Dive into Deep Learning in 1 Day

1 Basics · 2 Convnets · 3 Computation · 4 Sequences

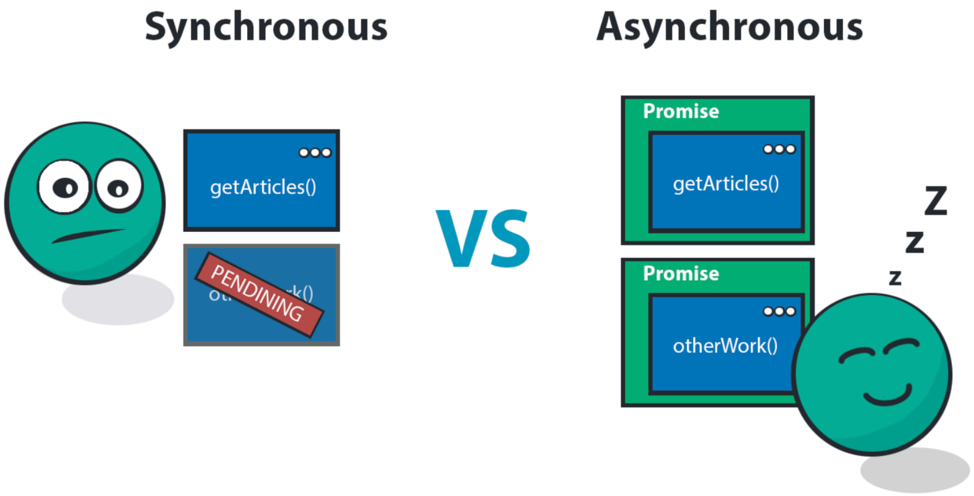
ODSC 2019

Alex Smola
http://courses.d2l.ai/odsc2019/

Outline

- Performance
  - Async-computation
  - Multi-GPU/machine training
- Computer Vision
  - Image augmentation
  - Fine tuning

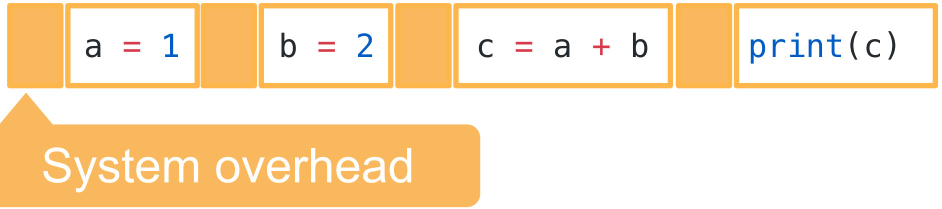
Asynchronous Computing & Parallelization



http://courses.d2l.ai/odsc2019/

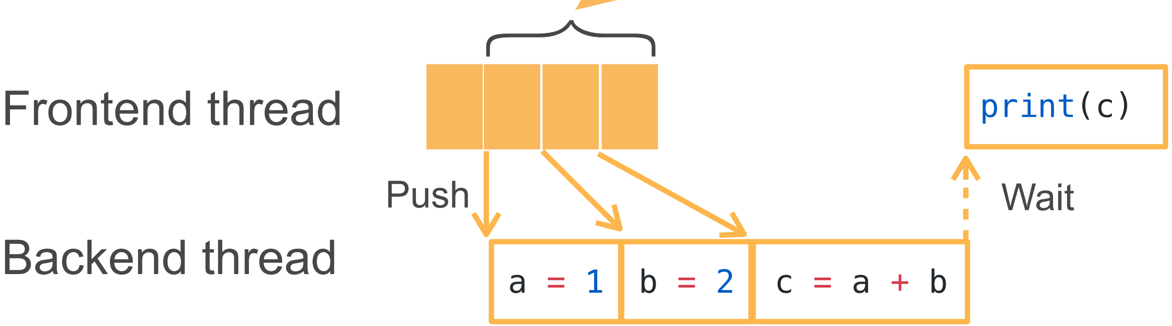
Asynchronous Execution

- Execute one-by-one



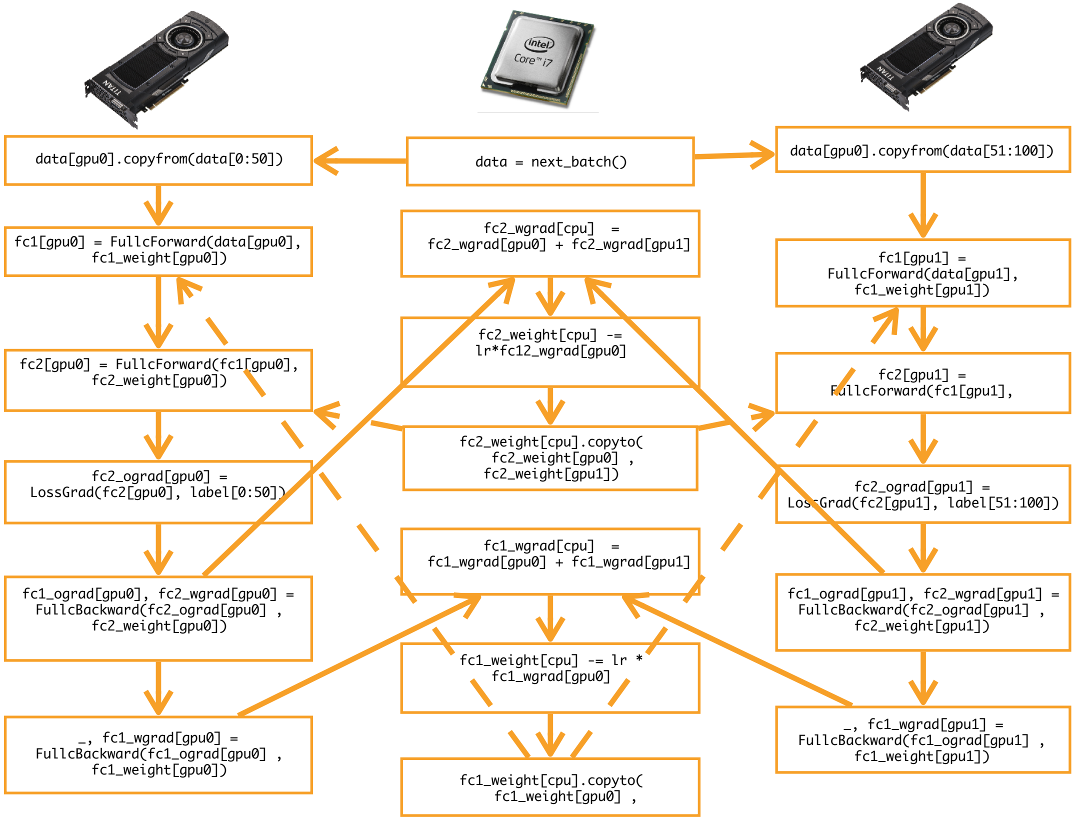

```
a = 1
b = 2
c = a + b
print(c)
```

- With a backend thread



http://courses.d2l.ai/odsc2019/

Writing Parallel Program is Painful



Single hidden-layer MLP with 2 GPUs

Scales to hundreds of layers and tens of GPUs

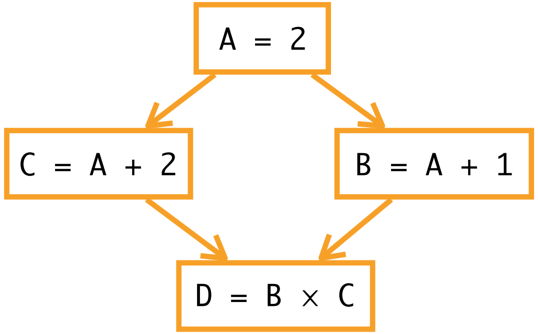
http://courses.d2l.ai/odsc2019/

Auto Parallelization

Write serial programs

```
A = nd.ones((2,2)) * 2
C = A + 2
B = A + 1
D = B * C
```

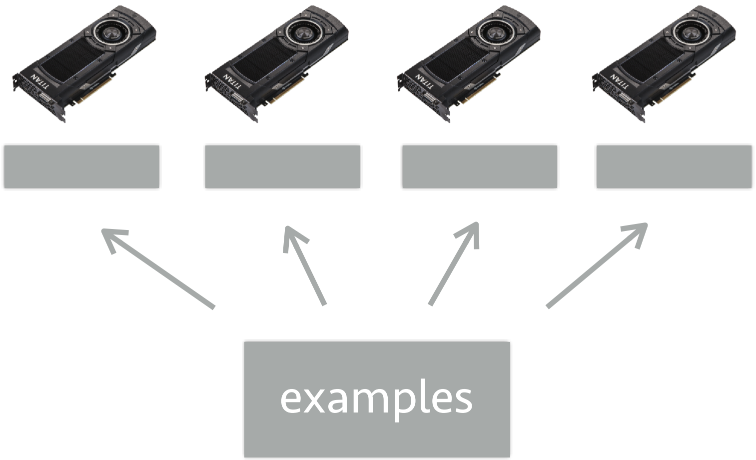
Run in parallel



Data Parallelism

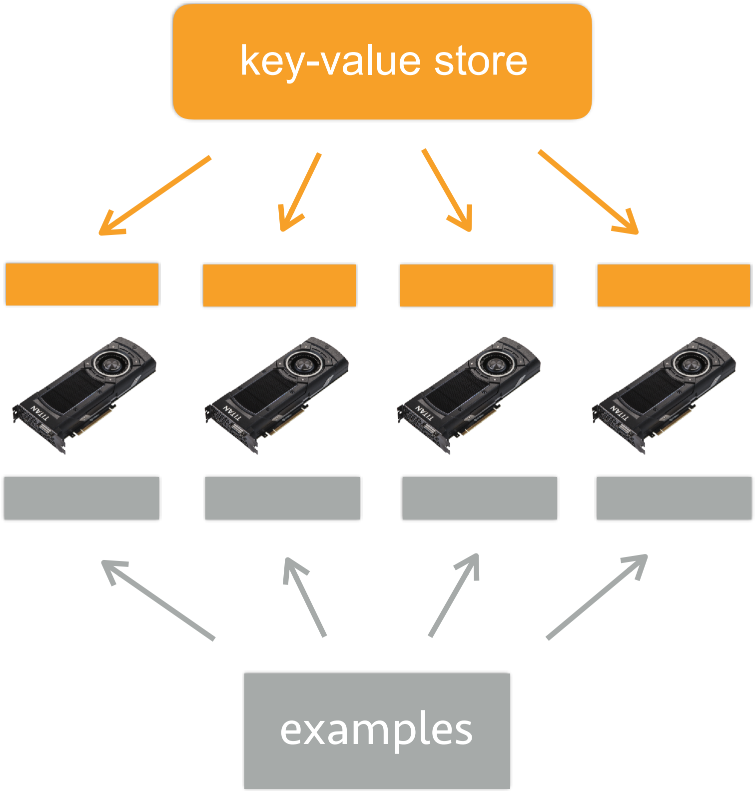
examples

Data Parallelism



1. Read a data partition

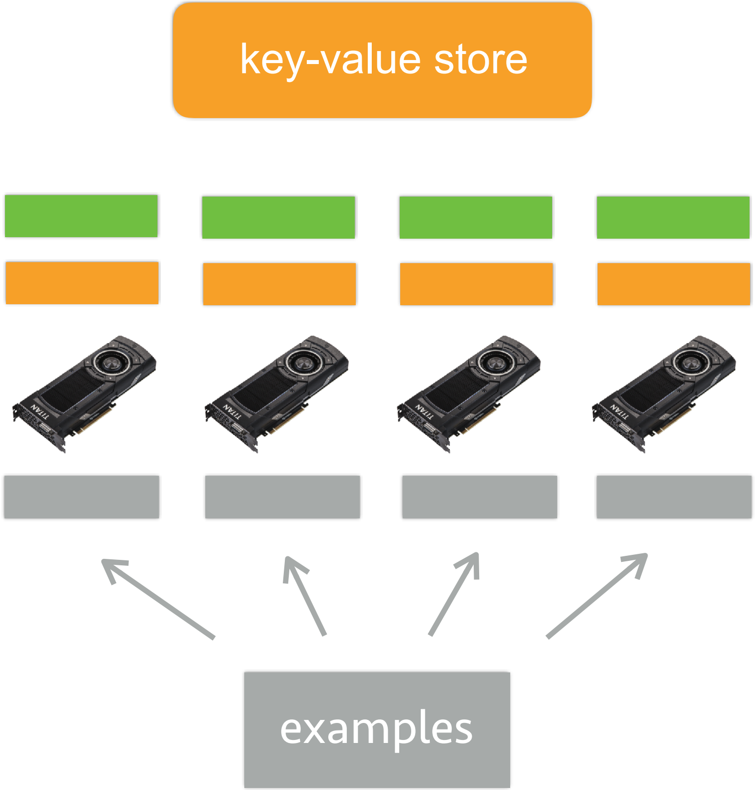
Data Parallelism



1. Read a data partition
2. Pull the parameters

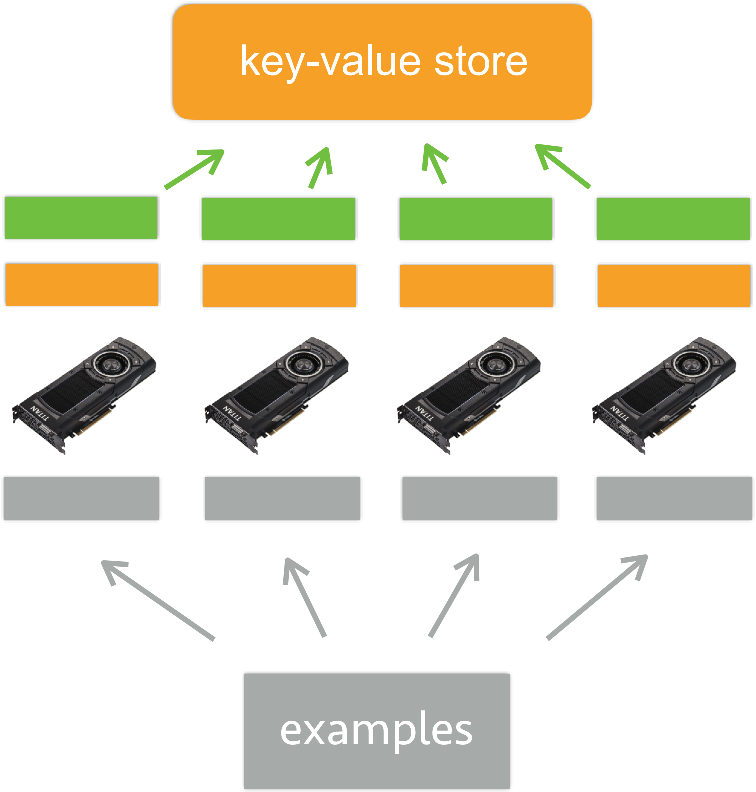
http://courses.d2l.ai/odsc2019/

Data Parallelism



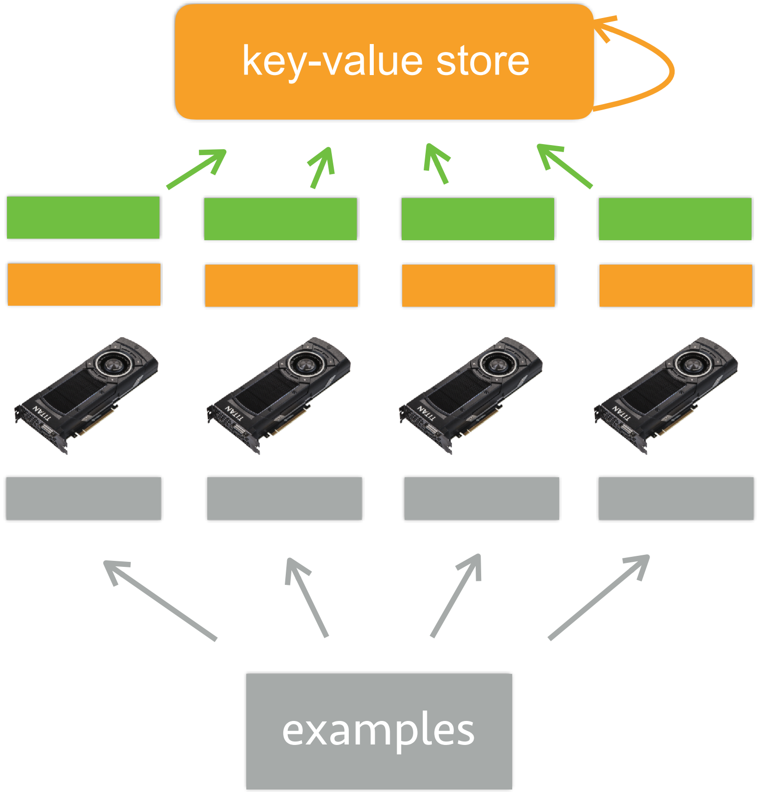
1. Read a data partition
2. Pull the parameters
3. Compute the gradient

Data Parallelism



1. Read a data partition
2. Pull the parameters
3. Compute the gradient
4. Push the gradient

Data Parallelism



1. Read a data partition
2. Pull the parameters
3. Compute the gradient
4. Push the gradient
5. Update the parameters

Distributed Training

http://courses.d2l.ai/odsc2019/

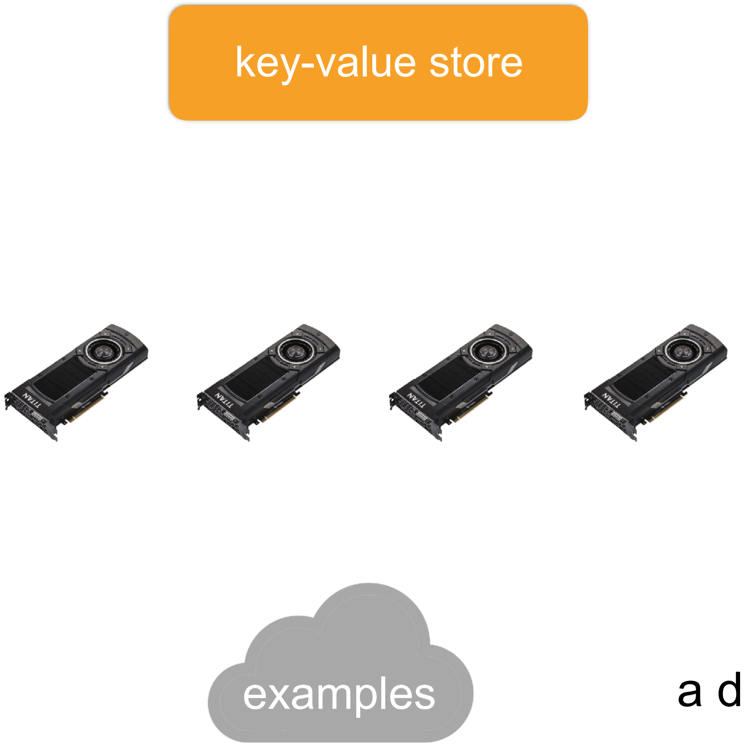
Distributed Computing

key-value store

examples

http://courses.d2l.ai/odsc2019/

Distributed Computing



Store data in
a distributed filesystem

Distributed Computing

key-value store

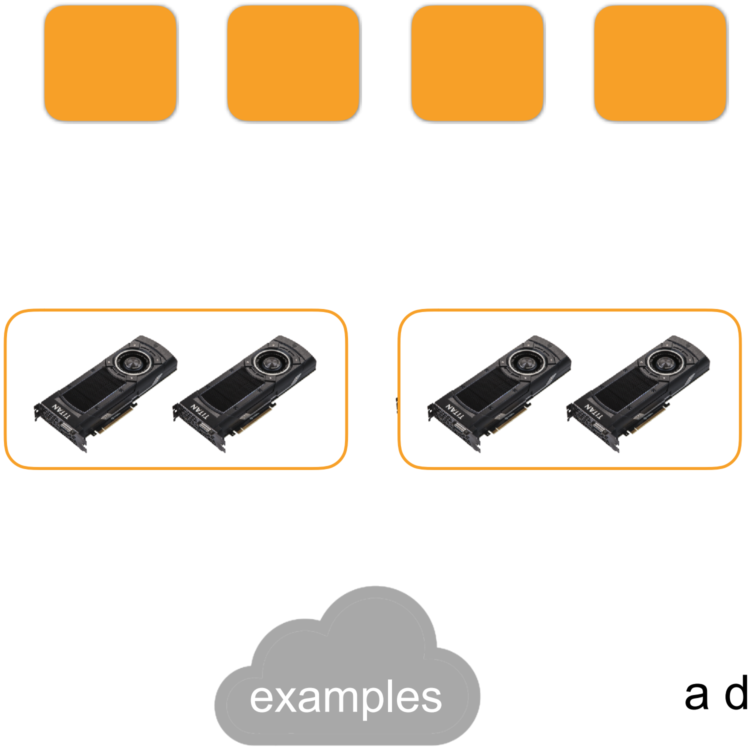
multiple
worker machines

examples

Store data in
a distributed filesystem

http://courses.d2l.ai/odsc2019/

Distributed Computing

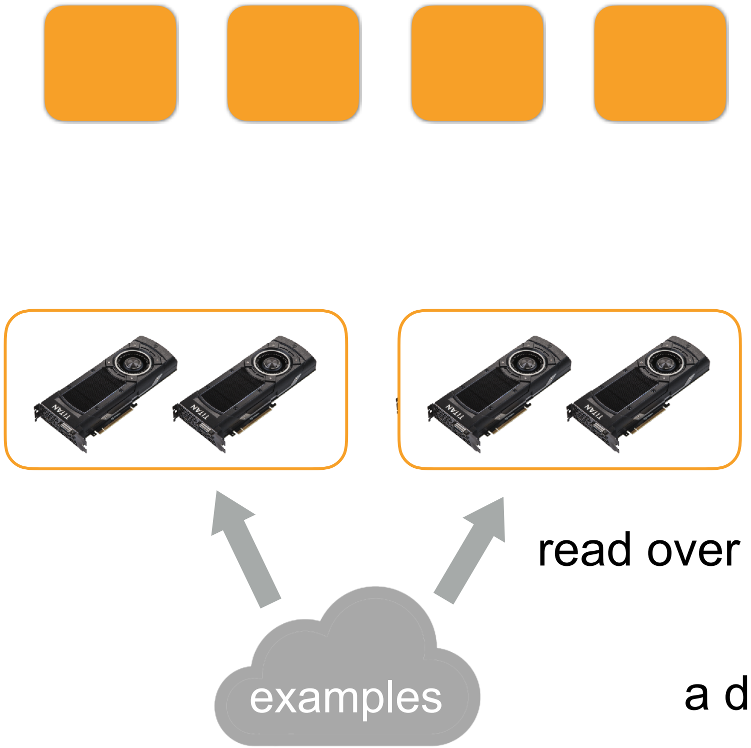


multiple
server machines

multiple
worker machines

Store data in
a distributed filesystem

Distributed Computing



multiple
server machines

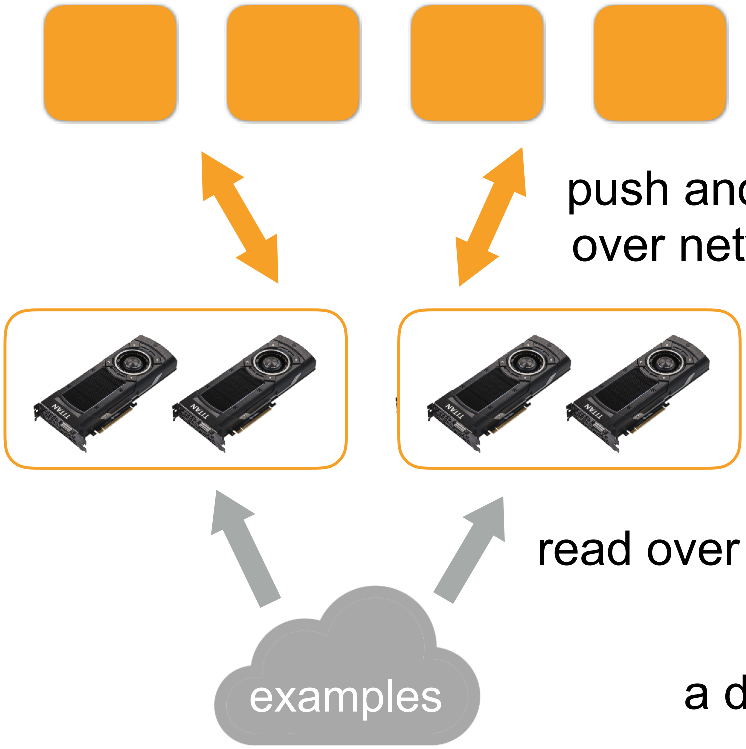
multiple
worker machines

read over network

Store data in
a distributed filesystem

http://courses.d2l.ai/odsc2019/

Distributed Computing



multiple
server machines

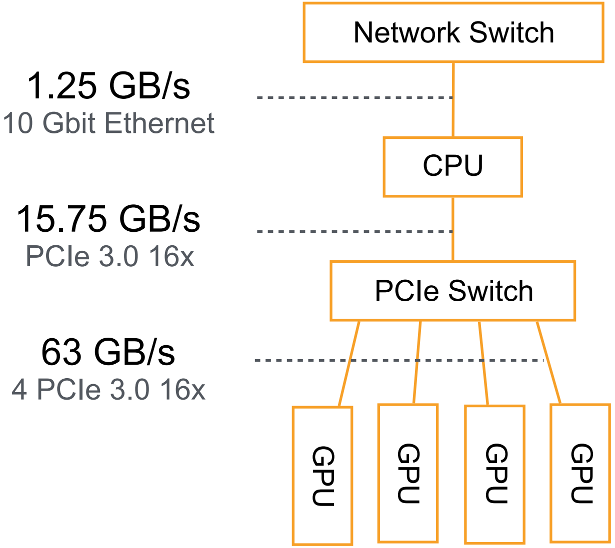
push and pull
over network

multiple
worker machines

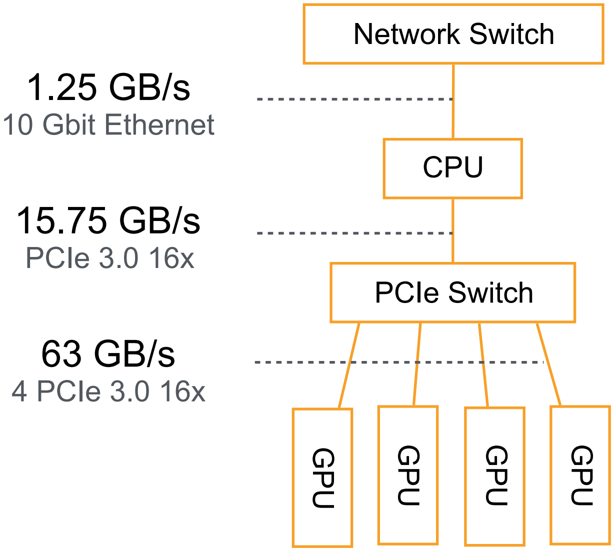
read over network

Store data in
a distributed filesystem

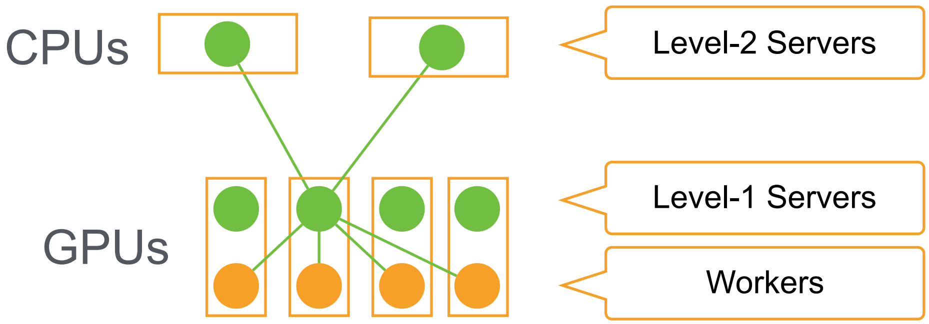
GPU Machine Hierarchy



GPU Machine Hierarchy

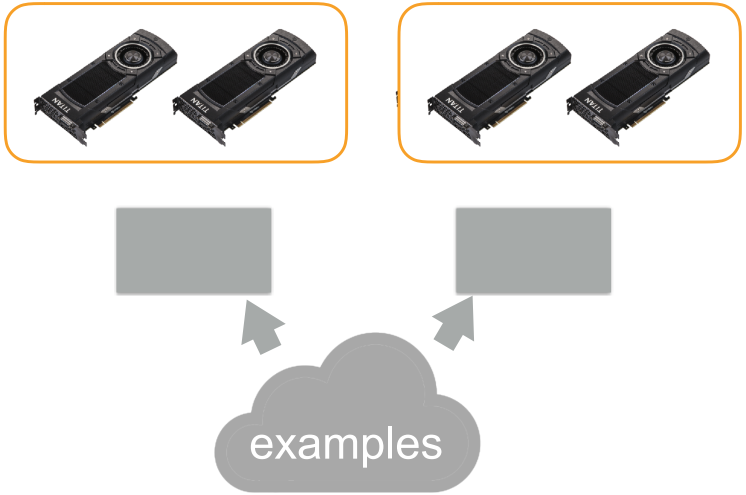


Hierarchical parameter server



Iterating a Batch

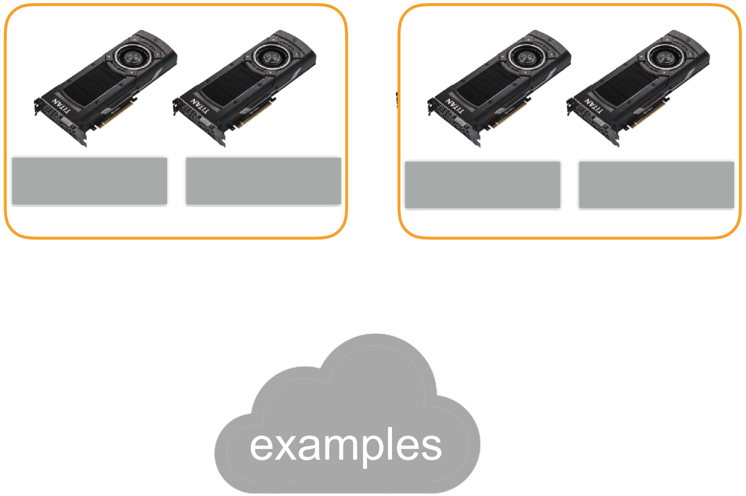
- Each worker machine read a part of the data batch



http://courses.d2l.ai/odsc2019/

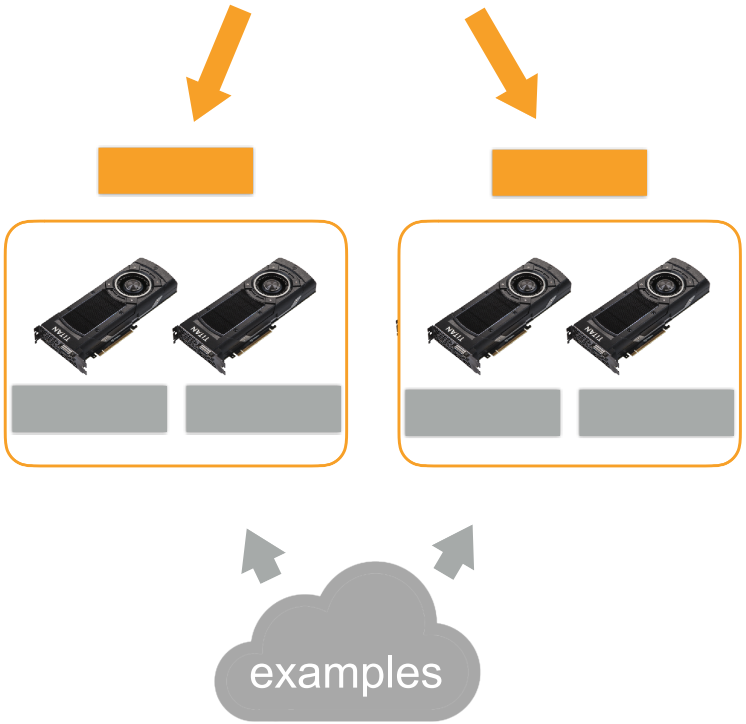
Iterating a Batch

- Further split and move to each GPU



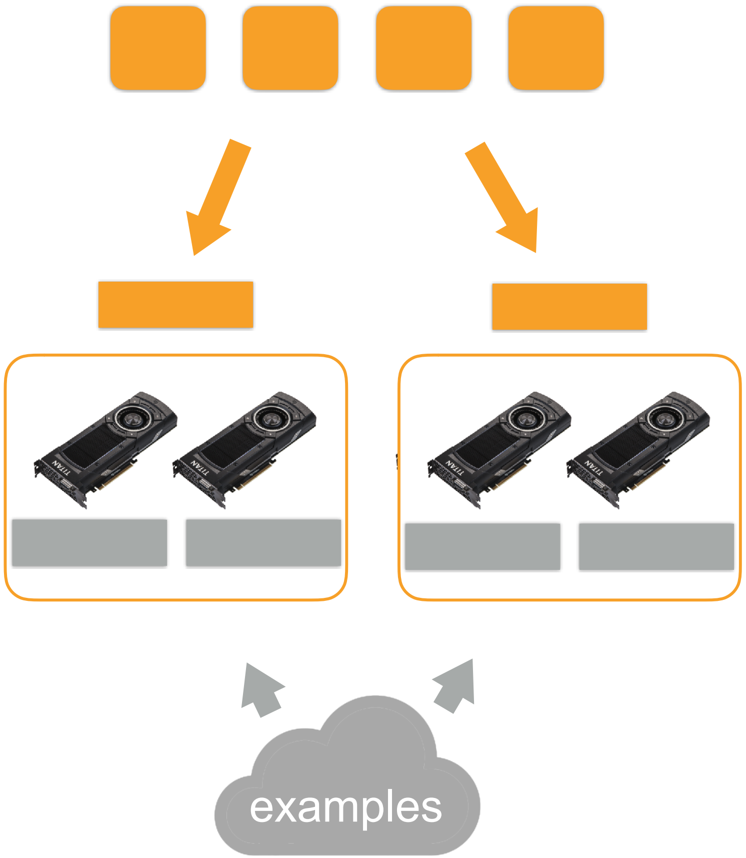
http://courses.d2l.ai/odsc2019/

Iterating a Batch



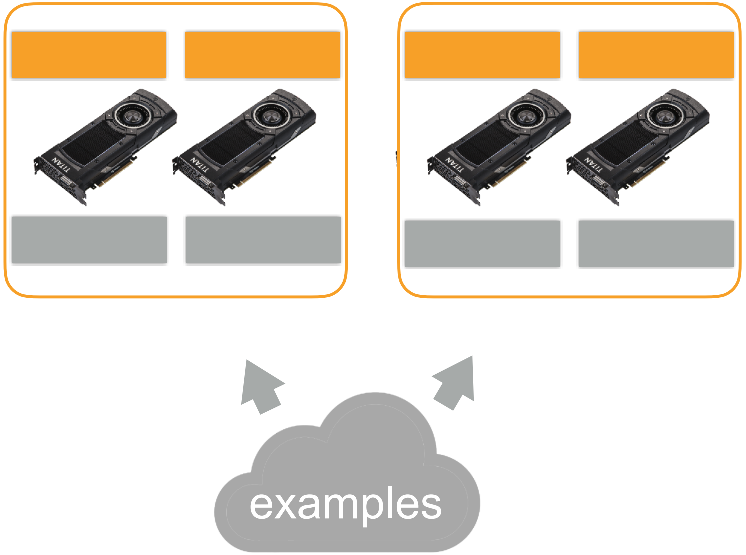
- Each server maintain a part of parameters
- Each worker pull the whole parameters from servers

Iterating a Batch



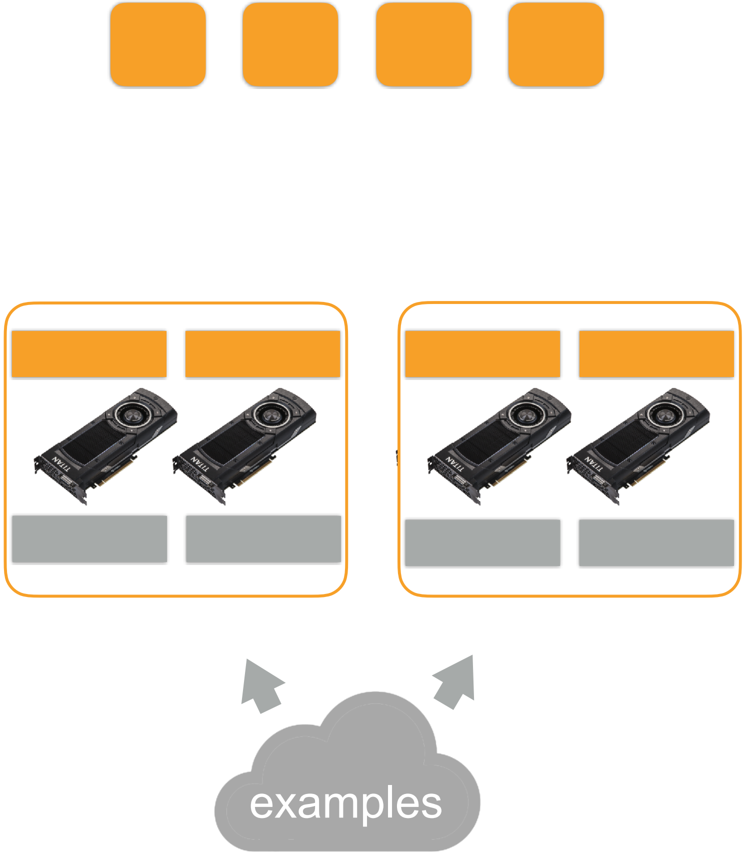
- Each server maintain a part of parameters
- Each worker pull the whole parameters from servers

Iterating a Batch



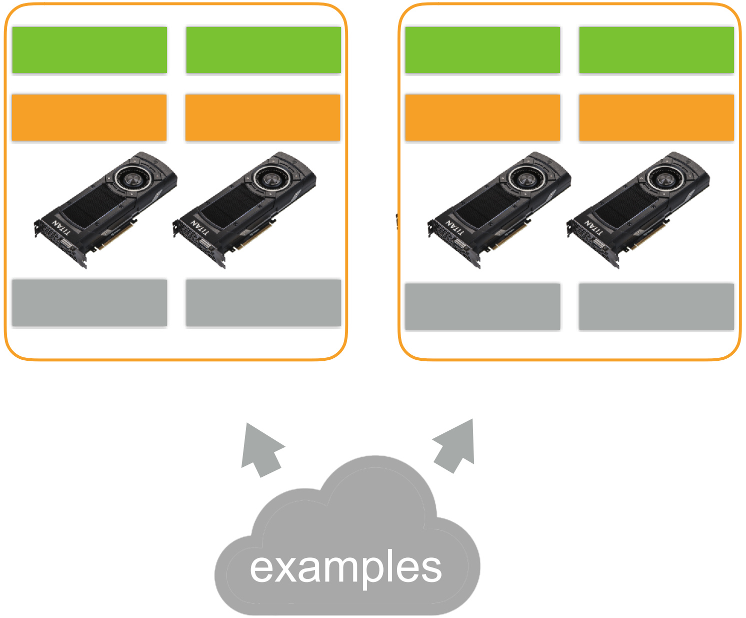
- Copy parameters into each GPU

Iterating a Batch



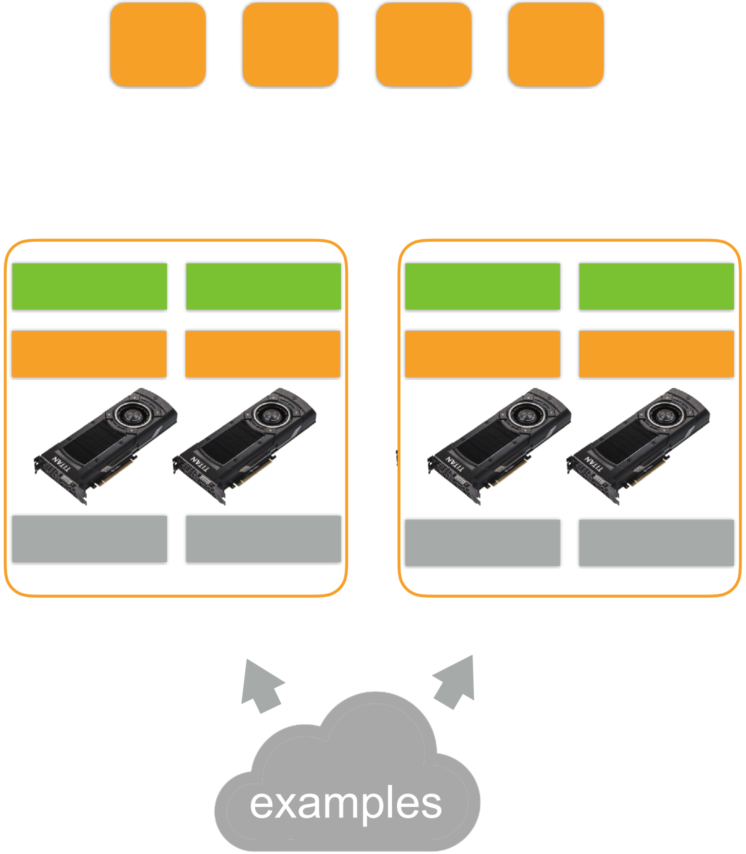
- Copy parameters into each GPU

Iterating a Batch



- Each GPU computes gradients

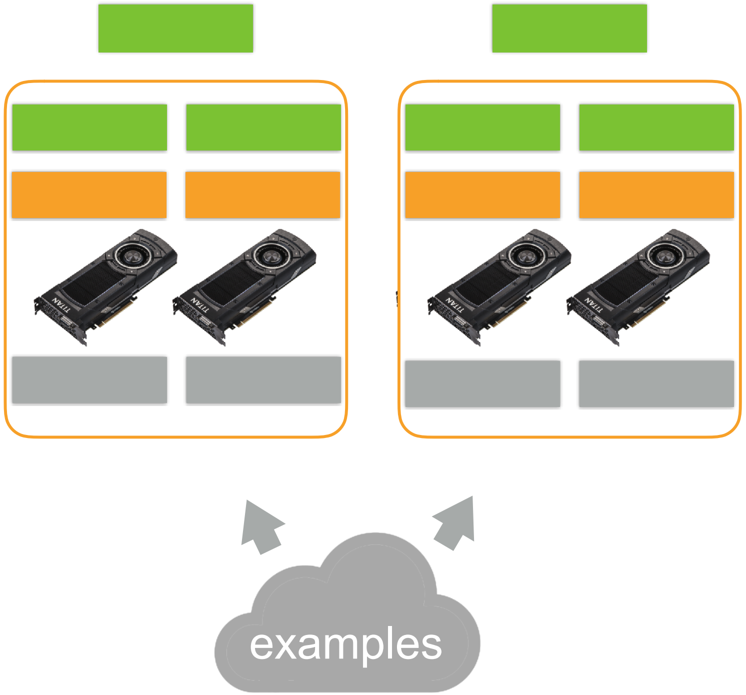
Iterating a Batch



- Each GPU computes gradients

http://courses.d2l.ai/odsc2019/

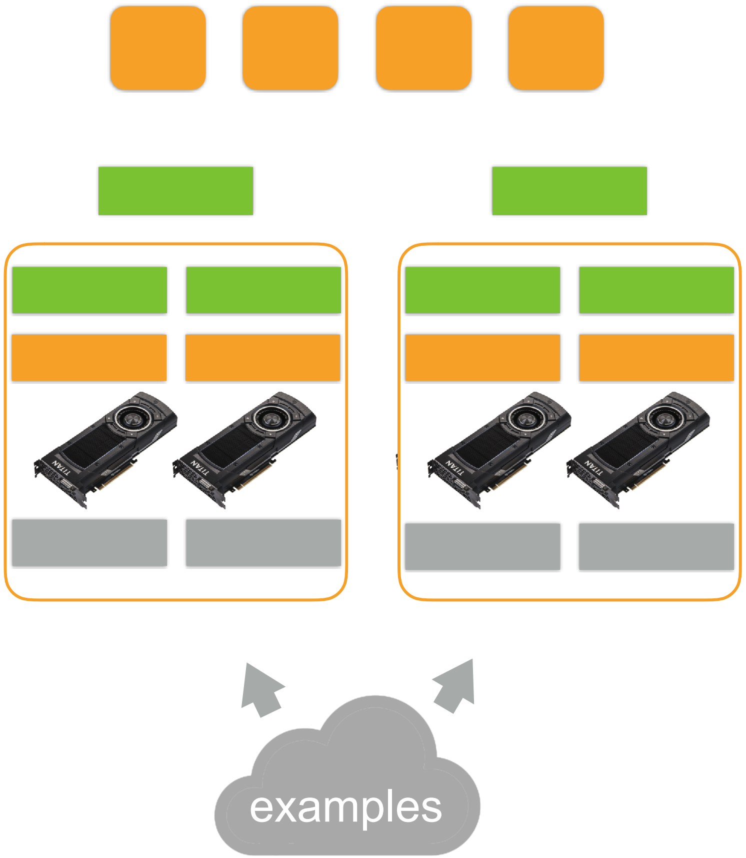
Iterating a Batch



- Sum the gradients over all GPU

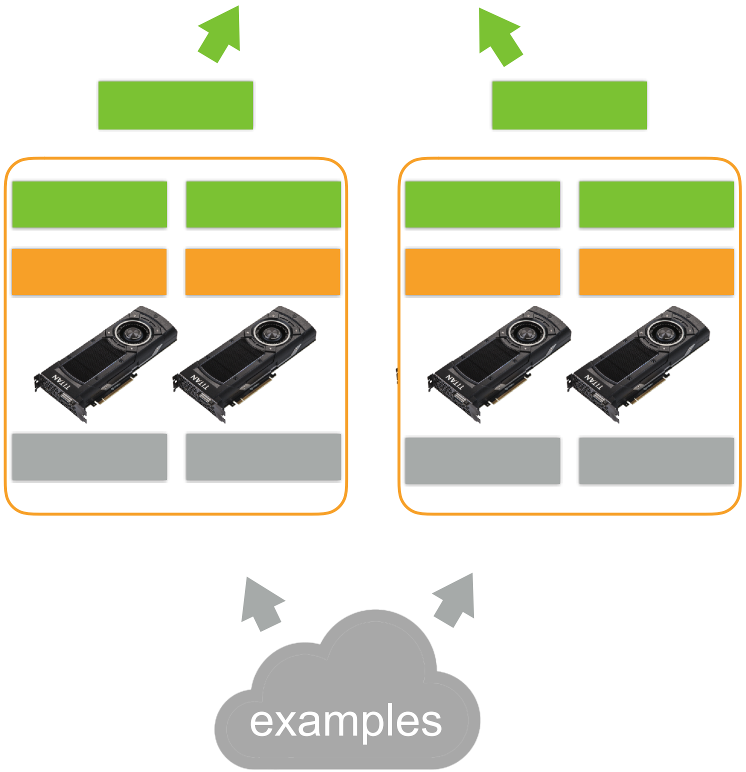
http://courses.d2l.ai/odsc2019/

Iterating a Batch



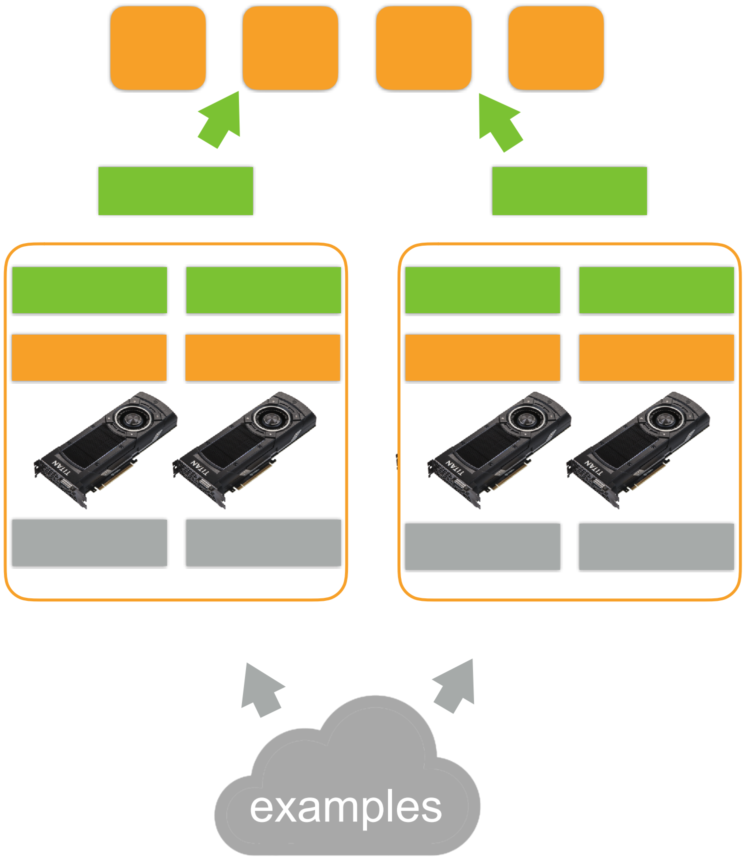
- Sum the gradients over all GPU

Iterating a Batch



- Push gradients into servers

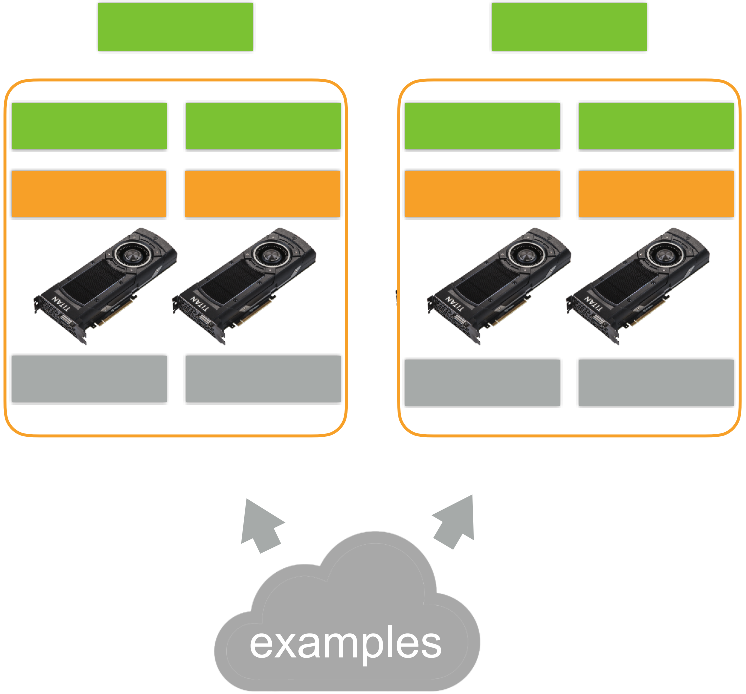
Iterating a Batch



- Push gradients into servers

http://courses.d2l.ai/odsc2019/

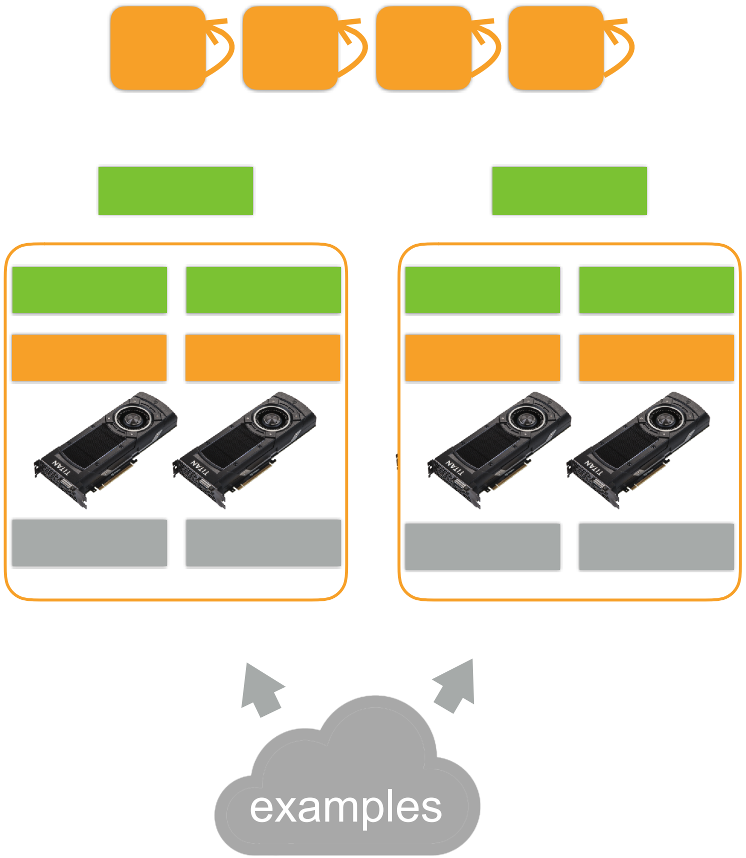
Iterating a Batch



- Each server sum gradients from all workers, then updates its parameters

http://courses.d2l.ai/odsc2019/

Iterating a Batch



- Each server sum gradients from all workers, then updates its parameters

http://courses.d2l.ai/odsc2019/

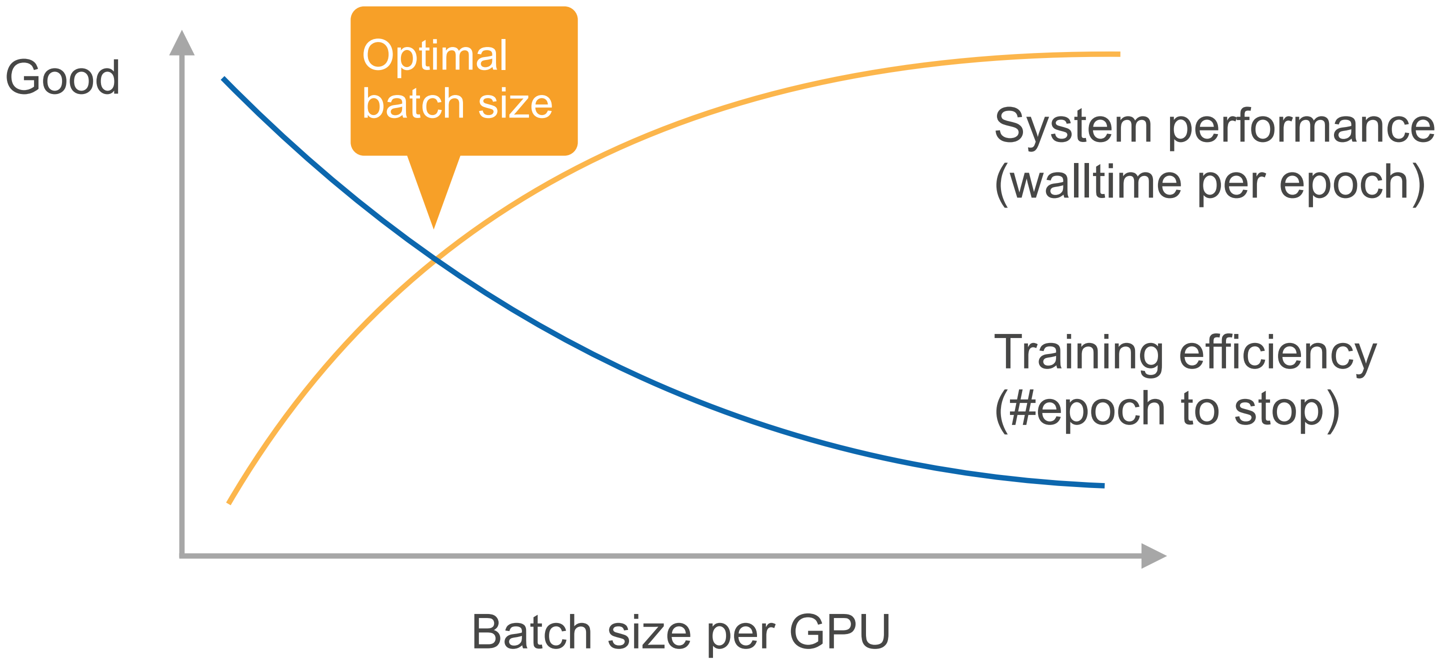
Synchronized SGD

- Each worker run synchronically
- If *n* GPUs and each GPU process *b* examples per time
    - Synchronized SGD equals to mini-batch SGD on a single GPU with a *nb* batch size
- In the ideal case, training with *n* GPUs will lead to a n times speedup compared to a single GPU training

Performance

- T1 = O(*b*): time to compute gradients for *b* example in a GPU
- T2 = O(*m*): time to send and receive *m* parameters/ gradients for a worker
- Wall-time for each batch is max(T1, T2)
    - Idea case: T1 > T2, namely using large enough *b*
- A too large *b* needs more data epochs to reach a desired model quality

Performance Trade-off



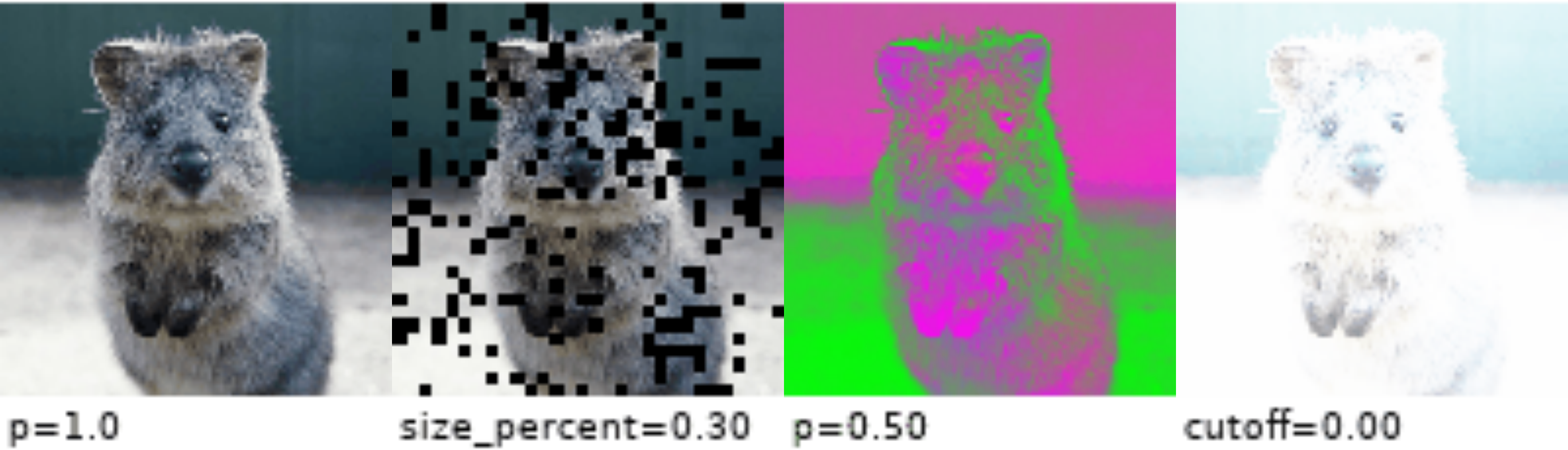
http://courses.d2l.ai/odsc2019/

Practical Suggestions

- A large dataset
- Good GPU-GPU and machine-machine bandwidth
- Efficient data loading/preprocessing
- A model with good computation (FLOP) vs communication (model size) ratio
    - ResNet > AlexNet
- A large enough batch size for good system performance
- Tricks for efficiency optimization with a large batch size

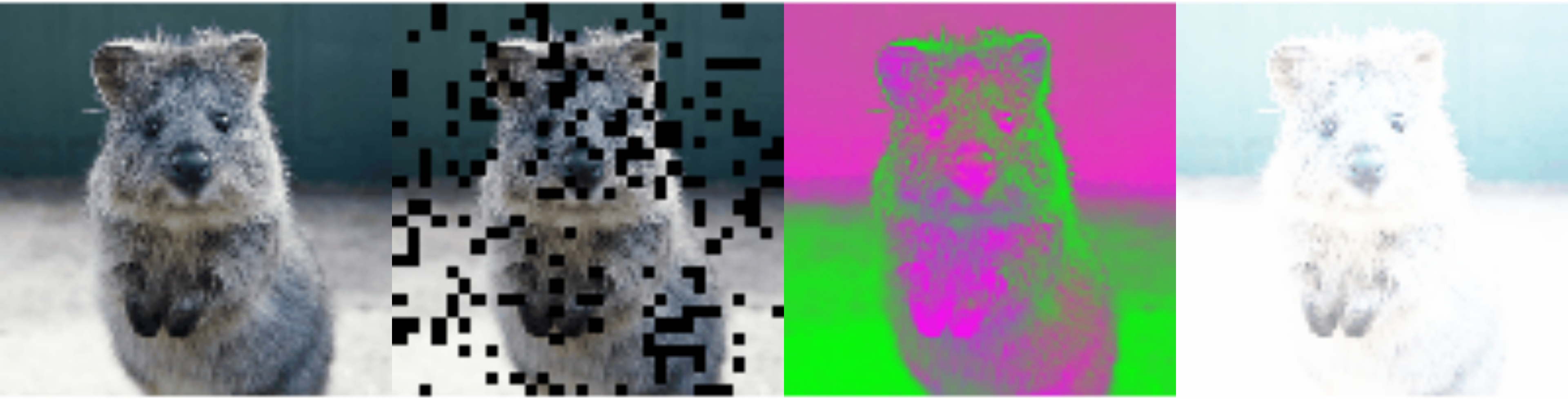
Multi-GPU Notebooks

Image Augmentation



http://courses.d2l.ai/odsc2019/

Image Augmentation



p=1.0 size_percent=0.30 p=0.50 cutoff=0.00

http://courses.d2l.ai/odsc2019/

Real Story from CES’19

- Startup with smart vending machine demo that identifies purchases via a camera
- Demo at CES failed
    - Different light temperature
    - Light reflection from table
- The fix
    - Collect new data
    - Buy tablecloth
    - Retrain all night

http://courses.d2l.ai/odsc2019/

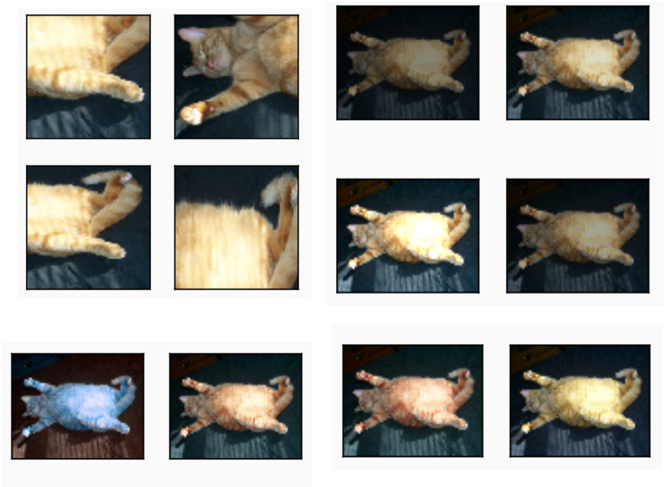
Data Augmentation

- Use prior knowledge about invariances to augment data
  - Add background noise to speech
  - Transform / augment image by altering colors, noise, cropping, distortions

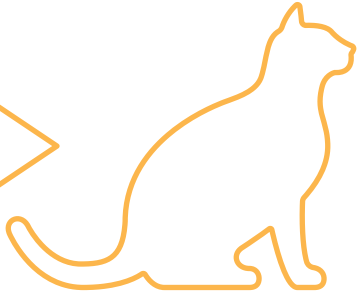
Training with Augmented Data

Original

Augmented Dataset



Model



Generate on the fly

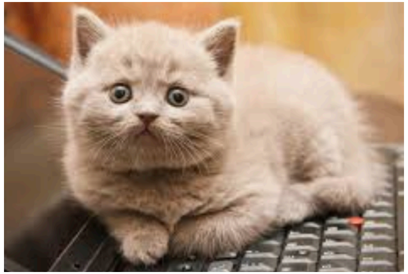
http://courses.d2l.ai/odsc2019/

aws

Flip

vertical

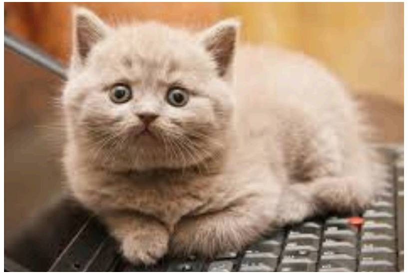
horizontal



Flip

vertical

horizontal

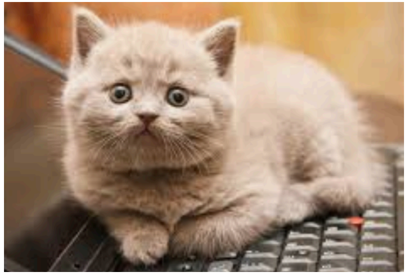


http://courses.d2l.ai/odsc2019/

Flip

vertical

horizontal



http://courses.d2l.ai/odsc2019/

Crop

- Crop an area from the image and resize it
  - Random aspect ratio (e.g. [3:4, 4:3])
  - Random area size (e.g. [8%, 100%])
  - Random position

Crop

- Crop an area from the image and resize it
  - Random aspect ratio (e.g. [3:4, 4:3])
  - Random area size (e.g. [8%, 100%])
  - Random position

http://courses.d2l.ai/odsc2019/

Color

Scale hue, saturation, and brightness (e.g. [0.5, 1.5])

Color

Scale hue, saturation, and brightness (e.g. [0.5, 1.5])

Brightness

Hue

http://courses.d2l.ai/odsc2019/

Many Other Augmentations

Many Other Augmentations

https://github.com/aleju/imgaug

Fine Tuning

Labelling a Dataset is Expensive

# examples	1.2M	50K	60K
# classes	1,000	100	10

My dataset

http://courses.d2l.ai/odsc2019/

Labelling a Dataset is Expensive

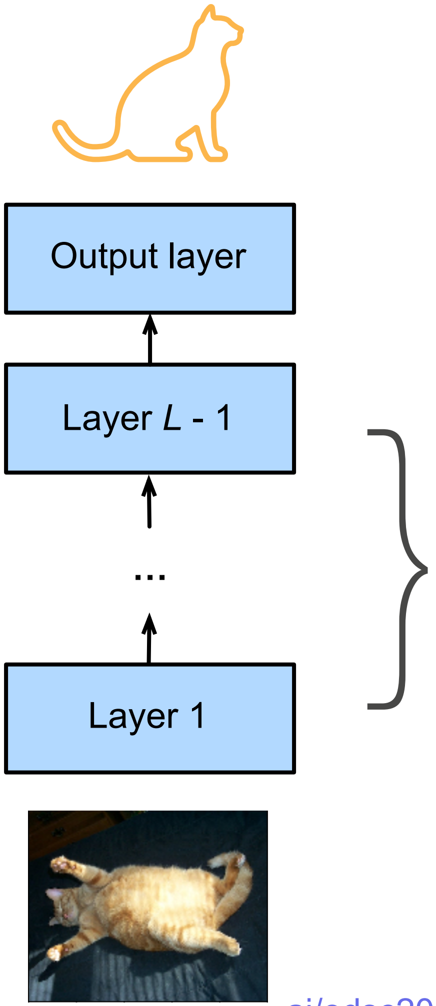
# examples	1.2M	50K	60K
# classes	1,000	100	10

Can we reuse this?

My dataset

http://courses.d2l.ai/odsc2019/

Network Structure



Softmax classifier

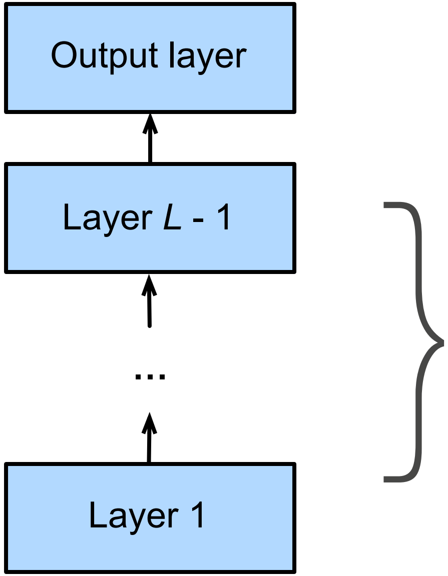
Feature extractor

Two components in deep network

- **Feature extractor to map raw pixels into linearly separable features.**
- Linear classifier for decisions

.ai/odsc2019/

Fine Tuning



Don't use last layer since classification problem is different

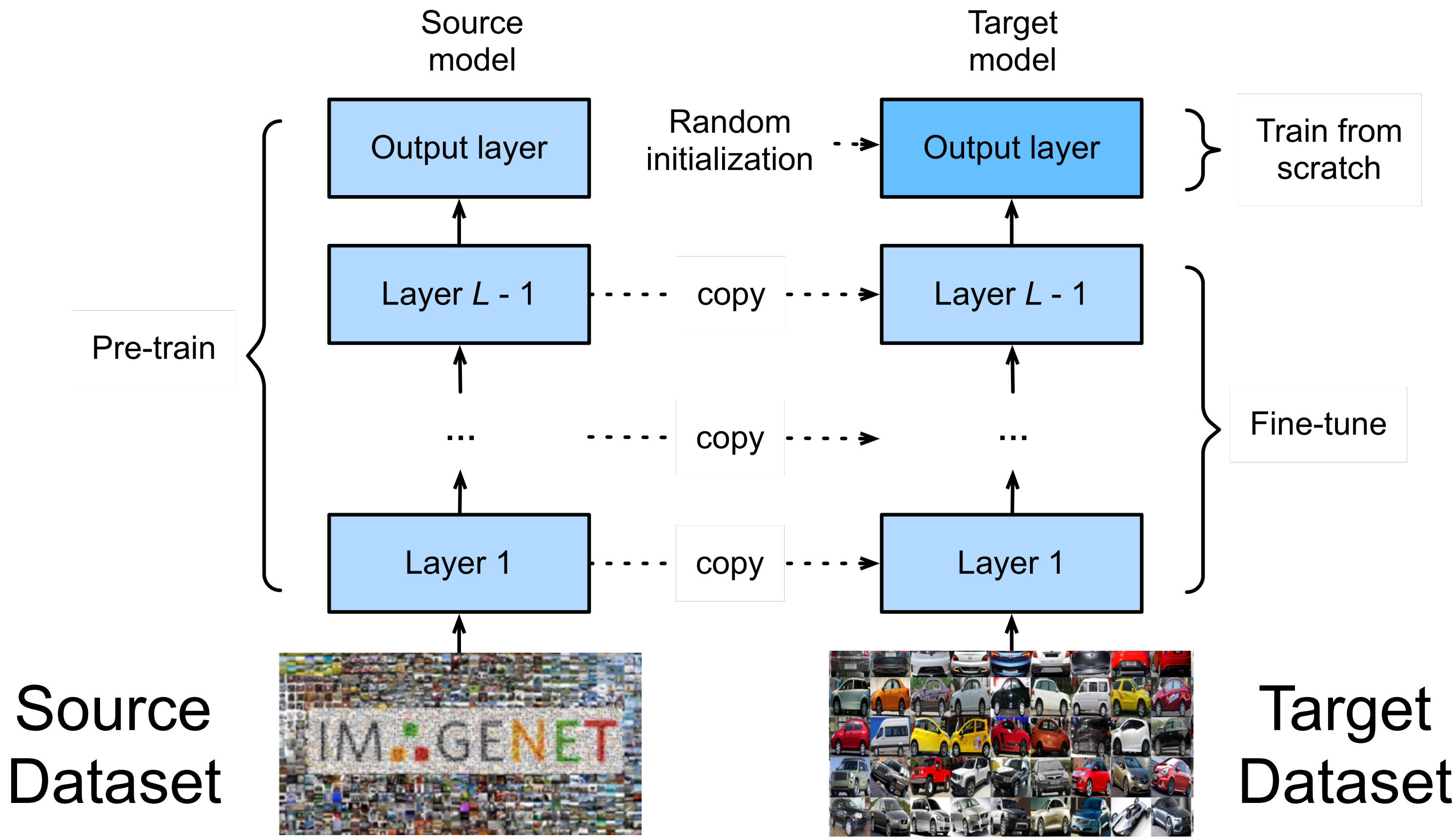
Likely good feature extractor for target

Source Dataset

Target Dataset

http://courses.d2l.ai/odsc2019/

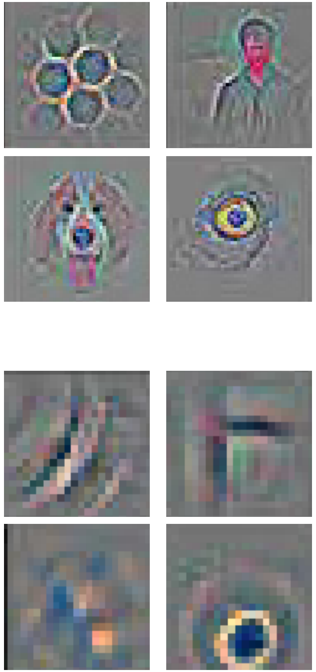
Weight Initialization for Fine Turning

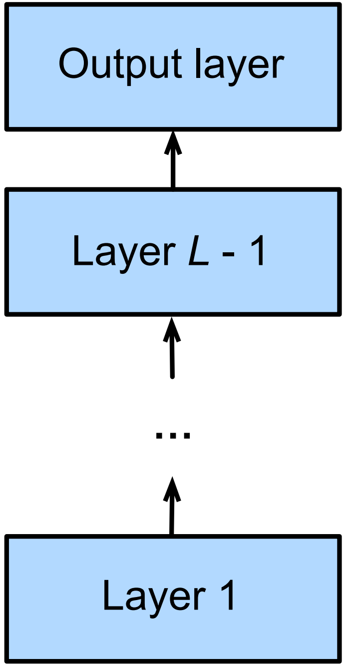


http://courses.d2l.ai/odsc2019/

Fix Lower Layers

- Neural networks learn hierarchical feature representations
  - Low-level features are universal
  - High-level features are more related to objects in the dataset
- **Fix the bottom layer parameters during fine tuning**
  (useful for regularization)

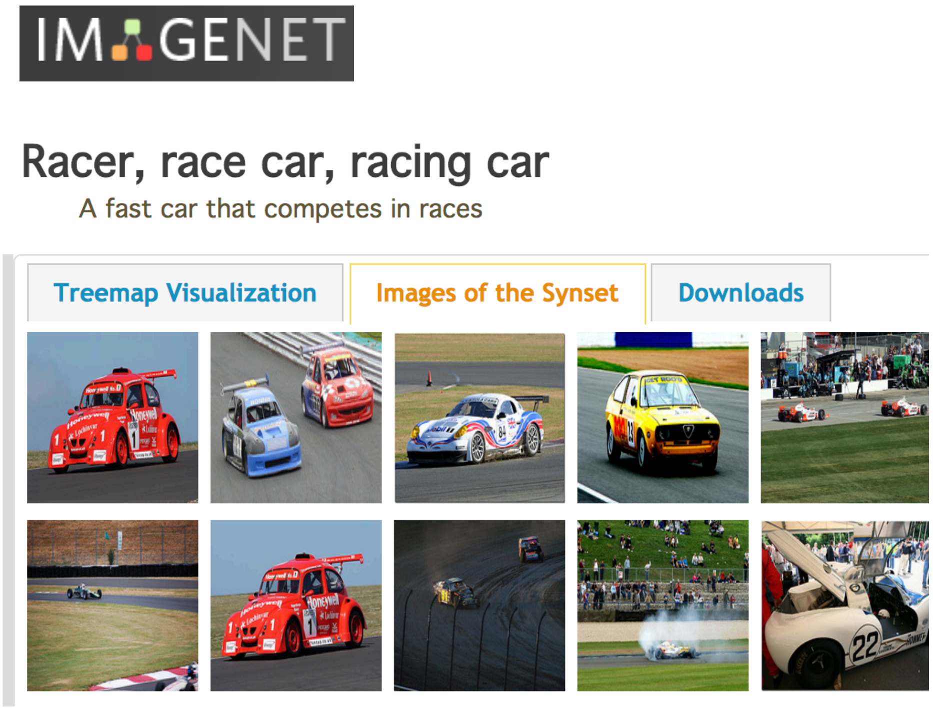




Re-use Classifier Parameters

Lucky break

- Source dataset may contain some of the target categories
- Use the according weight vectors from the pre-trained model during initialization



Fine-tuning Training Recipe

- Train on the target dataset as normal but with strong regularization
    - Small learning rate
    - Fewer epochs
- If source dataset is more complex than the target dataset, fine-tuning can lead to better models (source model is a good prior)

Fine-tuning Notebook

Summary

- To get good performance:
  - Optimize codes through hybridization
  - Use multiple GPUs/machines
- Augment image data by transformations
- Train with pre-trained models