**vellamike** commented a day ago

**@ViktorM** What issues did you have compiling from source? It was a bit tricky but not that hard.

**tfboyd** commented 20 hours ago Contributor

26-SEP-2017 was the GA for CUDA 9. If we release CUDA 9 + cuDNN 7 binaries in Q4 I think this will be the fastest we have upgraded cuDNN. I was not here for 8.5 to 9 so I have no idea. I would like us to go a little faster but this also means anyone with a CUDA 8 setup has to upgrade not just to CUDA 9 but they also need to upgrade their device driver to 384.x, which I can say is not something production people take lightly.

Ideally we would have infinite (or just a few more but the matrix explodes fast) builds, but that is another problem that would take a long time to explain and I doubt many people care.

**yaroslavvb** commented 14 hours ago — Contributor

BTW, I observed 85 T ops/second with CUDA 9 on float16 matmul/V100 using Nvidia's NGC TensorFlow container (as opposed to 8.8 T ops/s on my GTX 1080 at home). Really looking forward to having these improvements in the officially supported version!

**tfboyd** commented 14 hours ago — Contributor

**@yaroslavvb** Being very honest we are working through some FP16 issues. There is a path in the tf_cnn_benchmarks for FP16 and the focus is on ResNet50 first and we are working on auto scaling for FP16 as well. You can give it a try if you are interested but we are actively working through some problems. People are on it and it is just taking time. We finally have DGX-1s in house so we can also play with the same containers and try to keep track of performance on that exact platform moving forward.

**vickylance** commented 17 minutes ago

Ok, So I am going to install Ubuntu 17.10 and I just wanted to try all the latest stuffs for fun.
Before I do I just wanted to know did any one try the below stacks building from source and got any luck?

-> Ubuntu 17.10, CUDA 9.0, cuDNN 7.0, TF master
-> Ubuntu 17.10, CUDA 8.0, cnDNN 6.1, TF 1.4

**aluo-x** commented 7 minutes ago

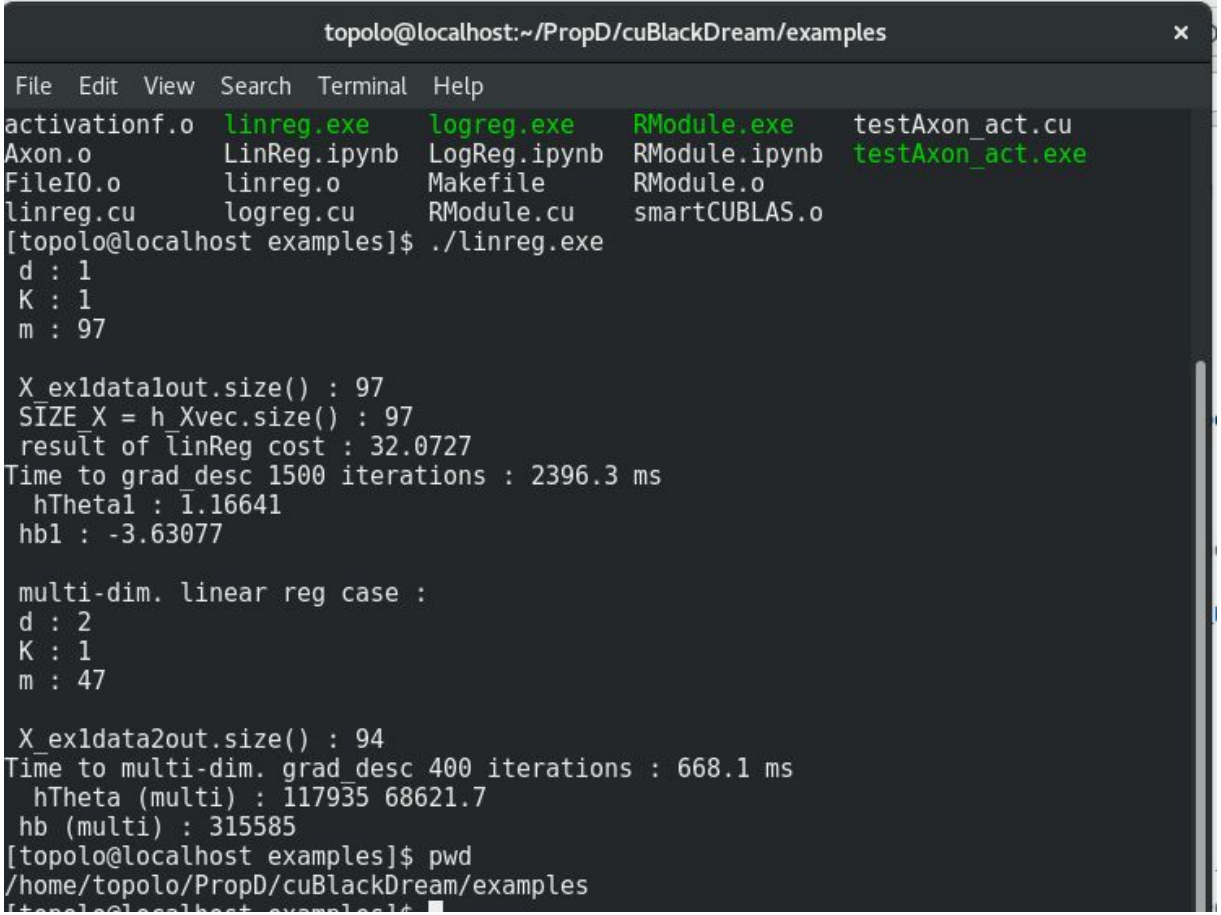
I am encountering the same issue as **@xsr-ai**, specifically using Python 3.6.3, VS 2017, CUDA 9, cuDNN 7.

# cuBlackDreams

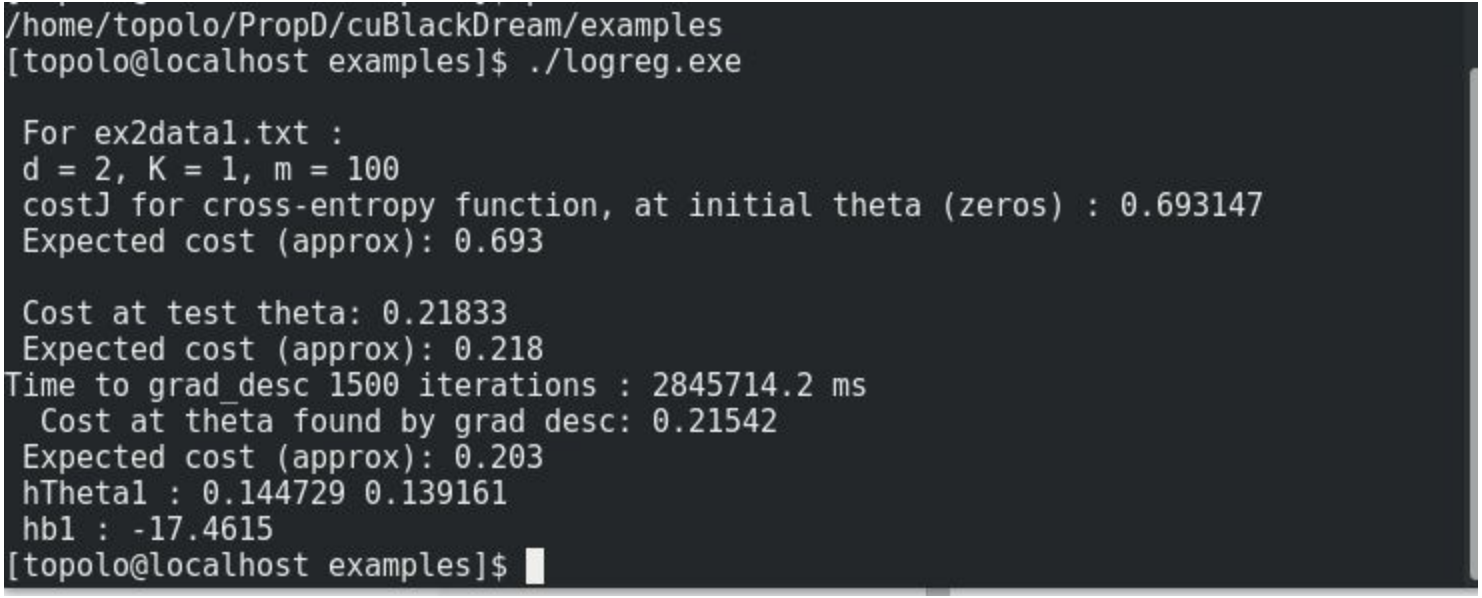
https://github.com/ernestyalumni/cuBlackDream

- CUDA C++14, CUDA 9, C++14 smart pointers
- Parallel programming, parallel algorithms only
- Fast File I/O with C++14 binary format (`std::ios::binary`)
- Multi-GPU ready (using CUDA Unified Memory Management)

Machine Learning and Deep Learning in CUDA, for the GPU




https://github.com/ernestyalumni/cuBlackDream




https://github.com/ernestyalumni/cuBlackDream