Large Scale Training

argonne-lcf / ai-science-training-series

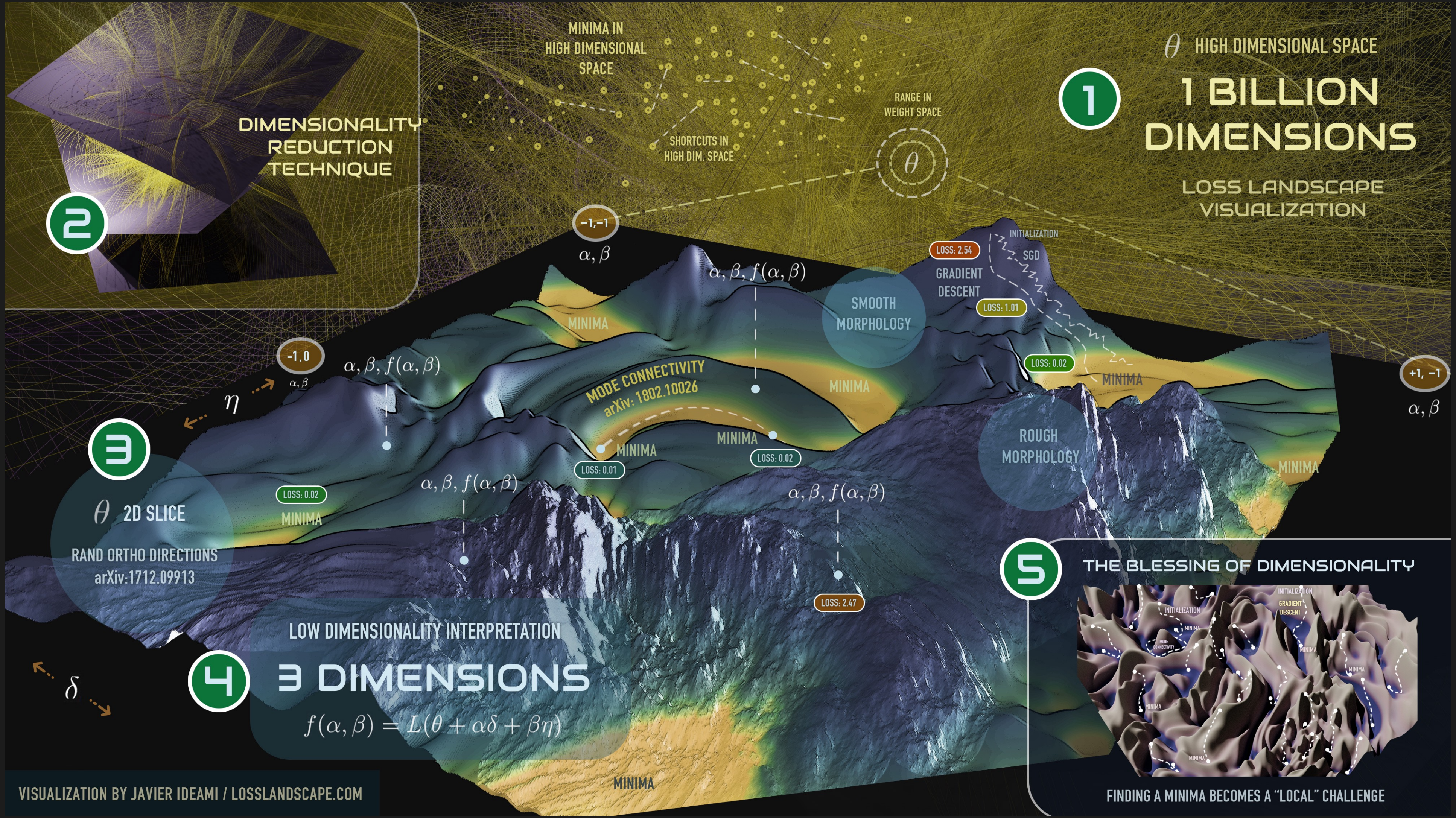
Sam Foreman

2022-11-01

Argonne NATIONAL LABORATORY

Why Distributed Training?

- Large batches may not fit in GPU memory
- Splitting data across workers $\longrightarrow$ larger batch size
- Smooth loss landscape
- Improved gradient estimators
- Less iterations needed for same number of epochs
    - May need to train for more epochs if another change is not made
    - e.g. scaling learning rate
- See Large Batch Training of Convolutional Networks



θ HIGH DIMENSIONAL SPACE
1
1 BILLION DIMENSIONS
LOSS LANDSCAPE VISUALIZATION
MINIMA IN HIGH DIMENSIONAL SPACE
SHORTCUTS IN HIGH DIM. SPACE
RANGE IN WEIGHT SPACE
θ
2
DIMENSIONALITY REDUCTION TECHNIQUE
-1,-1
α, β
α, β, f(α, β)
SMOOTH MORPHOLOGY
LOSS: 2.54
INITIALIZATION
SGD
GRADIENT DESCENT
LOSS: 1.01
LOSS: 0.02
MINIMA
+1, -1
α, β
-1, 0
α, β
η
δ
α, β, f(α, β)
MODE CONNECTIVITY
arXiv: 1802.10026
MINIMA
LOSS: 0.01
LOSS: 0.02
ROUGH MORPHOLOGY
3
θ 2D SLICE
RAND ORTHO DIRECTIONS
arXiv:1712.09913
LOSS: 0.02
α, β, f(α, β)
LOSS: 2.47
4
LOW DIMENSIONALITY INTERPRETATION
3 DIMENSIONS
f(α, β) = L(θ + αδ + βη)
5
THE BLESSING OF DIMENSIONALITY
FINDING A MINIMA BECOMES A “LOCAL” CHALLENGE
VISUALIZATION BY JAVIER IDEAMI / LOSSLANDSCAPE.COM

Recent Progress

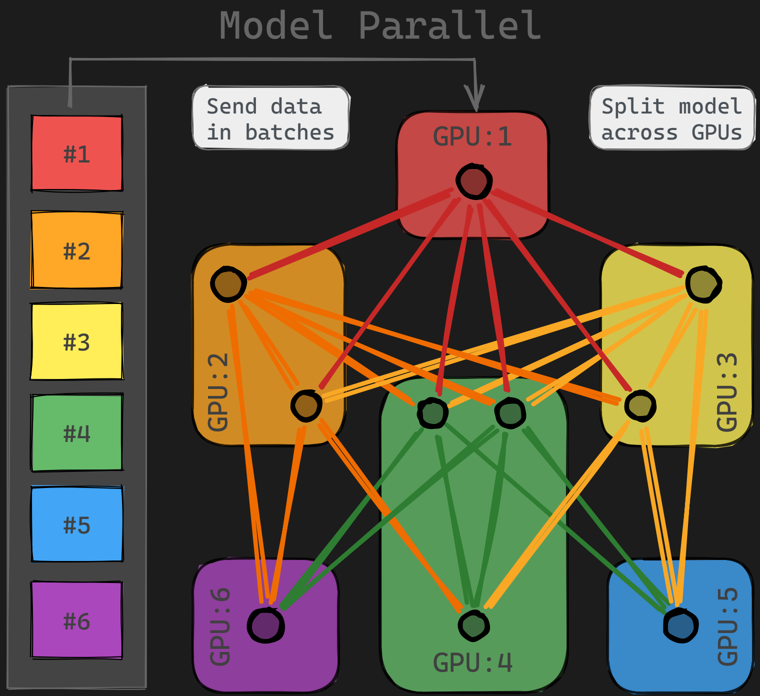
Year	Author	Batch Size	Processor	DL Library	Time	Accuracy
2016	He et al. [1]	256	Tesla P100 x8	Caffe	29 Hrs	75.3%
	Goyal et al. [2]	8192	Tesla P100	Caffe 2	1 hour	76.3%
	Smith et al. [3]	8192 → 16,384	full TPU pod	TensorFlow	30 mins	76.1%
	Akiba et al. [4]	32,768	Tesla P100 x1024	Chainer	15 mins	74.9%
	Jia et al. [5]	65,536	Tesla P40 x2048	TensorFLow	6.6 mins	75.8%
	Ying et al. [6]	65,536	TPU v3 x1024	TensorFlow	1.8 mins	75.2%
	Mikami et al. [7]	55,296	Tesla V100 x3456	NNL	2.0 mins	75.29%
2019	**Yamazaki et al**	**81,920**	**Tesla V100 x 2048**	**MXNet**	**1.2 mins**	**75.08%**

Argonne NATIONAL LABORATORY

Model Parallel Training

Model Parallel Training

- Split up network over multiple workers
    - Each receives disjoint subset
    - All communication associated with subsets are distributed
- Communication whenever dataflow between two subsets
- Typically **more complicated** to implement than data parallel training
- Suitable when the model is too large to fit onto a single device (CPU / GPU)

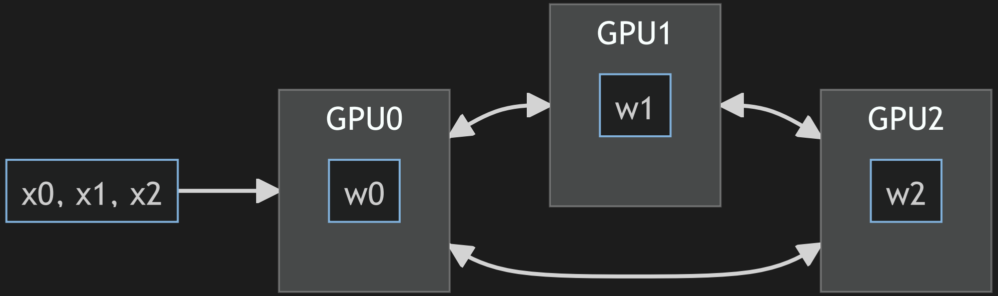


Argonne NATIONAL LABORATORY

Model Parallel Training

$$y = w_0 * x_0 + w_1 * x_1 + w_2 * x_2$$

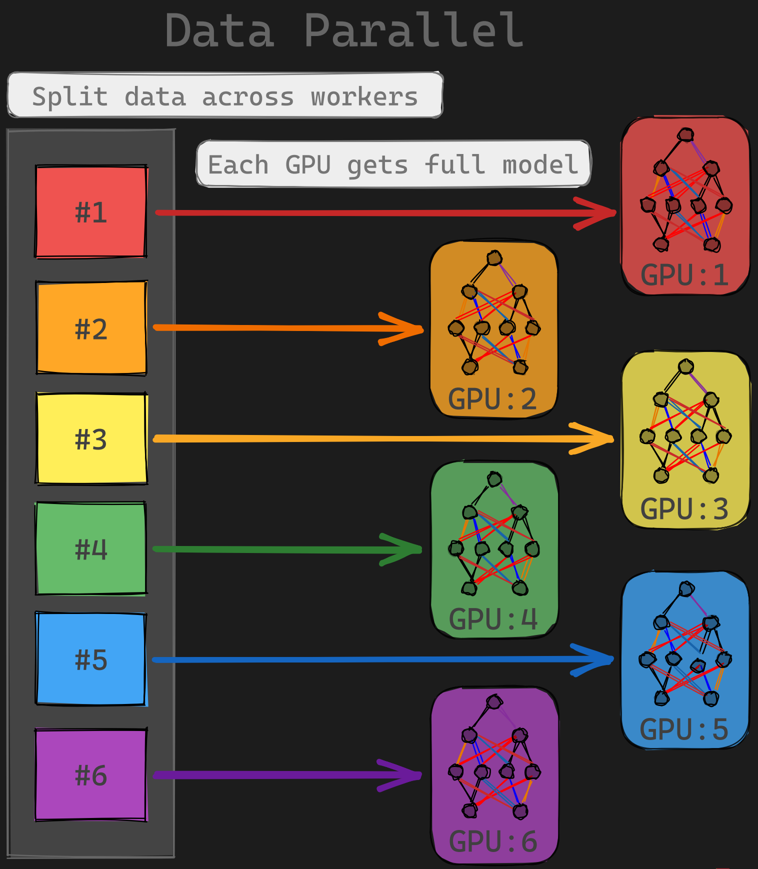
1. Compute $y_0 = w_0 * x_0$ and send to $\longrightarrow$ GPU1
2. Compute $y_1 = y_0 + w_1 * x_1$ and send to $\longrightarrow$ GPU2
3. Compute $y = y_1 * w_2 * x_2$ ✅



Data Parallel Training

Data Parallel Training

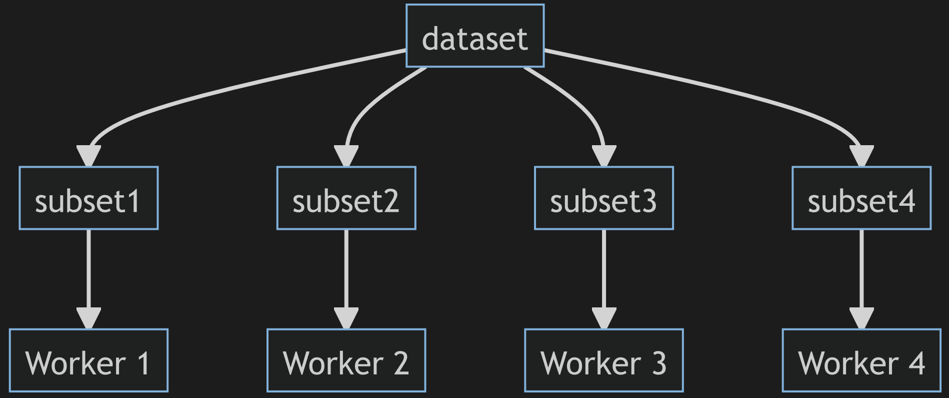
- Each worker has identical copy of complete model
- Each Worker computes the corresponding loss and gradients w.r.t **local** data
- Before updating parameters, loss and gradients averaged across workers
- Typically easier / simpler to implement



Argonne NATIONAL LABORATORY

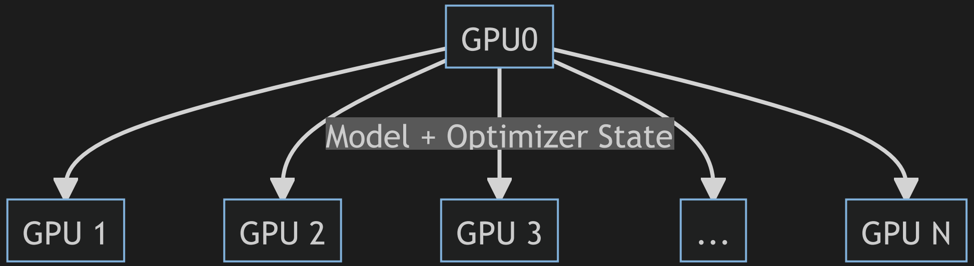
Data Parallel Training

- Each worker has identical copy of model
- **Global batch of data split across workers**
- Loss + Grads averaged across workers before updating parameters



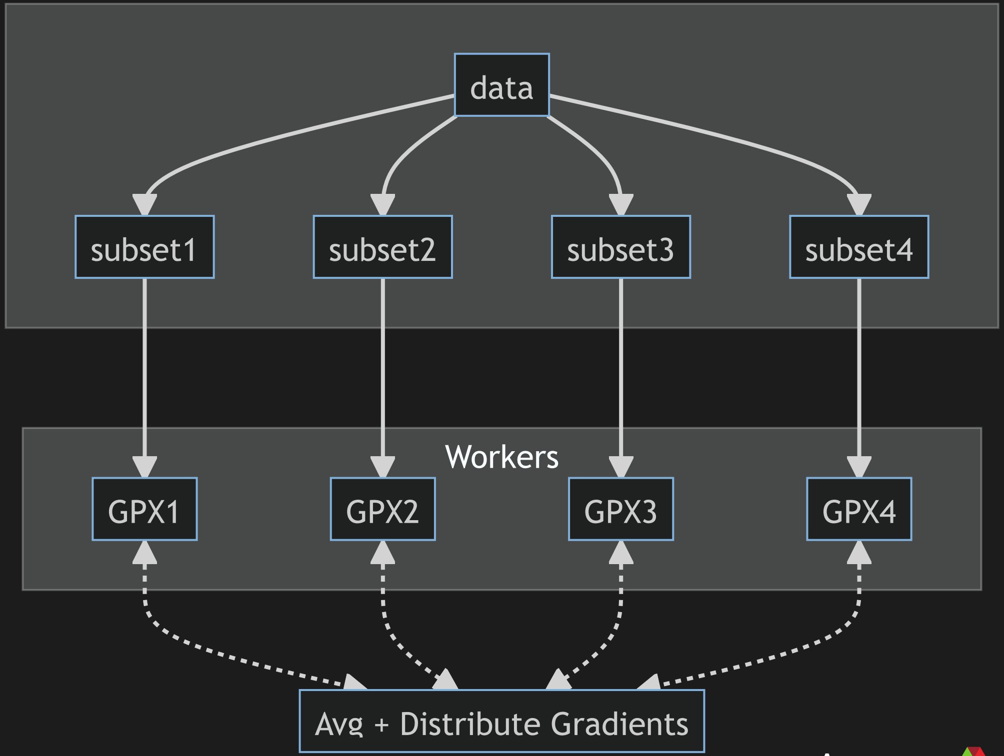
Broadcast Initial State

- At the start of training (or when loading from a checkpoint), we want all of our workers to be initialized consistently
  - **Broadcast** the model and optimizer states from `hvd.rank() == 0` worker

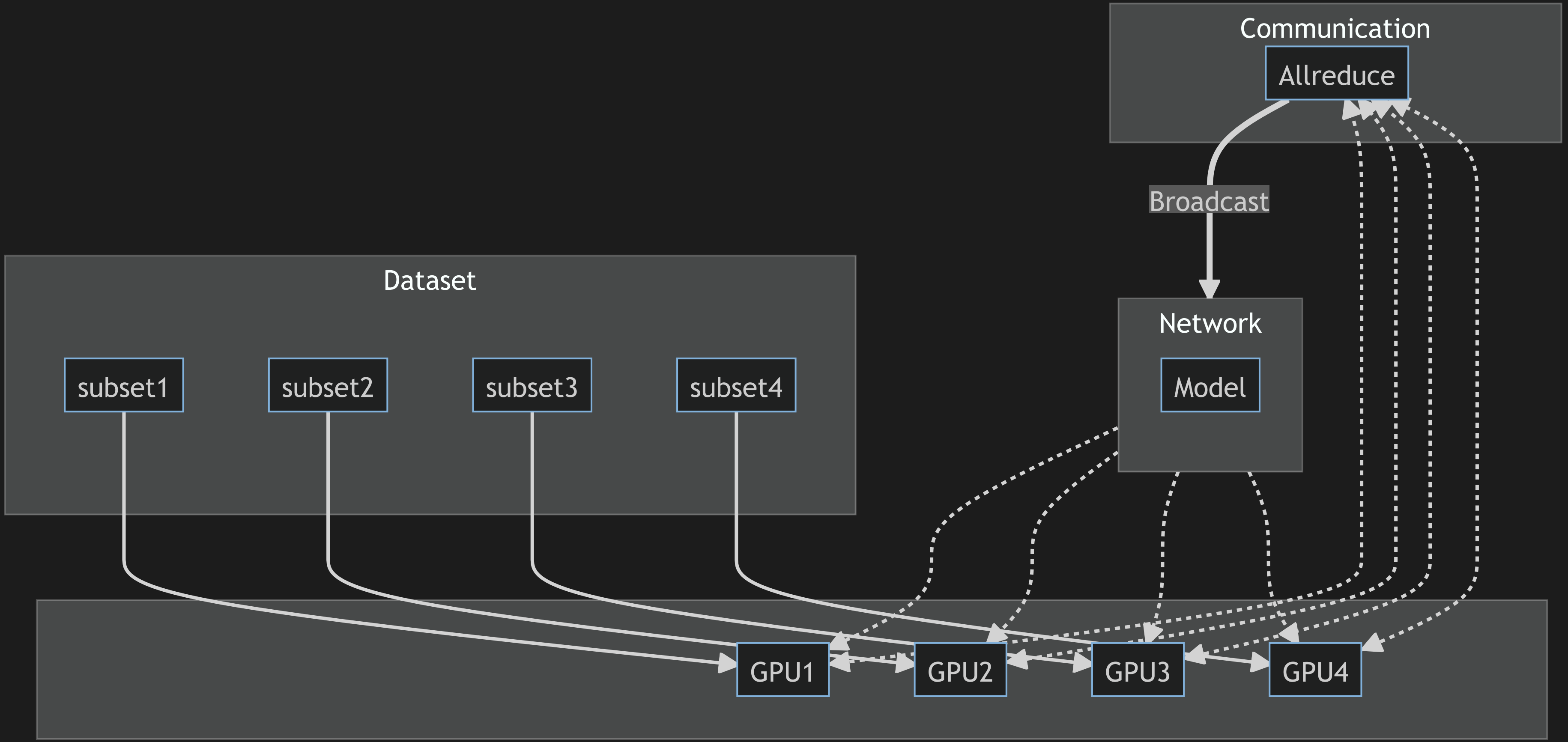


Data Parallel Training

- Disjoint subsets of a neural network are assigned to different devices
- Each worker receives:
    - **identical copy of model**
    - **unique subset of data**



Argonne NATIONAL LABORATORY



Argonne NATIONAL LABORATORY

TensorFlow + Horovod

- Set one GPU per process ID (`hvd.local_rank()`)

```
gpus = tf.config.experimental.list_physical_devices('GPU')
for gpu in gpus:
    tf.config.experimental.set_memory_growth(gpu, True)
if gpus:
    local_rank = hvd.local_rank()
    tf.config.experimental.set_visible_devices(gpus[local_rank], 'GPU')
```

Scale the Learning Rate

- Scale the learning rate by the number of workers to account for the increased batch size

```
import horovod.tensorflow as hvd
optimizer = tf.optimizers.Adam(lr_init * hvd.size())
```

Thank you!

- Organizers
- ALCF Data Science & Operations
- Feel free to reach out!

Acknowledgements

This research used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357.