UT Southwestern Medical Center

Lyda Hill Department of Bioinformatics

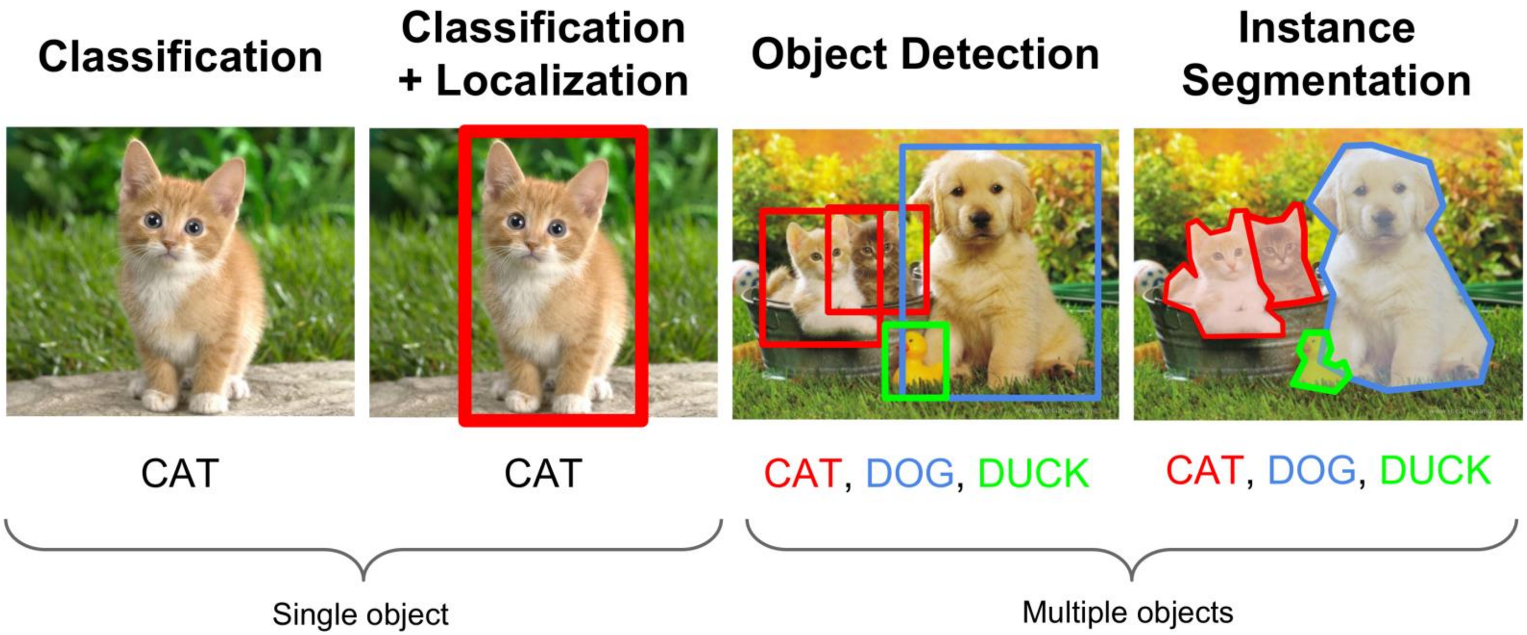
BioHPC

Distributed Deep Learning

BioHPC – 6/22/2022

Machine Learning on Images

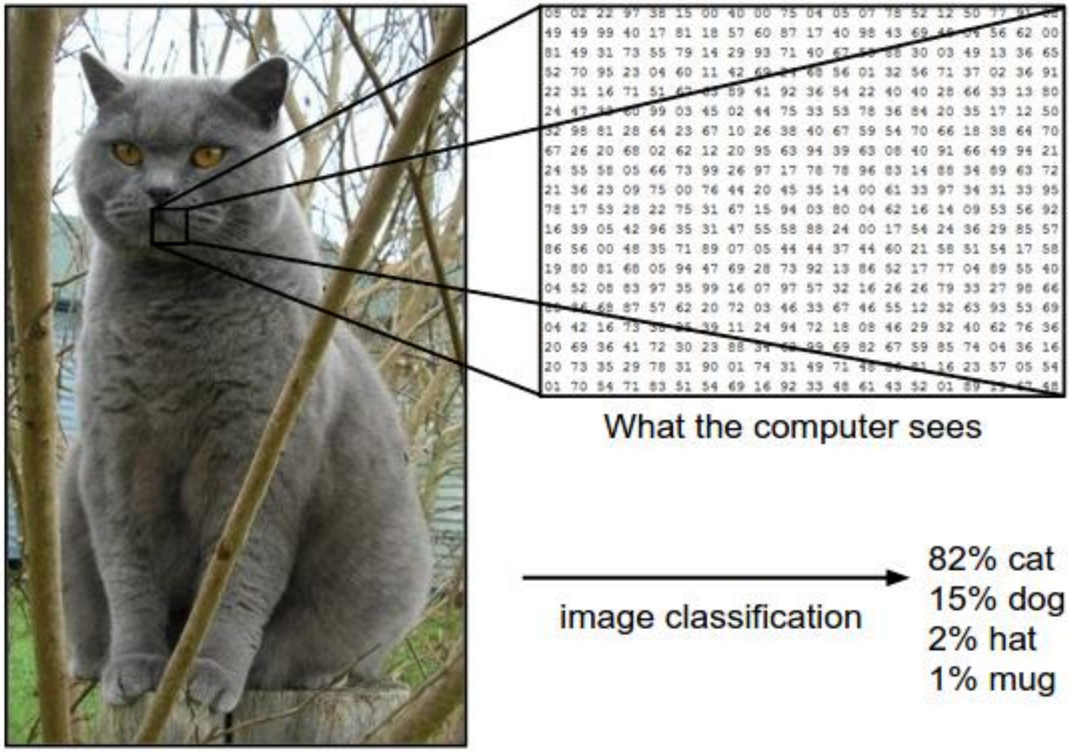
Given examples, can we train a computer to do:



We want the DL model to LEARN features! We want to TRAIN the DL model!

Source - http://cs231n.github.io/

Machine Learning on Images



```
08 02 22 97 38 15 00 40 00 75 04 05 07 78 52 12 50 77 91 08
49 49 99 40 17 81 18 57 60 87 17 40 98 43 69 48 04 56 62 00
81 49 31 73 55 79 14 29 93 71 40 67 53 88 30 03 49 13 36 65
52 70 95 23 04 60 11 42 69 24 68 56 01 32 56 71 37 02 36 91
22 31 16 71 51 67 63 89 41 92 36 54 22 40 40 28 66 33 13 80
24 47 32 60 99 03 45 02 44 75 33 53 78 36 84 20 35 17 12 50
32 98 81 28 64 23 67 10 26 38 40 67 59 54 70 66 18 38 64 70
67 26 20 68 02 62 12 20 95 63 94 39 63 08 40 91 66 49 94 21
24 55 58 05 66 73 99 26 97 17 78 78 96 83 14 88 34 89 63 72
21 36 23 09 75 00 76 44 20 45 35 14 00 61 33 97 34 31 33 95
78 17 53 28 22 75 31 67 15 94 03 80 04 62 16 14 09 53 56 92
16 39 05 42 96 35 31 47 55 58 88 24 00 17 54 24 36 29 85 57
86 56 00 48 35 71 89 07 05 44 44 37 44 60 21 58 51 54 17 58
19 80 81 68 05 94 47 69 28 73 92 13 86 52 17 77 04 89 55 40
04 52 08 83 97 35 99 16 07 97 57 32 16 26 26 79 33 27 98 66
88 36 68 87 57 62 20 72 03 46 33 67 46 55 12 32 63 93 53 69
04 42 16 73 38 25 39 11 24 94 72 18 08 46 29 32 40 62 76 36
20 69 36 41 72 30 23 88 34 62 99 69 82 67 59 85 74 04 36 16
20 73 35 29 78 31 90 01 74 31 49 71 48 86 81 16 23 57 05 54
01 70 54 71 83 51 54 69 16 92 33 48 61 43 52 01 89 19 67 48
```

What the computer sees

image classification

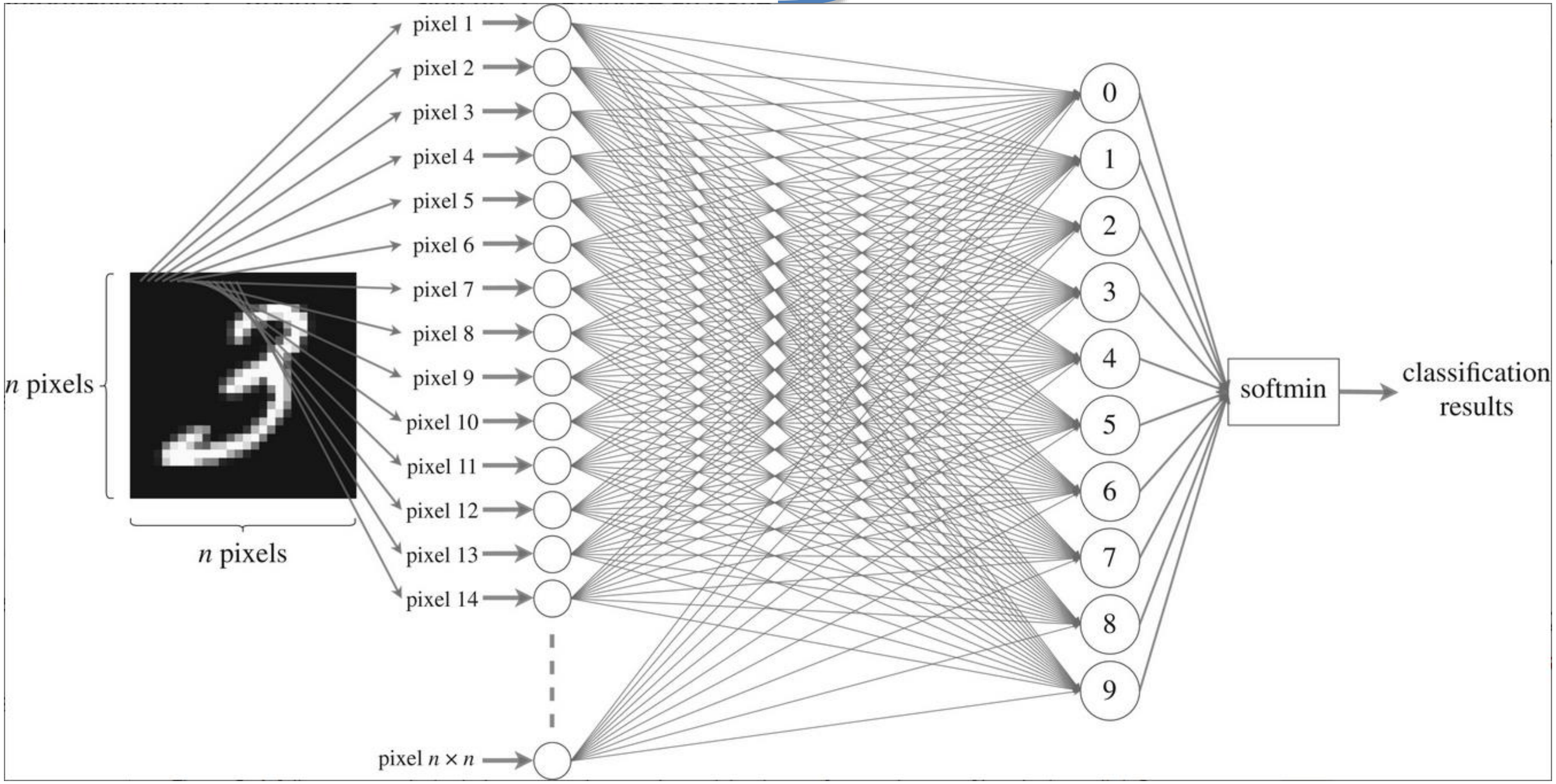
82% cat
15% dog
2% hat
1% mug

Source - http://cs231n.github.io/

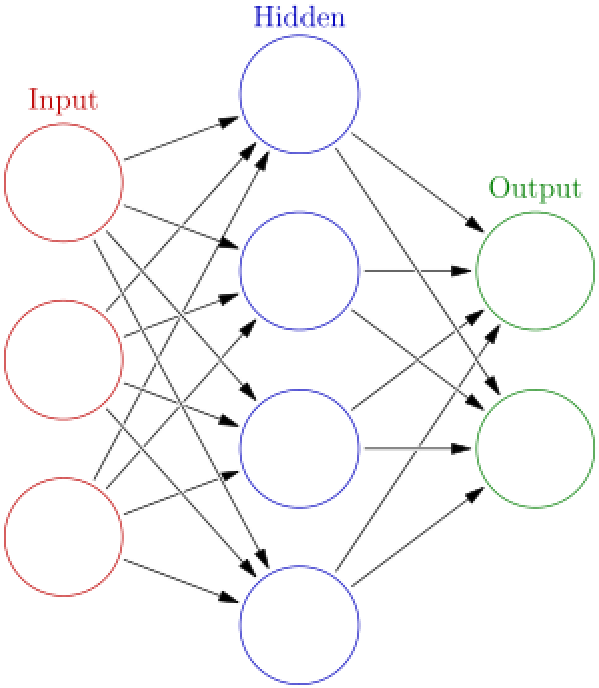
Artificial Neural Networks

- What are **neurons**?
- How are they **connected**?

Neural Networks



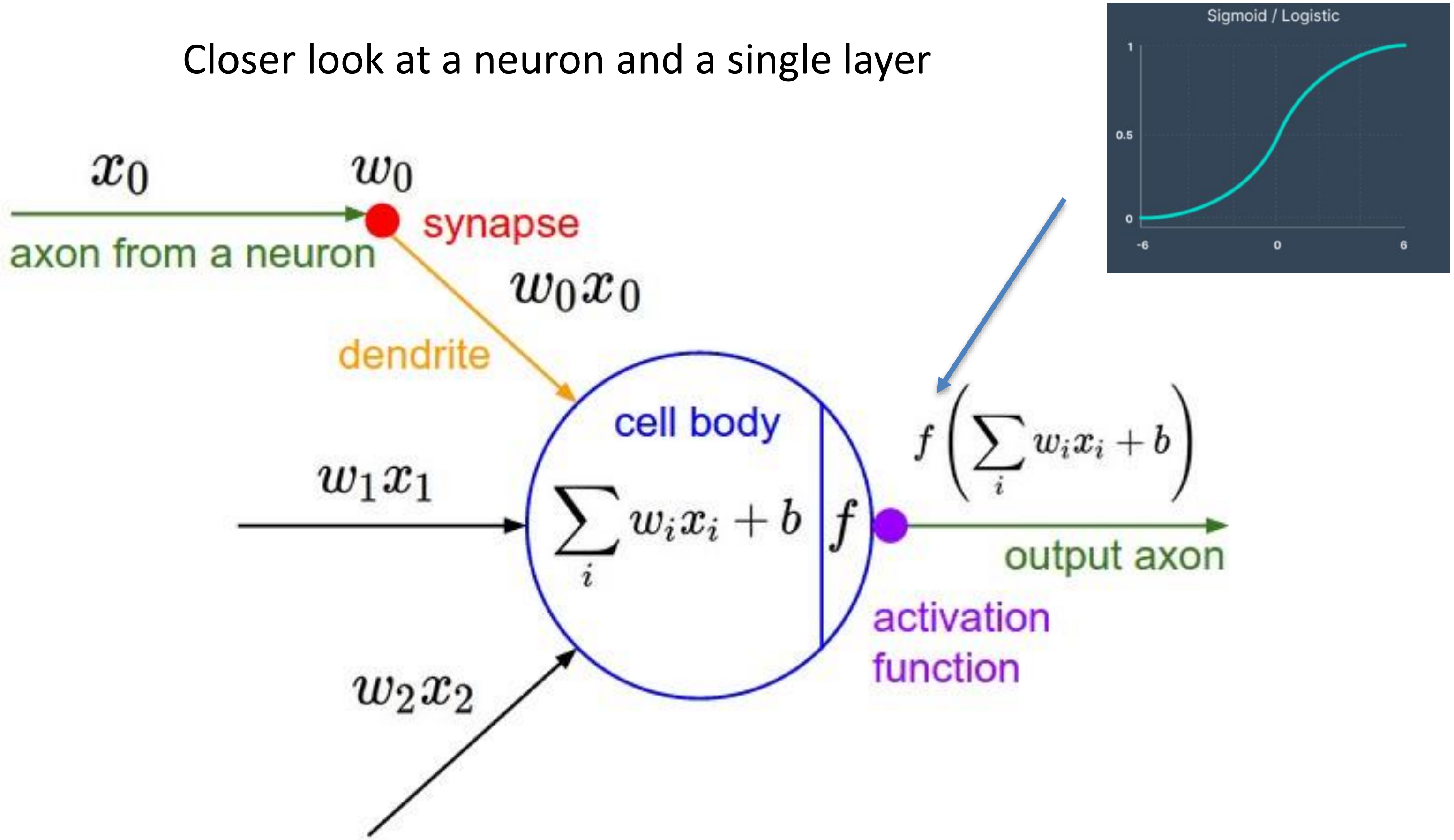
Artificial Neural Networks



- Signal goes in, via input layer
- Weighted links transfer input values to neurons in hidden layers
- Signals are summed at hidden neurons and passed through transfer/activation function
- Processed signal arrives at output layer
- Decisions made using output signal(s)

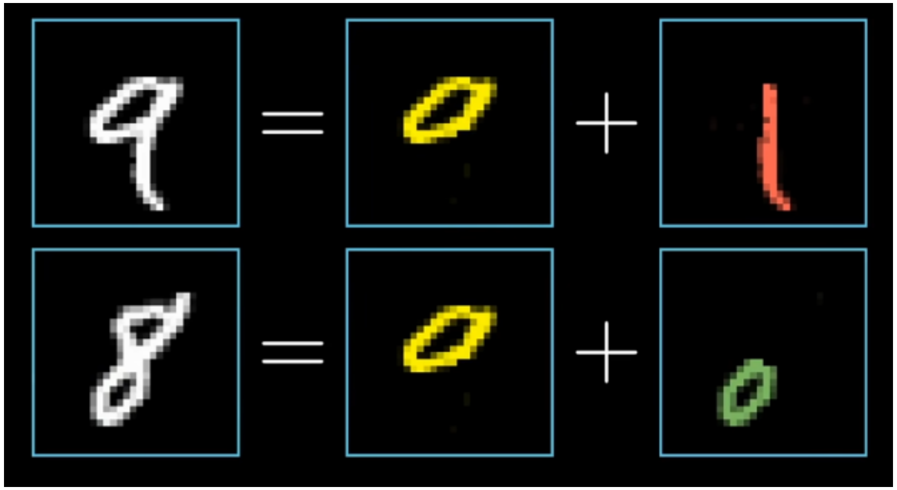
What’s in an (Artificial) Neuron?

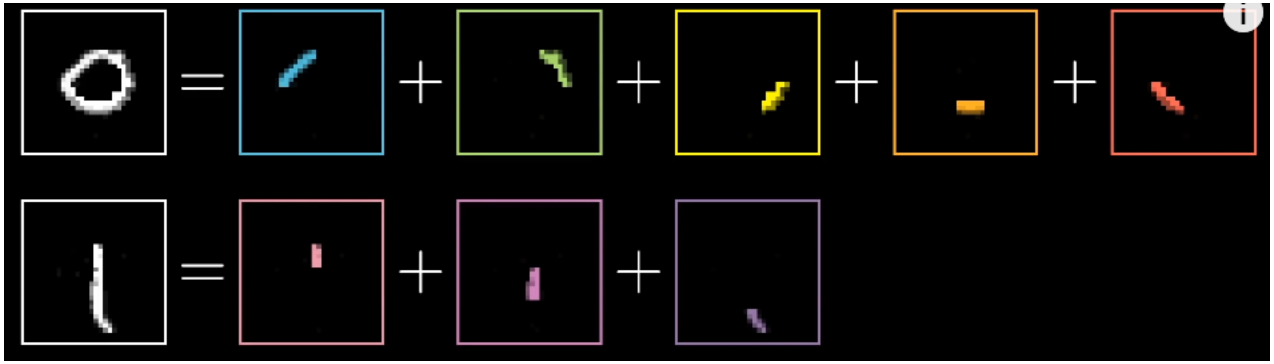
Closer look at a neuron and a single layer



Sources - http://cs231n.github.io/
https://www.v7labs.com/blog/neural-networks-activation-functions

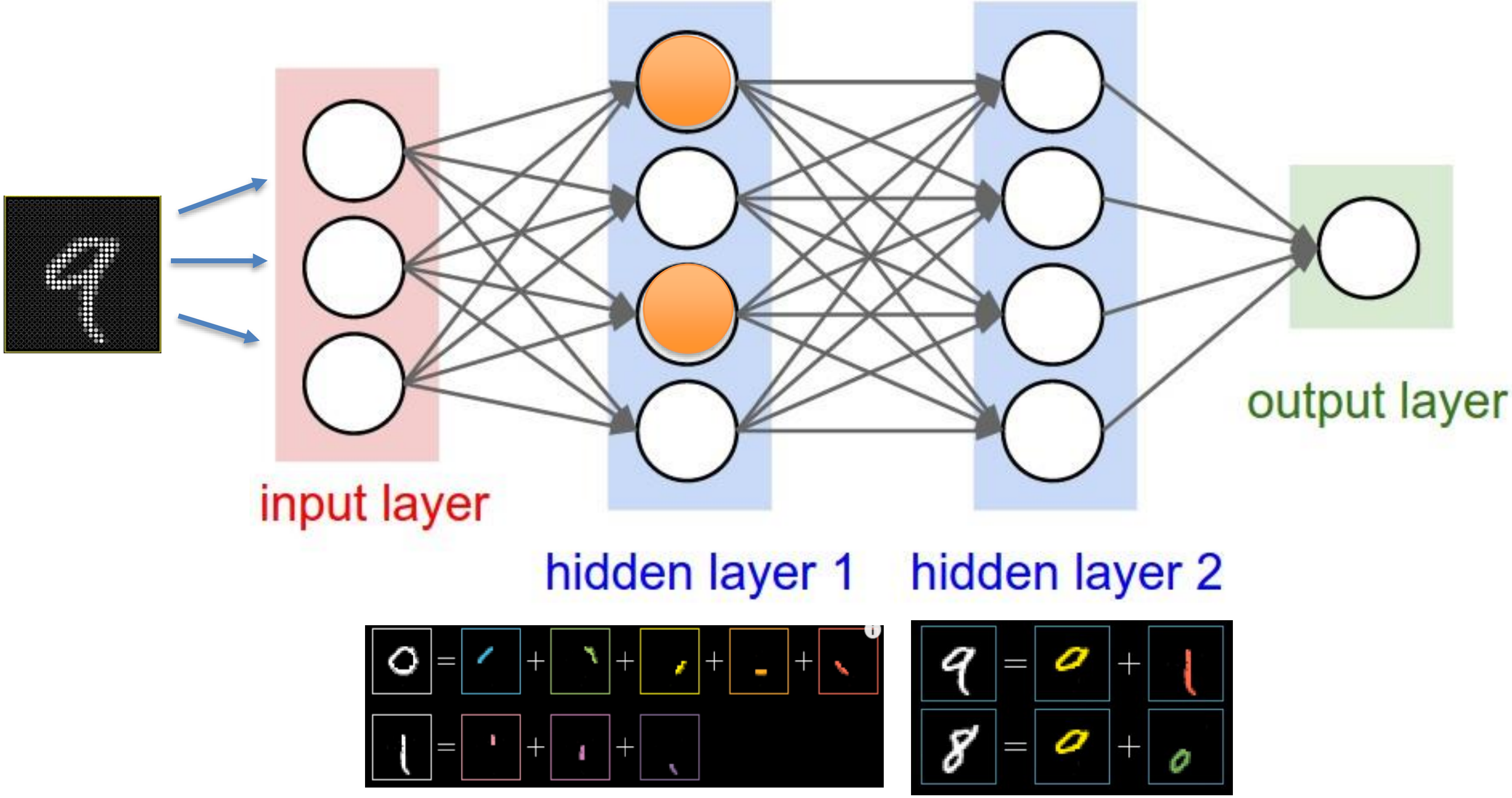
What do we expect from the hidden layers?





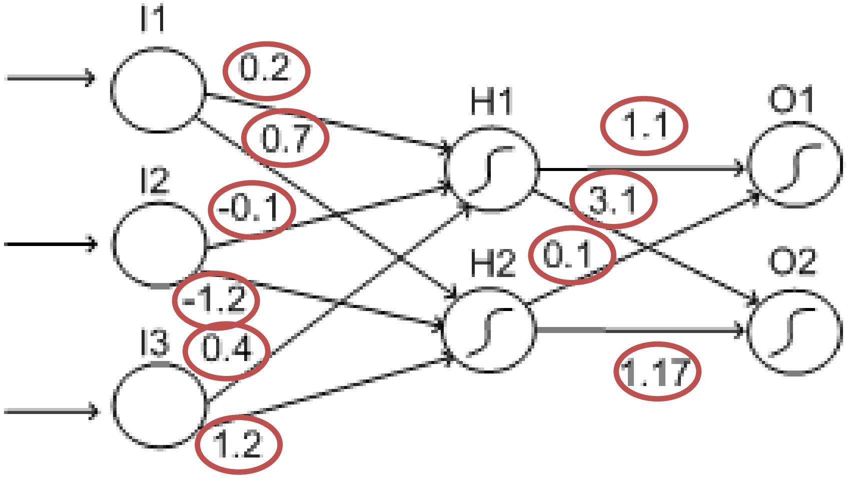
https://www.3blue1brown.com/lessons/neural-networks

What we expect from Deep Learning?



More layers can encapsulate more knowledge.
More weights to train – need more data, need more computation

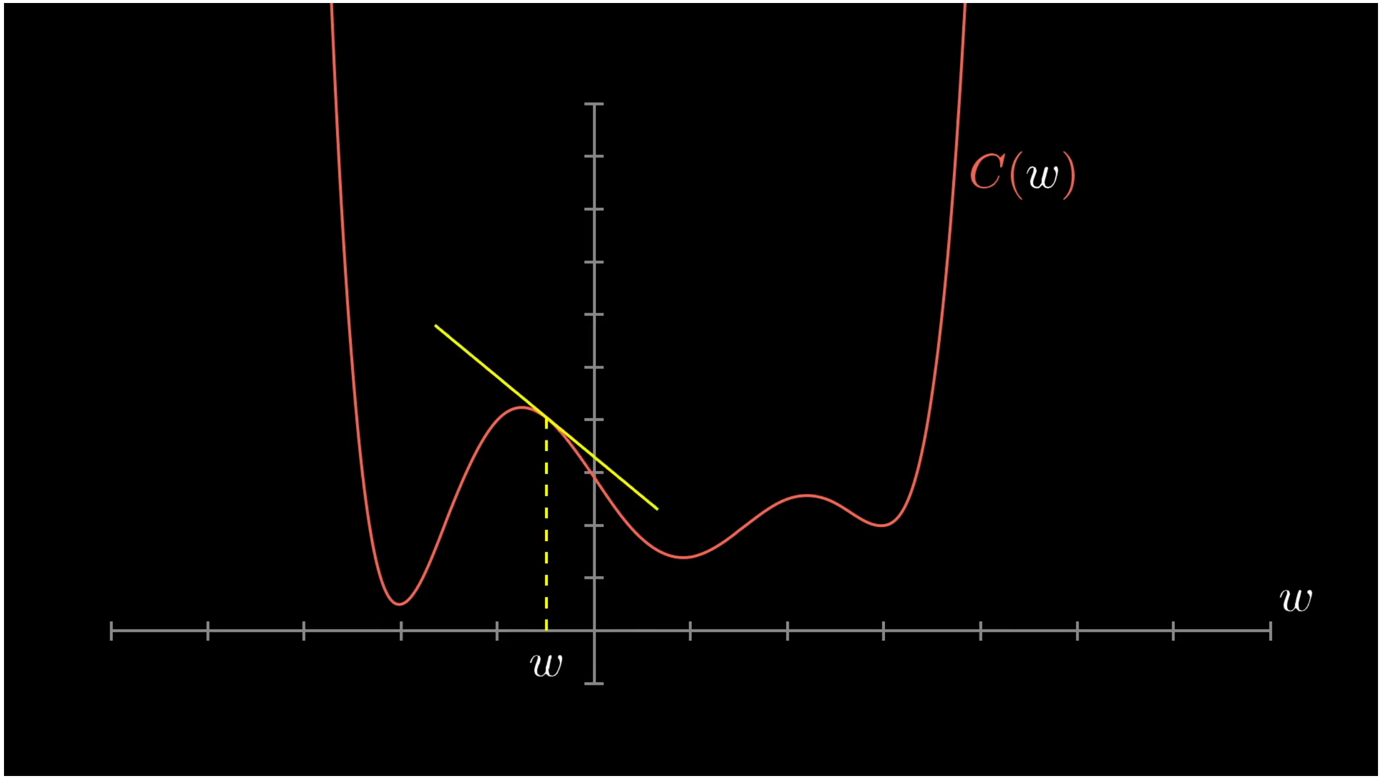
Where’s the Knowledge?



How the Neural Network learns?

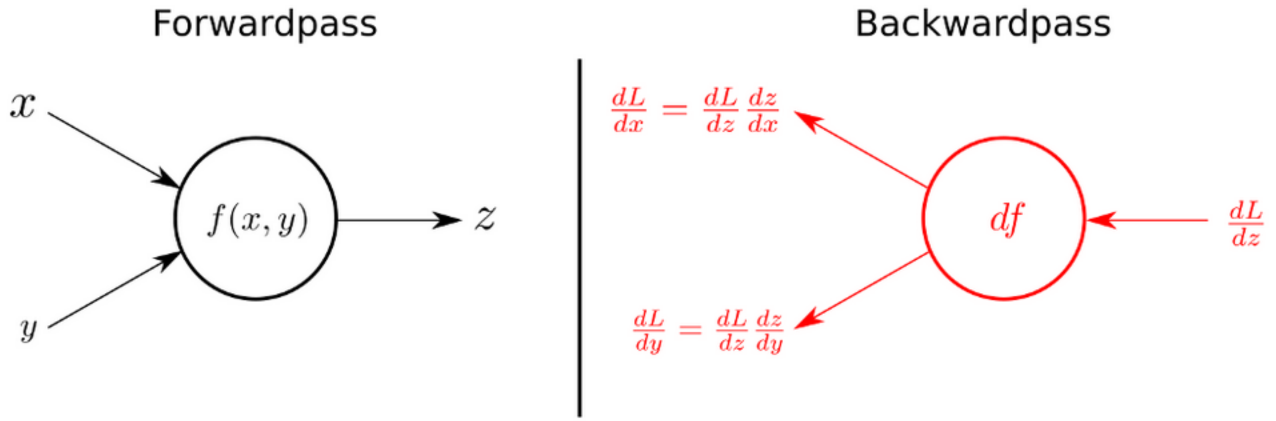
- Weights encapsulate the knowledge of a network
- Network learns using an algorithm that optimize weights given **training data.**
- Minimize cost function
  - Cost function: summation over the differences between the NN prediction and the actual value of a class

Minimizing the cost function by using gradient descent algorithm



https://www.3blue1brown.com/lessons/neural-networks

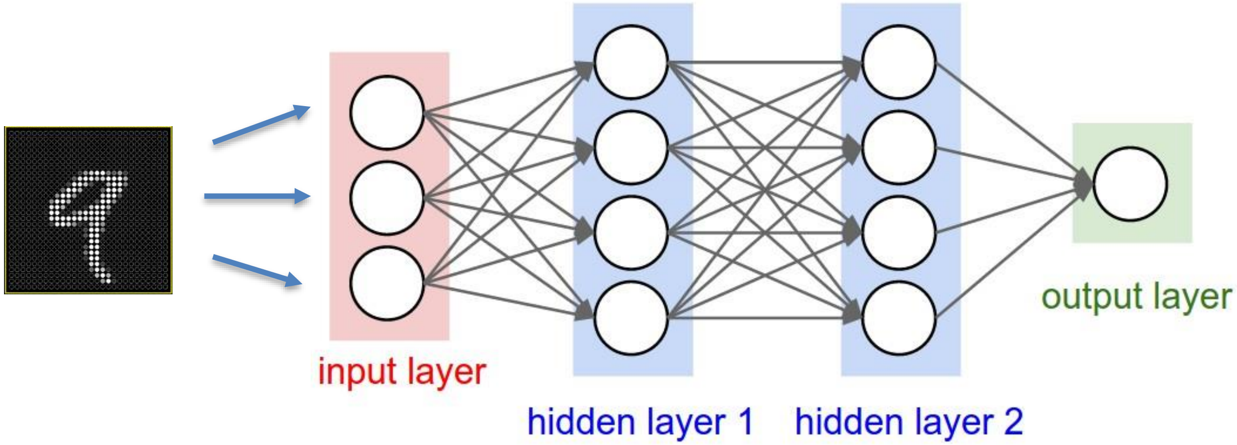
Backpropagation


Forwardpass
Backwardpass
x
y
$f(x,y)$
z
$\frac{dL}{dx} = \frac{dL}{dz}\frac{dz}{dx}$
df
$\frac{dL}{dz}$
$\frac{dL}{dy} = \frac{dL}{dz}\frac{dz}{dy}$

Q: X and Y are fed to the network and Z is the model predication - How good is this prediction or Z? How much error is associated with X and Y?
A: Calculate the derivative of the loss function with respect to X and Y (dL/dx and dL/dy) - Then subtract X and Y by dL/dx and dL/dy

https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199

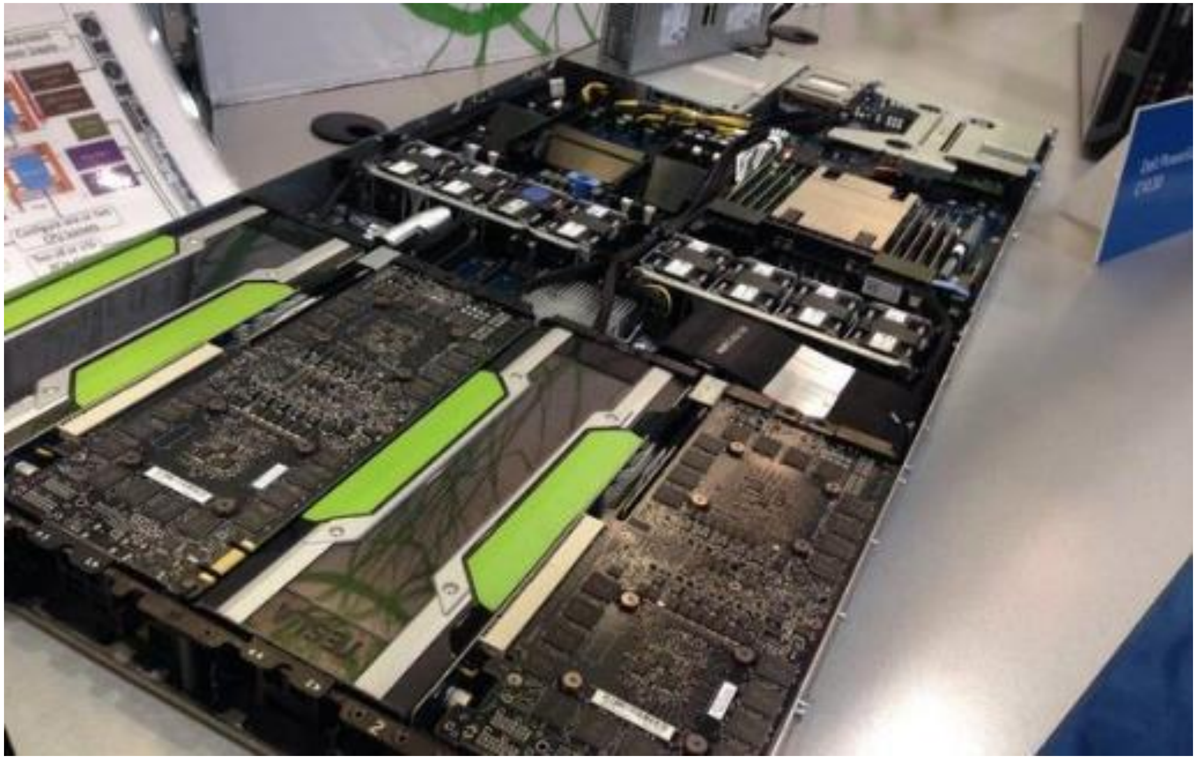
Overview of the training process



Training process review:

1. Forward pass input data through the network
2. Get the output from the network
3. Measure the error/loss between the network output and the true labels of data
4. Perform the backpropagation
5. Update network weights with the gradients

GPUs to the Rescue!



GPU cards are exceptionally well suited to Neural Network Mathematics

Orders of magnitude faster than CPU-based training

https://keras.io

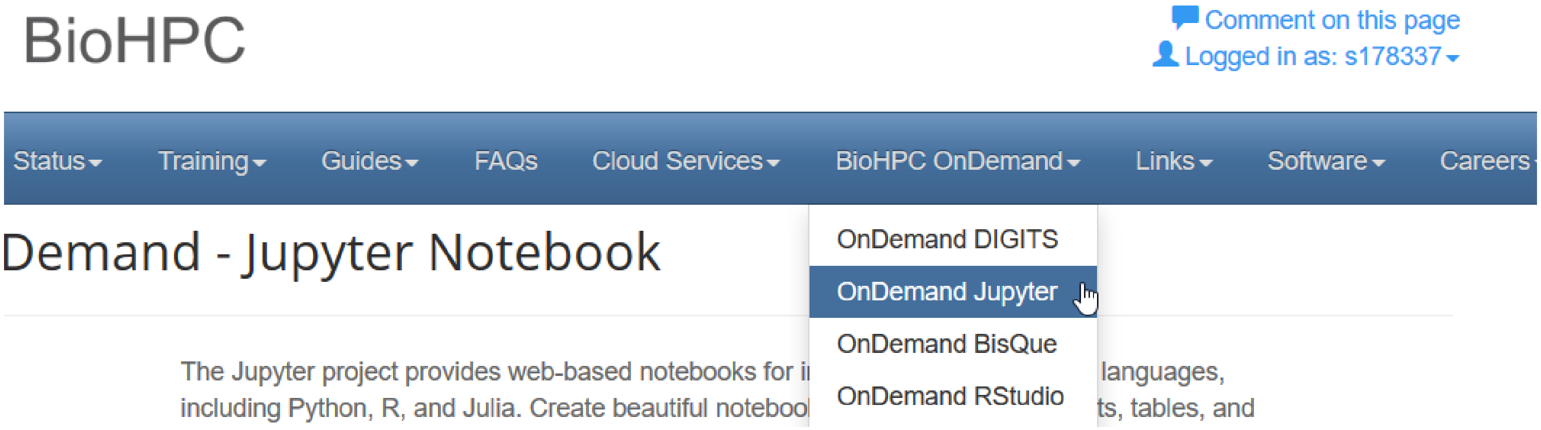
- High-level, open-source Python API
- *“Being able to go from idea to result with the least possible delay is key to doing good research”*
- Interface for TensorFlow, Microsoft Cognitive Toolkit, and Theano

Installing a Conda Environment for Keras and TensorFlow with Jupyter Support

```
$ module load python/3.6.4-anaconda

$ conda create --name py3.6-keras python=3.6 ipykernel keras
tensorflow-gpu pillow matplotlib


$ ipython kernel install --user --name py3.6-tfgpu --display-
name=“Keras (GPU)"
```



Demand - Jupyter Notebook

The Jupyter project provides web-based notebooks for i languages, including Python, R, and Julia. Create beautiful noteboo ts, tables, and

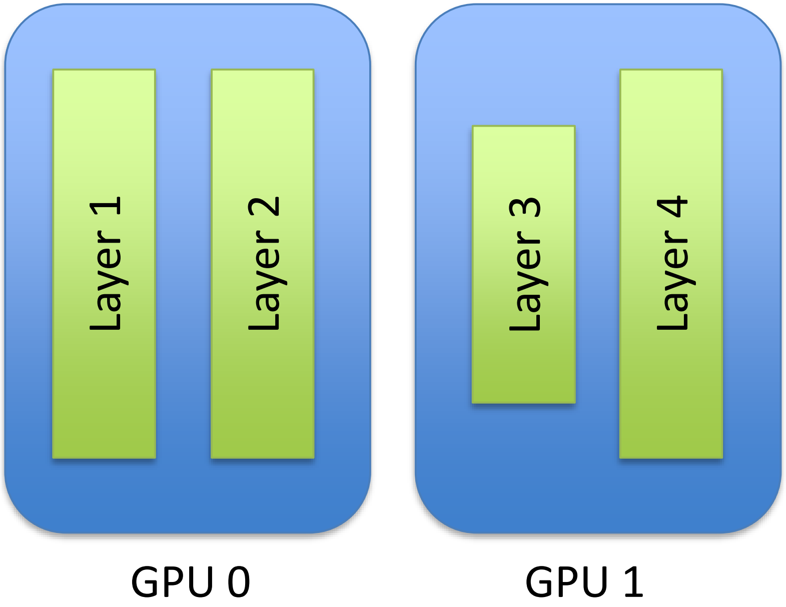
Basic concepts in distributed and parallel training Deep Learning: Performance metrics

- Speedup
- Throughput
- Scalability

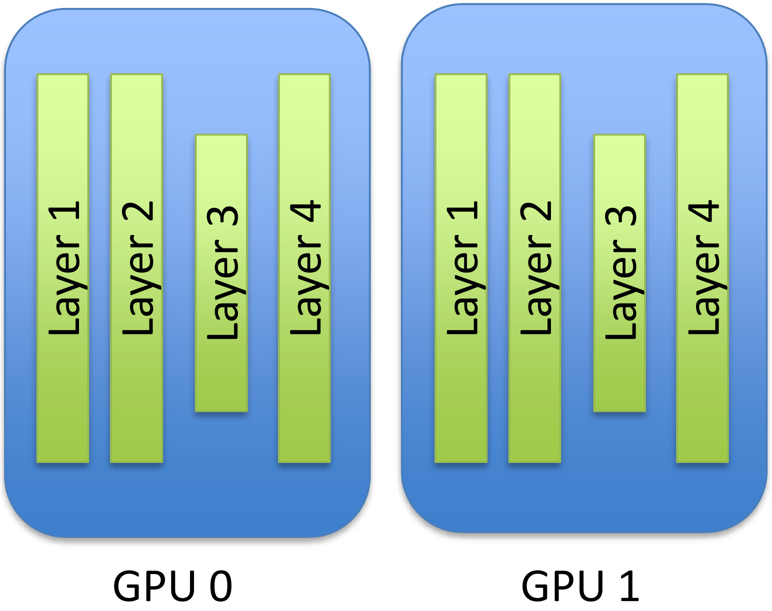
How to speedup the training process?

Basic concepts in distributed and parallel training Deep Learning: Types of parallelism

Model Parallelism

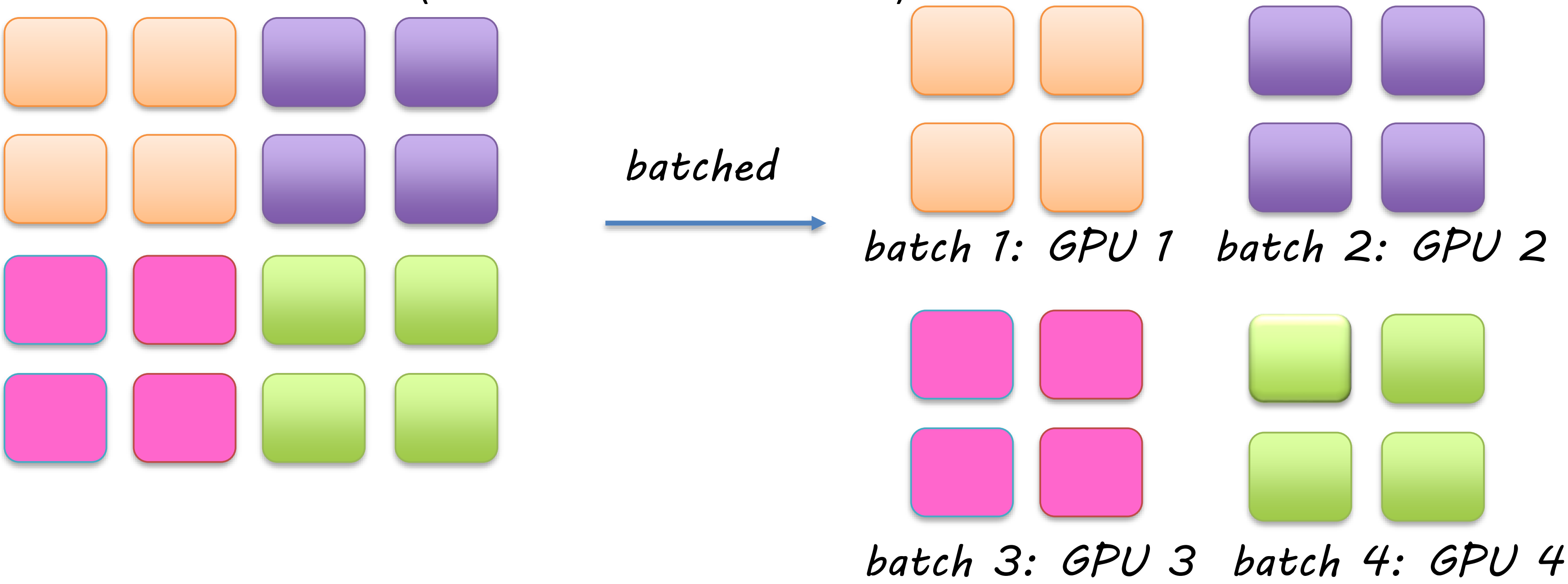


Data Parallelism



Basic concepts in distributed and parallel training Deep Learning: Batch size

The scale of data parallelism corresponds to the *batch size*.
(for the most common NNs)



Divide dataset into batches -> stochastic gradient descent -> takes longer to converge

```
Example: python my_dl_model.py --model resnet50 --batch-size 128
```

Basic concepts in distributed and parallel training Deep Learning: Relationship between batch size and training time

But what are the limits of the data parallelism approach, and when should we expect to see large speedups?

> *“while simple data parallelism can provide large speedups for some workloads at the limits of today‘s hardware (e.g. Cloud TPU Pods), and perhaps beyond, some workloads require moving beyond simple data parallelism in order to benefit from the largest scale hardware that exists today…”*

Maximum useful batch size depends on:

- *Model architecture*
- *The dataset*
- *The optimizer*

Read more: https://ai.googleblog.com/2019/03/measuring-limits-of-data-parallel.html

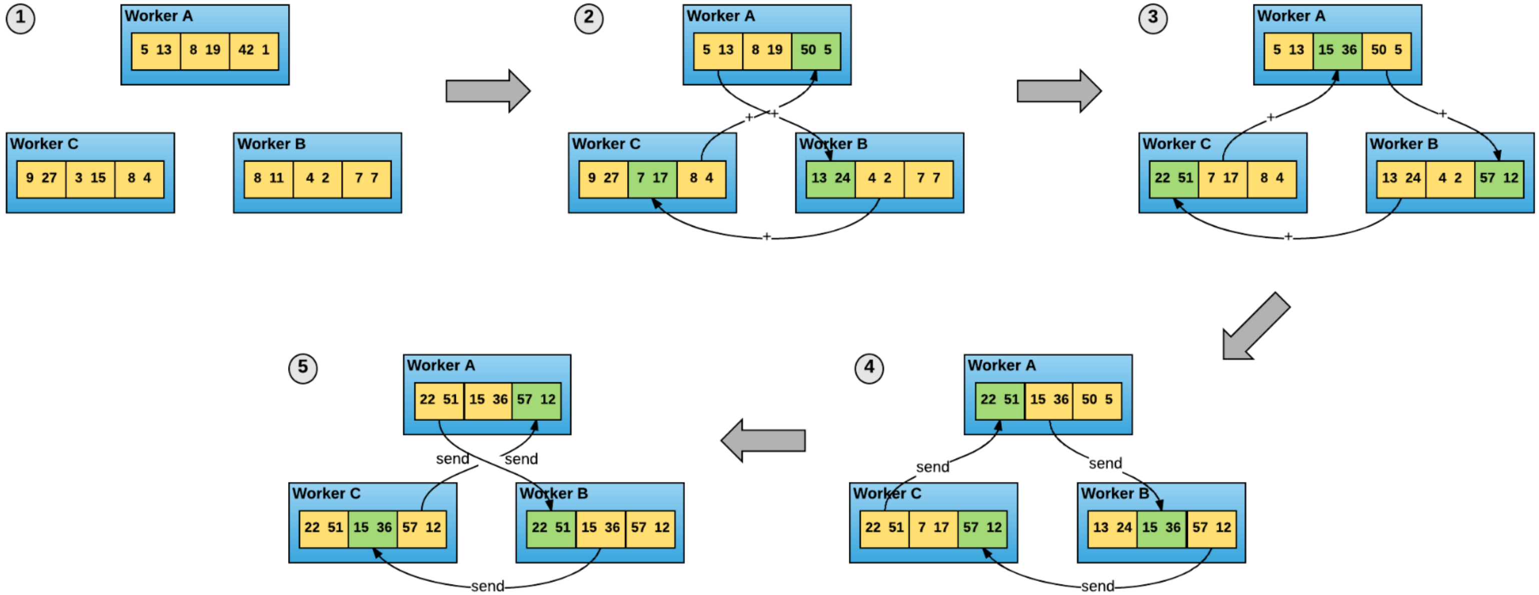
Distributed Deep Learning through Horovod

- Distributed DL training framework for:
  - Tensorflow/Keras
  - Pytorch
  - MXNet
- Open-source
- Separates infrastructure from DL/ML
- Installation: `pip install horovod` (not recommend)
  - Use existing Docker image and convert to Singularity
- Uses bandwidth-optimal communication protocols (e.g. Infiniband) if available
- Some terminology: Horovod master and worker nodes
  - Master sends the variables to the worker during the initialization
  - Worker nodes do the training job!

horovod.ai

```
Example: python my_dl_model_horovod.py --model resnet50 --batch-size 128
```

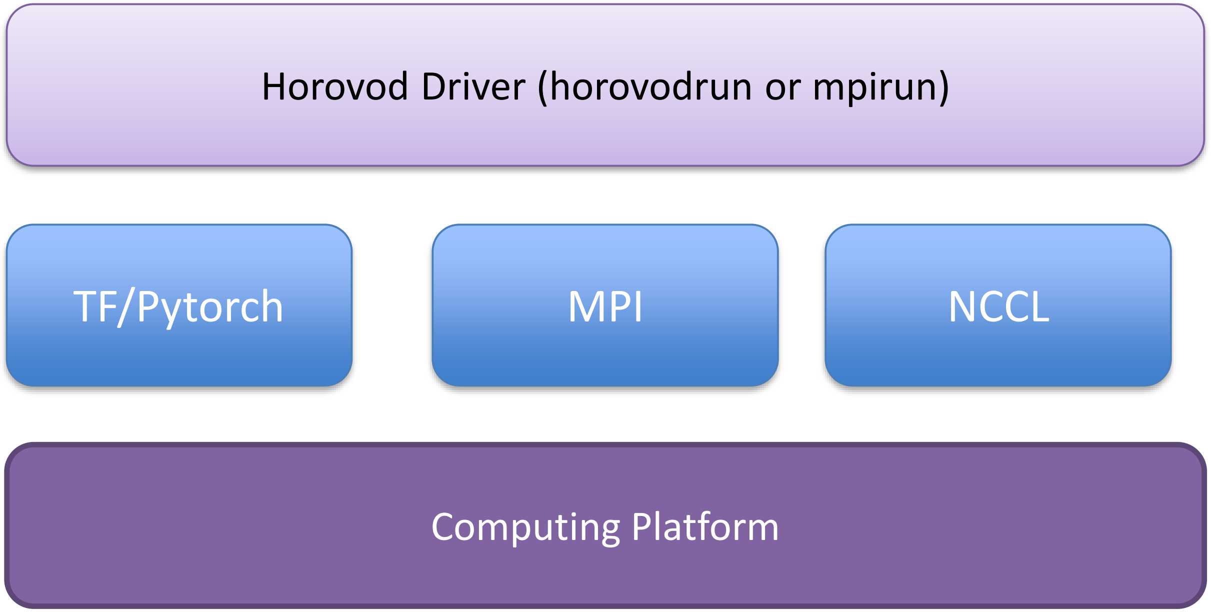
Distributed Deep Learning through Horovod (cont.)



Horovod Algorithm: Ring All-reduce

https://arxiv.org/pdf/1802.05799.pdf

Distributed Deep Learning through Horovod (contd.)



Horovod Stack

NCCL: Used for GPU-2-GPU communications
MPI: Used for CPU-2CPU comminications

Distributed Deep Learning through Horovod (cont.)

Simplified example of showing how to use Horovod with Tensorflow:

1. Initialize Horovod:

```
import horovod.tensorflow as hvd

hvd.init()
```

2. Pin GPU to be used to process local rank (one GPU per process):

```
config = tf.ConfigProto()

config.gpu_options.visible_device_list = str(hvd.local_rank())
```

3. Add Horovod Distributed Optimizer and scale the learning rate:

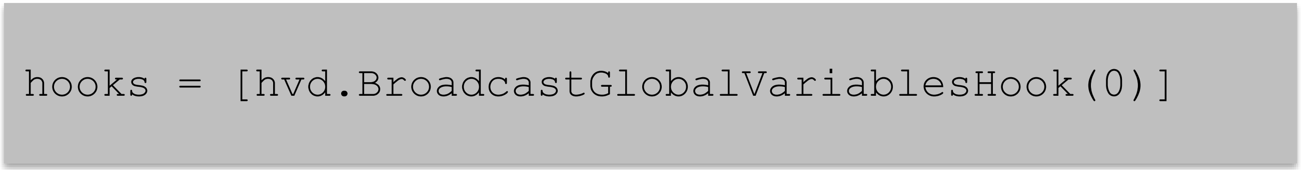
```
opt = tf.train.AdagradOptimizer(0.01 * hvd.size())

opt = hvd.DistributedOptimizer(opt)
```

https://horovod.ai/getting-started/

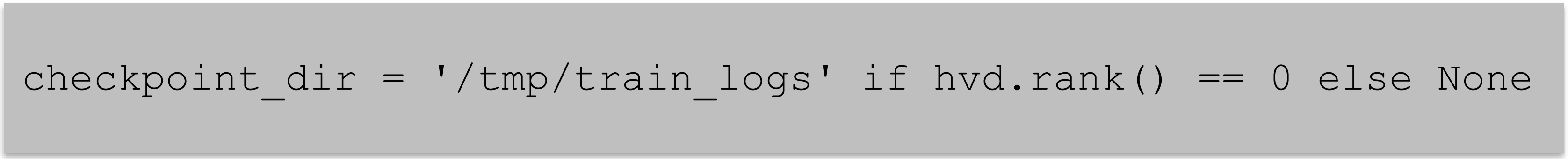
Distributed Deep Learning through Horovod (cont.)

4. Broadcast variables from rank 0 to all other processes during initialization.



```
hooks = [hvd.BroadcastGlobalVariablesHook(0)]
```

5. Save checkpoints only on worker 0 to prevent other workers from corrupting them.



```
checkpoint_dir = '/tmp/train_logs' if hvd.rank() == 0 else None
```

Read more: https://towardsdatascience.com/distributed-deep-learning-with-horovod-2d1eea004cb2

Horovod On BioHPC

1. Either install it via pip OR pull the Docker image of Horovod and convert to Singularity as following:

```
module load singularity/3.0.2
singularity pull docker://horovod/horovod:latest horovod_latest.sif
```

2. Get inside the Singularity container and check the mpi version:

```
singularity run horovod_latest.sif
mpirun --version
```

3. Based on the MPI version in the container, install the same version of openmpi in your account (instruction uploaded BioHPC Portal -> Training -> Training Slides & Hands out)

4. Submit the job to the cluster – Demo
(code is uploaded BioHPC Portal -> Training -> Training Slides & Hands out)

Try on GPUA100 or GPUV100s