

Dive into Deep Learning in 1 Day

1 Basics · 2 Convnets · 3 Computation · 4 Sequences

AMLC 2019

Mu Li and Alex Smola

<http://amlc-d2l.corp.amazon.com/>

Outline

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

A Hybrid of Imperative and Symbolic Programming



Imperative Programming

- The common way to program in Python, Java, C/C++, ...
- Straightforward, easy to debug
- Requires (Python) interpreter
 - Hard to deploy models
(smart phones, browser, embedded)
 - Performance problems

Interpreter compiles
into bytecode

Execute on virtual
machine

a = 1
b = 2
c = a + b

3 calls
in total

Symbolic Programming

- Define the program first, feed with data to execute later
- Math, SQL, ...
- Easy to optimize, less frontend overhead, portable
- Hard to use

May be used without Python interpreter

Know the whole program, easy to optimize

```
expr = "c = a + b"  
exec = compile(expr)  
exec(a=1, b=2)
```

Single call

Hybridization in Gluon

- Define a model through `nn.HybridSequential` or `nn.HybridBlock`
- Call `.hybridize()` to switch from imperative execution to symbolic execution

```
net = nn.HybridSequential()
net.add(nn.Dense(256, activation='relu'),
        nn.Dense(10))
net.hybridize()
```

Hybridize Notebook

<http://amlc-d2l.corp.amazon.com/>



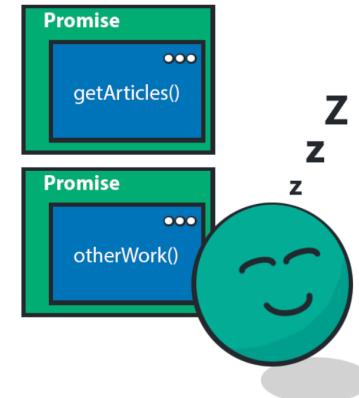
Asynchronous Computing

Synchronous



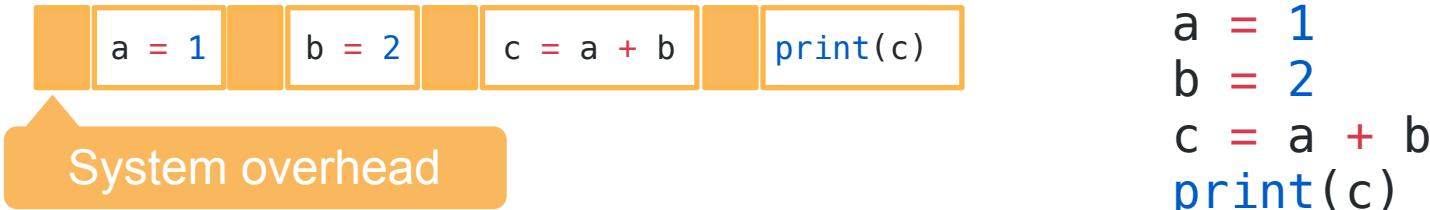
Asynchronous

VS

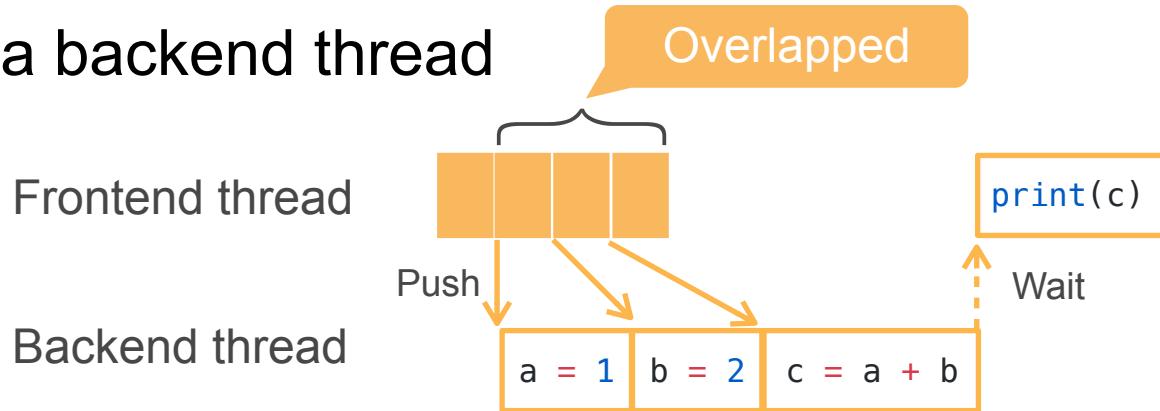


Asynchronous Execution

- Execute one-by-one



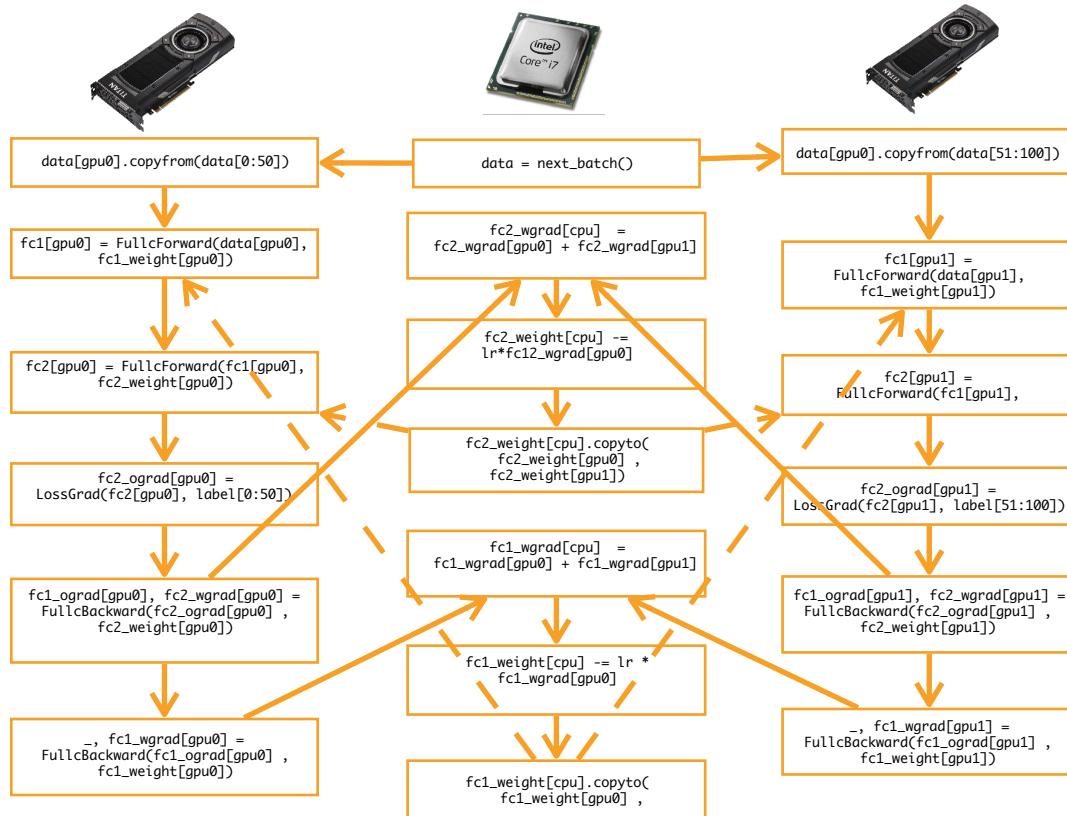
- With a backend thread



Automatic Parallelism



Writing Parallel Program is Painful



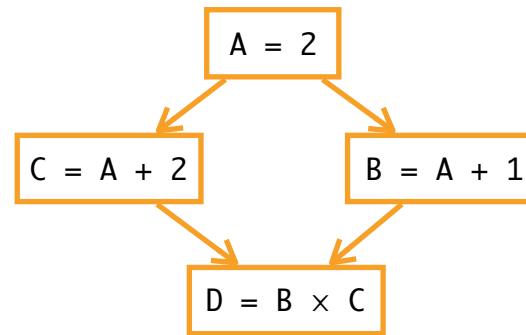
- Single hidden-layer MLP with 2 GPUs
- Scales to hundreds of layers and tens of GPUs

Auto Parallelization

Run in parallel

Write serial programs

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



Multi-GPU Training

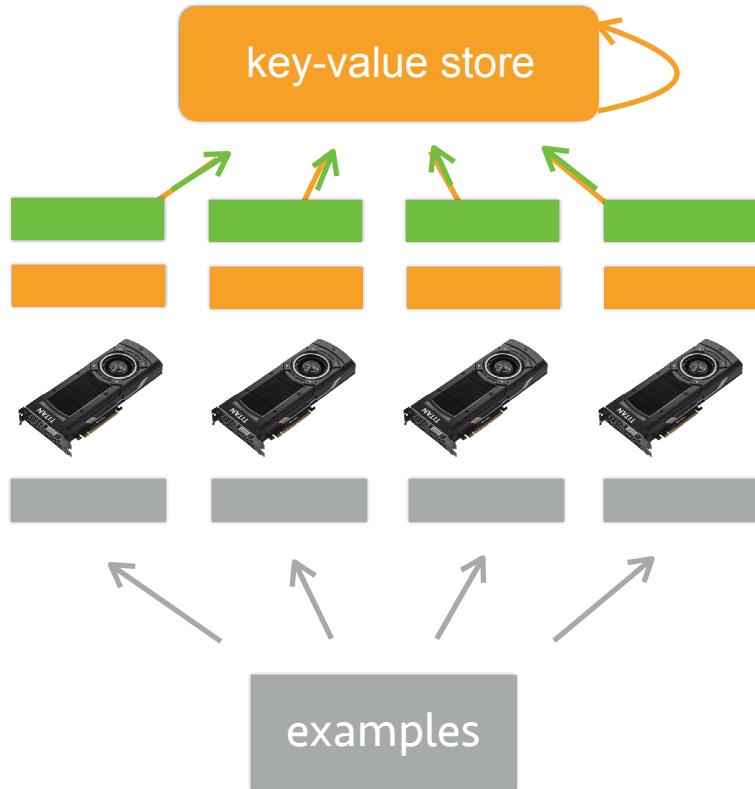


(Lunar new year, 2014)

<http://amlc-d2l.corp.amazon.com/>

aws

Data Parallelism



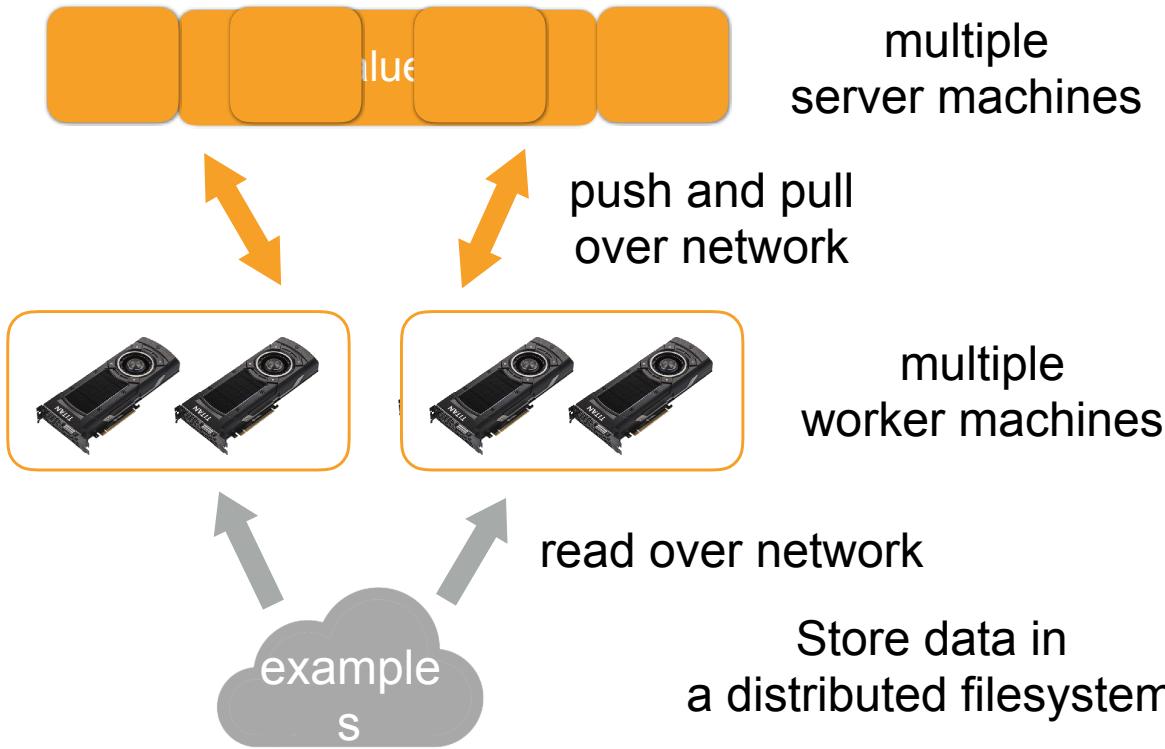
1. Read a data partition
2. Pull the parameters
3. Compute the gradient
4. Push the gradient
5. Update the parameters

Distributed Training

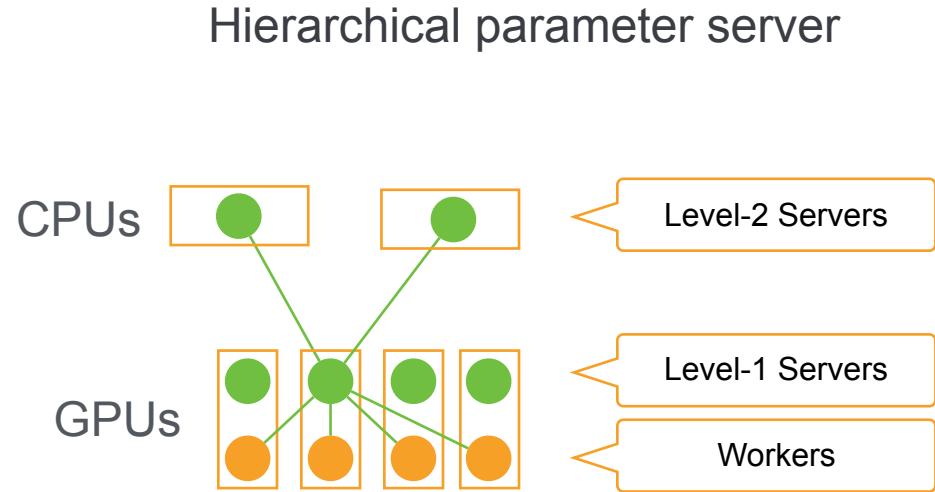
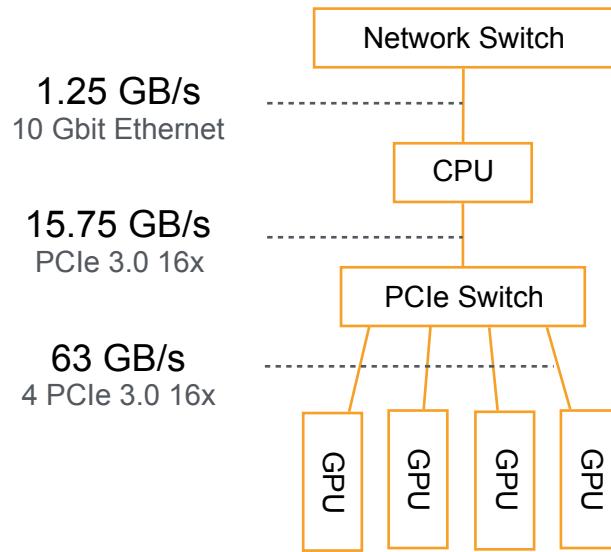


(Alex's frugal GPU cluster at CMU, 2015)

Distributed Computing

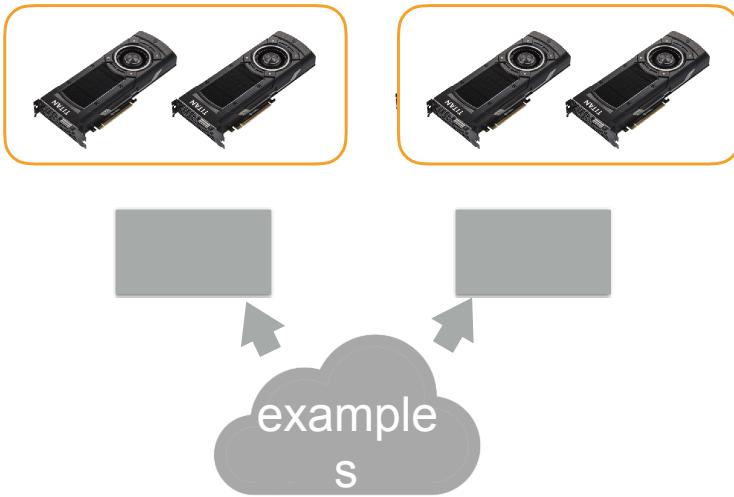


GPU Machine Hierarchy



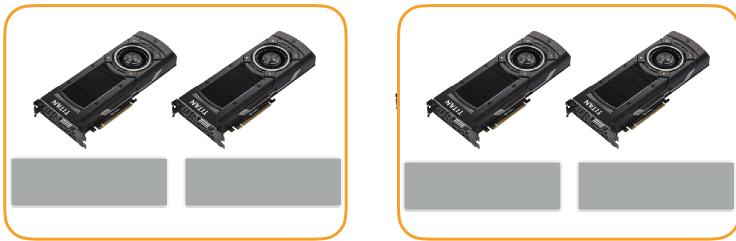
Iterating a Batch

- Each worker machine read a part of the data batch

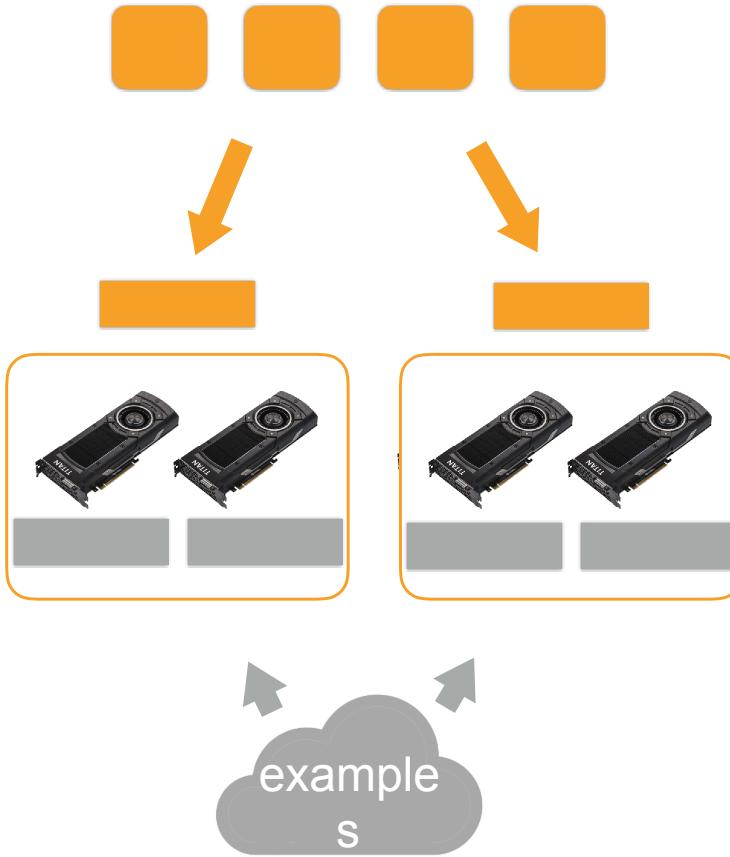


Iterating a Batch

- Further split and move to each GPU



Iterating a Batch

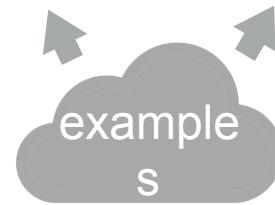
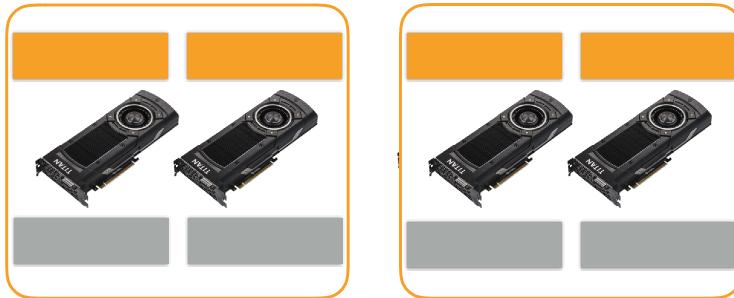


- Each server maintain a part of parameters
- Each worker pull the whole parameters from servers

Iterating a Batch



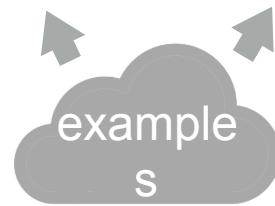
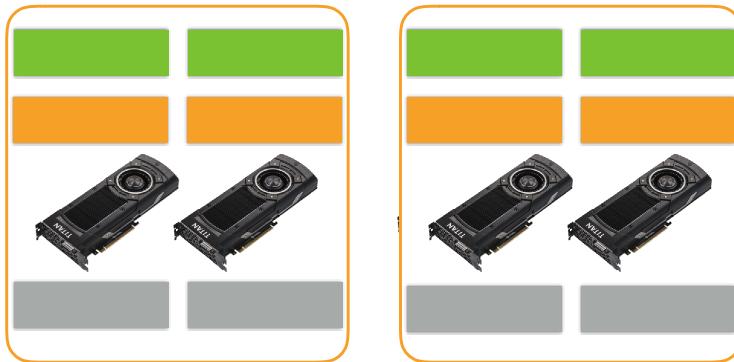
- Copy parameters into each GPU



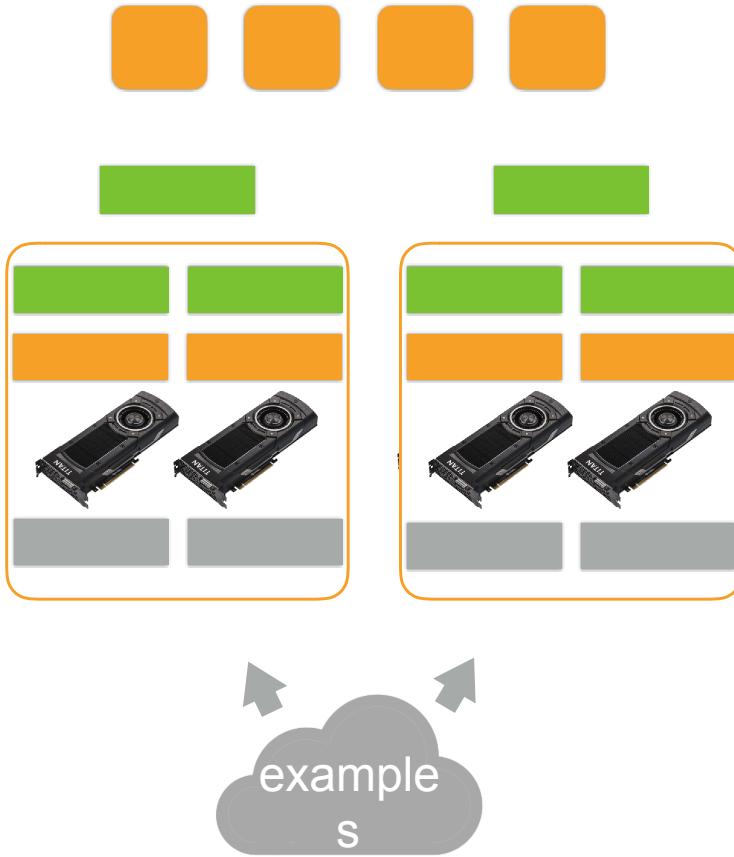
Iterating a Batch



- Each GPU computes gradients

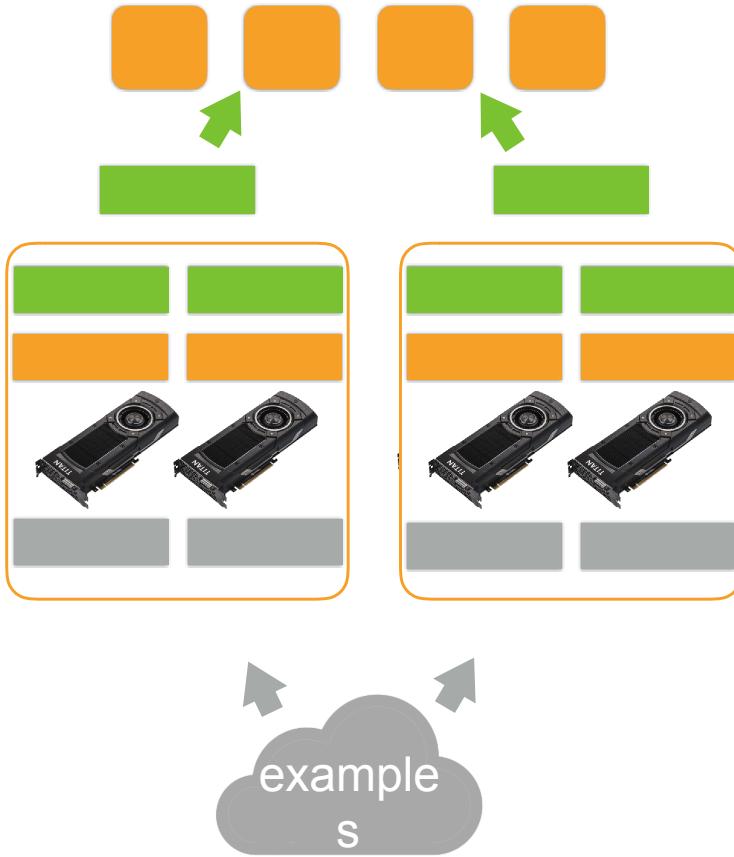


Iterating a Batch



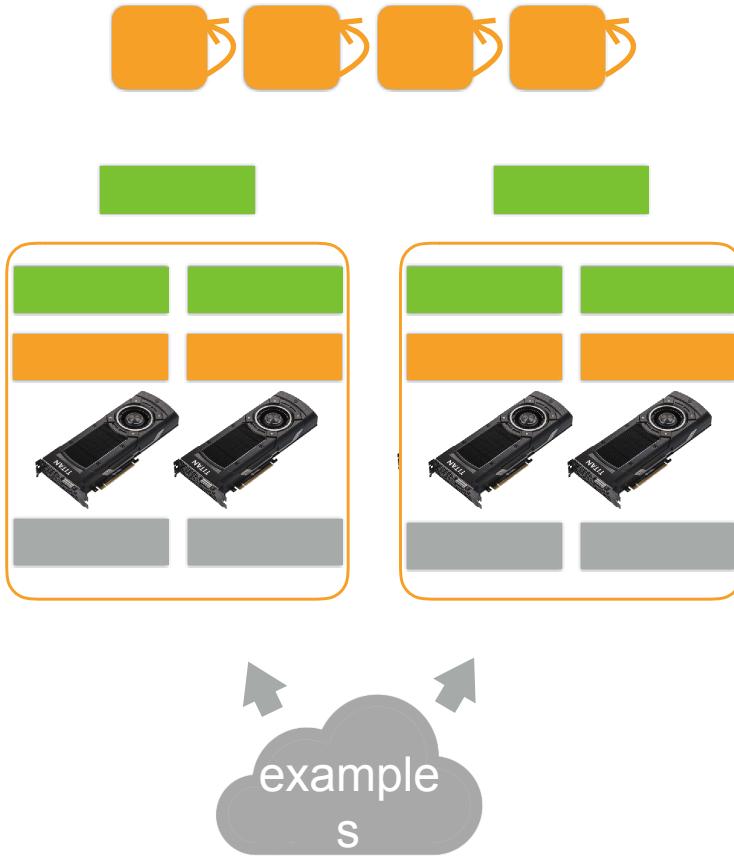
- Sum the gradients over all GPU

Iterating a Batch



- Push gradients into servers

Iterating a Batch



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

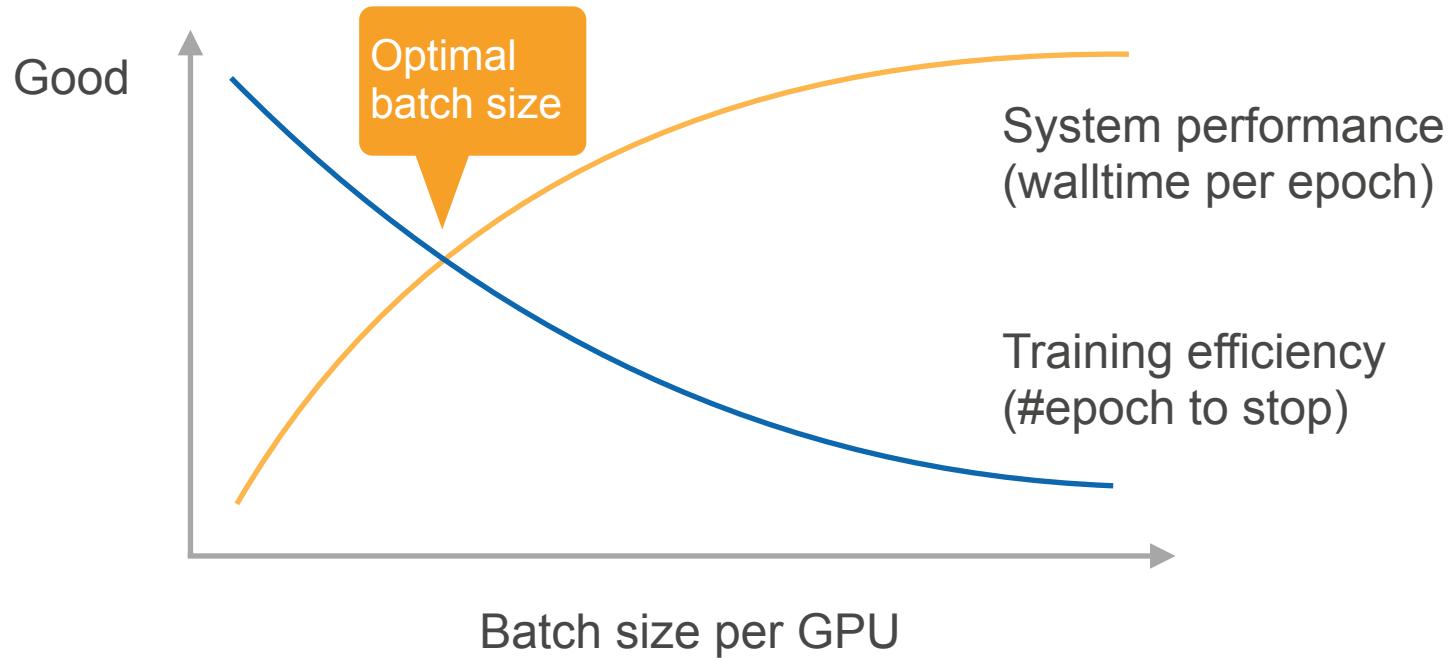
Synchronized SGD

- 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

- $T_1 = O(b)$: time to compute gradients for b example in a GPU
- $T_2 = O(m)$: time to send and receive m parameters/ gradients for a worker
- Wall-time for each batch is $\max(T_1, T_2)$
 - Idea case: $T_1 > T_2$, namely using large enough b
- A too large b needs more data epochs to reach a desired model quality

Performance Trade-off



Practical Suggestions

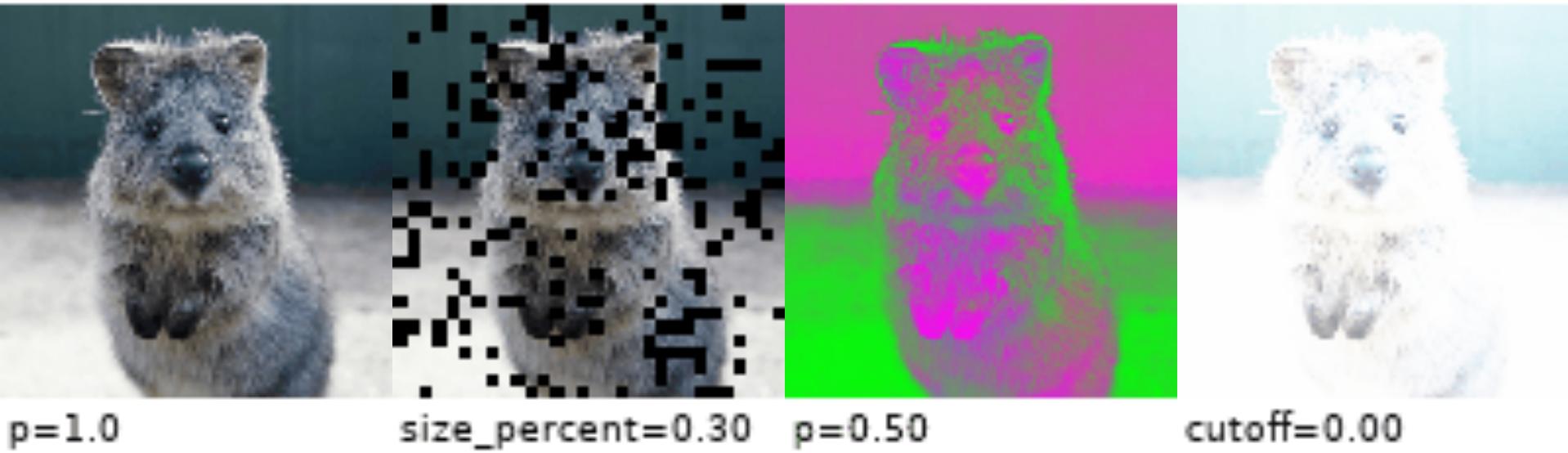
- A large dataset
- Good GPU-GPU and machine-machine bandwidth
- Efficient data loading/preprocessing
- A model with good computation (FLOP) vs communication (model size) ratio
 - ResNet > AlexNet
- A large enough batch size for good system performance
- Tricks for efficiency optimization with a large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

Multi-GPU Notebooks

<http://amlc-d2l.corp.amazon.com/>



Image Augmentation



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



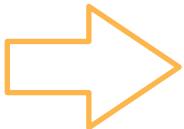
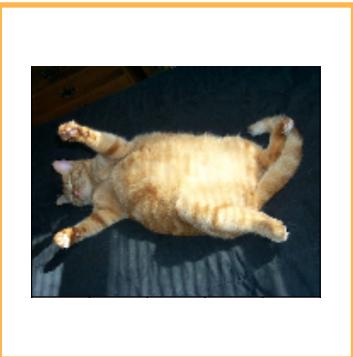
Data Augmentation



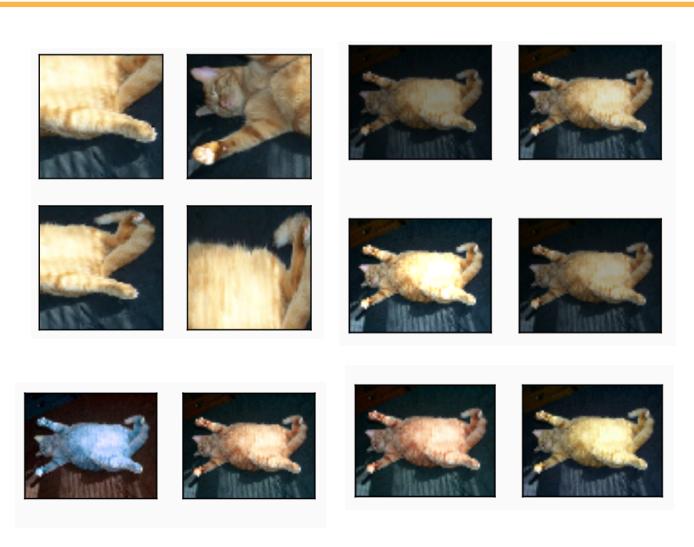
- Use prior knowledge about invariances to augment data
 - Add background noise to speech
 - Transform / augment image by altering colors, noise, cropping, distortions

Training with Augmented Data

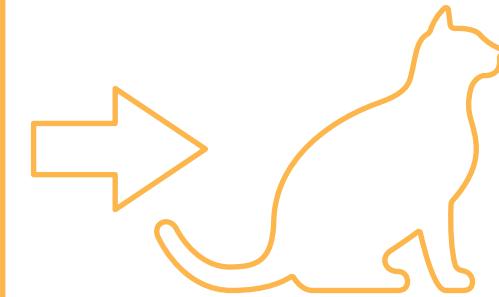
Original



Augmented Dataset



Model



Generate on the fly

Flip

vertical

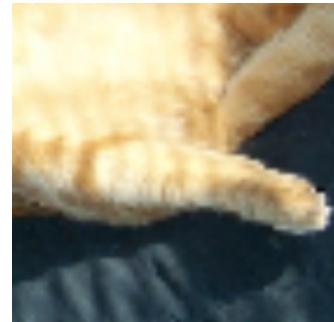


horizontal



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



Color

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

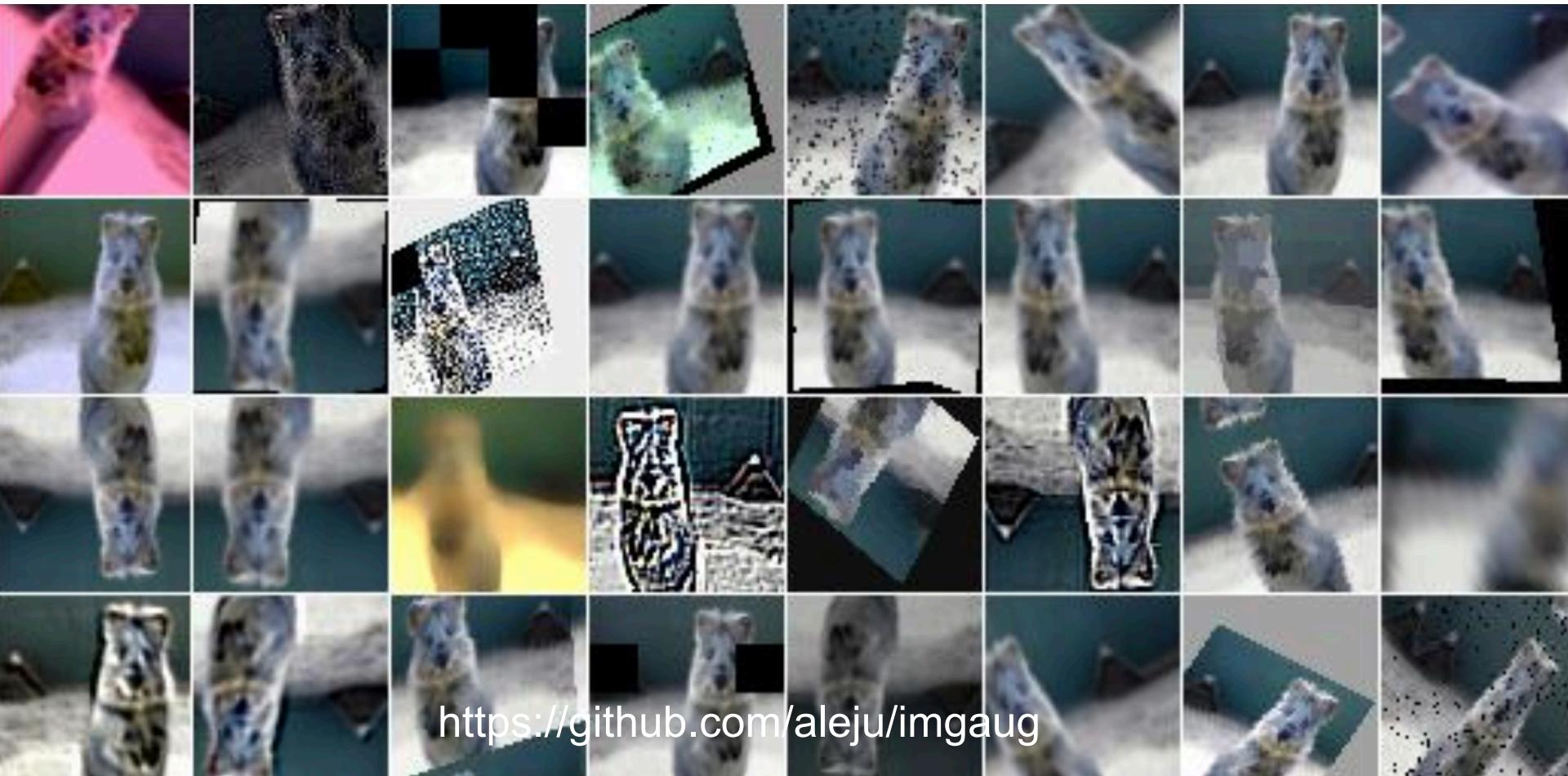


Brightness



Hue

Many Other Augmentations



<https://github.com/aleju/imgaug>

A close-up photograph of a person's hand reaching towards a row of numerous grey cylindrical knobs on a sound mixing console. The knobs are arranged in a grid pattern, with some having colored caps (yellow, blue, green, orange). The background is blurred, showing more of the console and other rows of knobs.

Fine Tuning

Labelling a Dataset is Expensive

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

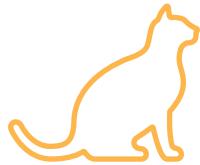


Can we
reuse this?

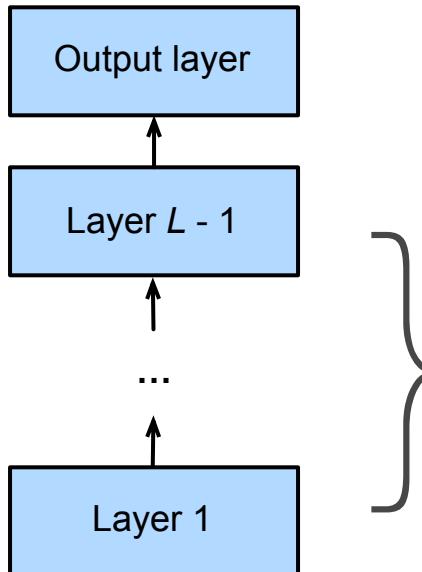


My dataset

2 2 2 2 2 2 2 2 2 2
3 3 3 3 3 3 3 3 3 3
4 4 4 4 4 4 4 4 4 4
5 5 5 5 5 5 5 5 5 5
6 6 6 6 6 6 6 6 6 6



Network Structure



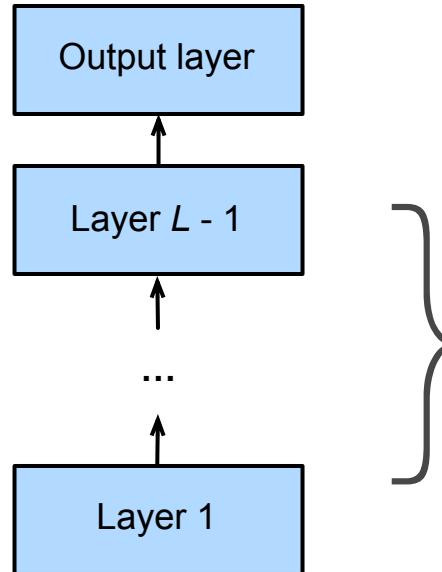
Softmax
classifier

Feature
extractor

Two components in
deep network

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

Fine Tuning



Don't use last layer
since classification
problem is different



Likely good feature
extractor for target

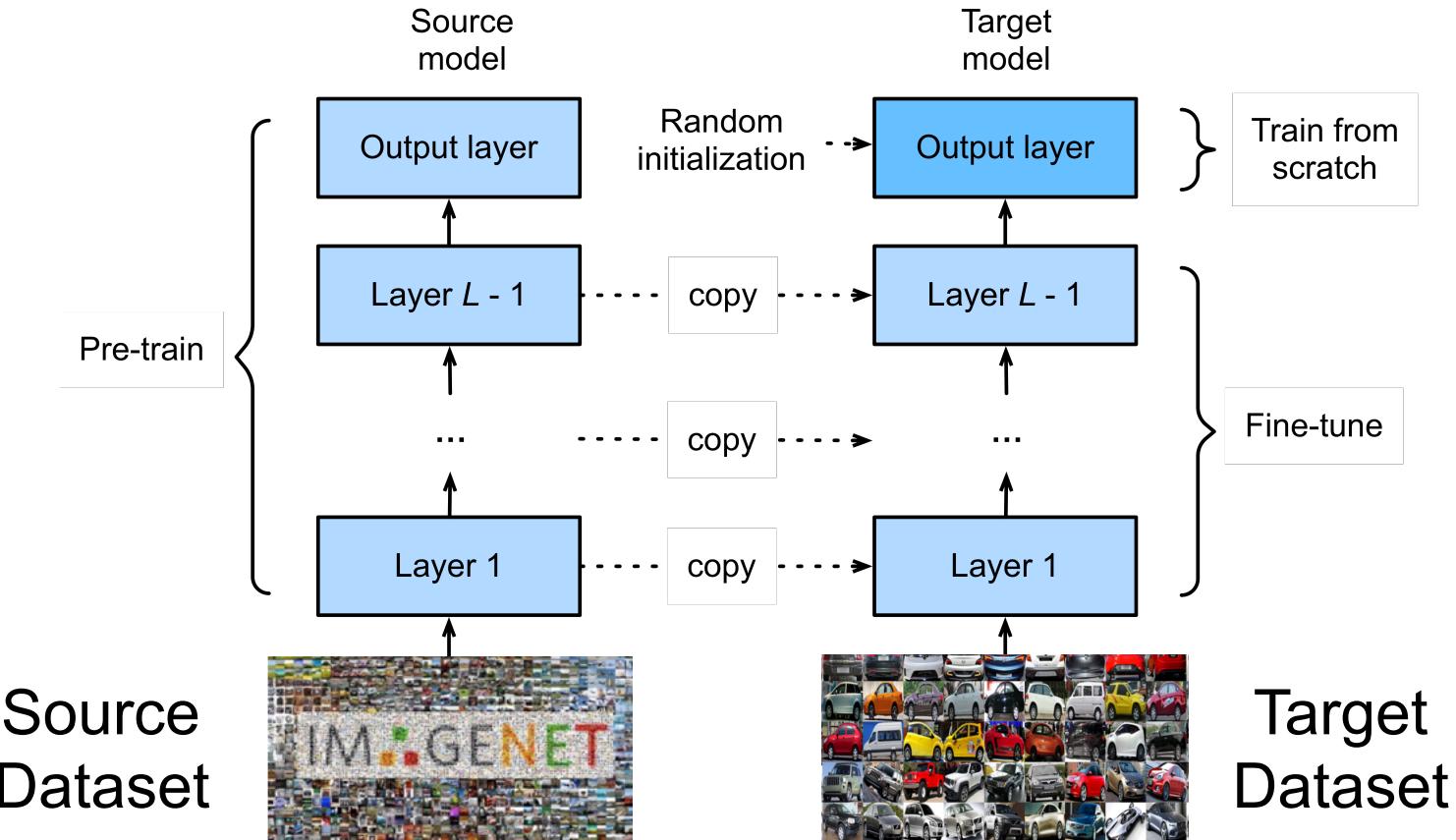


Source
Dataset



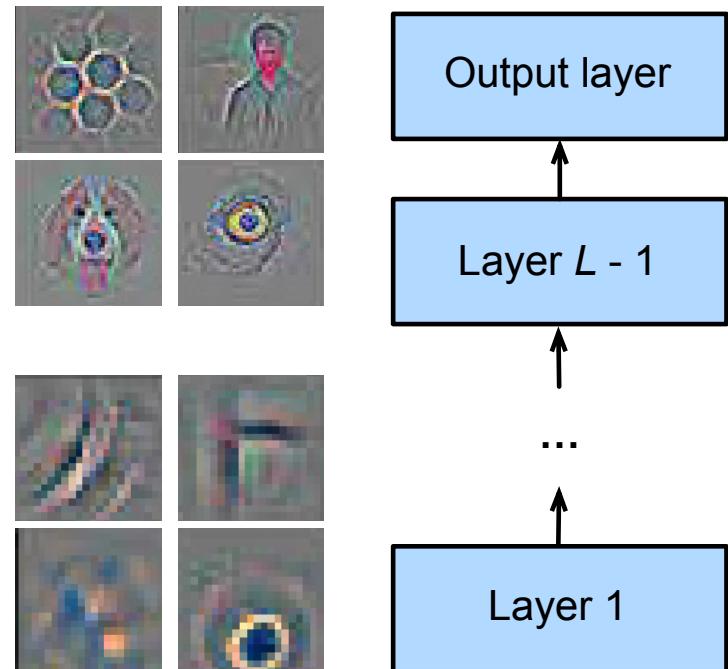
Target
Dataset

Weight Initialization for Fine Turning



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



Racer, race car, racing car

A fast car that competes in races



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