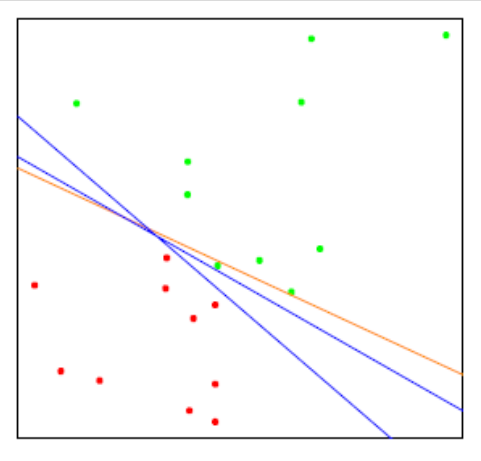


Parallel & Scalable Machine Learning

Introduction to Machine Learning Algorithms

Dr. –Ing. Gabriele Cavallaro

Postdoctoral Researcher

High Productivity Data Processing Group

Juelich Supercomputing Centre

Lecture 8 - 18/02/2020

PARALLEL AND DISTRIBUTED TRAINING OF ANN

JÜLICH Forschungszentrum | JÜLICH SUPERCOMPUTING CENTRE

HELMHOLTZ AI | ARTIFICIAL INTELLIGENCE COOPERATION UNIT

PRACE PARTNERSHIP FOR ADVANCED COMPUTING IN EUROPE TRAINING PORTAL

UNIVERSITY OF ICELAND
SCHOOL OF ENGINEERING AND NATURAL SCIENCES
FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

DEEP Projects

COURSE OUTLINE

- Parallel and Scalable Machine Learning Driven by HPC
- Introduction to Machine Learning Fundamentals
- Supervised Learning with a Simple Learning Model
- Artificial Neural Networks (ANNs)
- Introduction to Statistical Learning Theory
- Validation and Regularization
- Pattern Recognition Systems
- Parallel and Distributed Training of ANN
- Supervised Learning with Deep Learning
- Unsupervised Learning – Clustering
- Clustering with HPC
- Introduction to Deep Reinforcement Learning

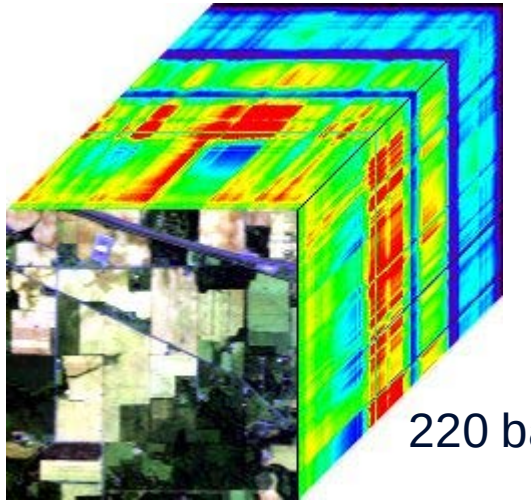
OUTLINE

- Classification of small hyperspectral images
- Distribute training with Horovod

PRACTICAL

Classification of Hyperspectral Images

- Indian Pine Dataset (small site)
  - It contains 2/3 agriculture, and 1/3 forest or other natural perennial vegetation
  - There are two major dual lane highways, a rail line, as well as some low density housing, other built structures, and smaller roads
  - Since the scene is taken in June some of the crops present, corn, soybeans, are in early stages of growth with less than 5% coverage



[1] Rasti Behnood, et al.

Groundtruth classes for the Indian Pines scene and their respective samples number

#	Class	Samples
1	Alfalfa	46
2	Corn-notill	1428
3	Corn-mintill	830
4	Corn	237
5	Grass-pasture	483
6	Grass-trees	730
7	Grass-pasture-mowed	28
8	Hay-windrowed	478
9	Oats	20
10	Soybean-notill	972
11	Soybean-mintill	2455
12	Soybean-clean	593
13	Wheat	205
14	Woods	1265
15	Buildings-Grass-Trees-Drives	386
16	Stone-Steel-Towers	93

[2] Hyperspectral Images

COMPETITION

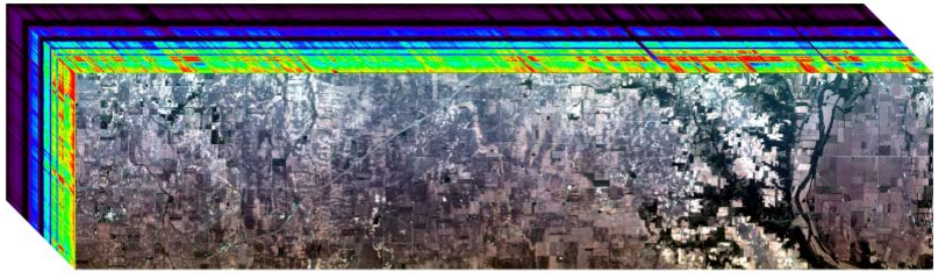
- **Use the same train size = 0.1**

Group Name	Test Accuracy

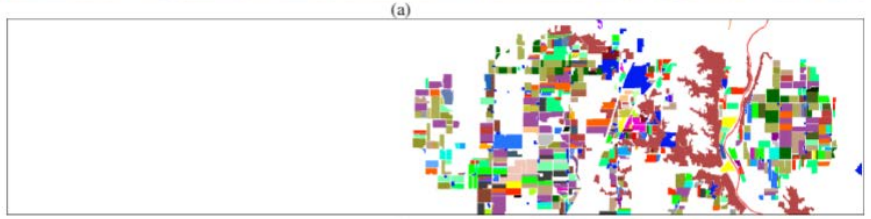
Group Name	Test Accuracy

HOMEWORK

- Indian Pine Dataset (large site)



(a)



(b)

Thematic classes:

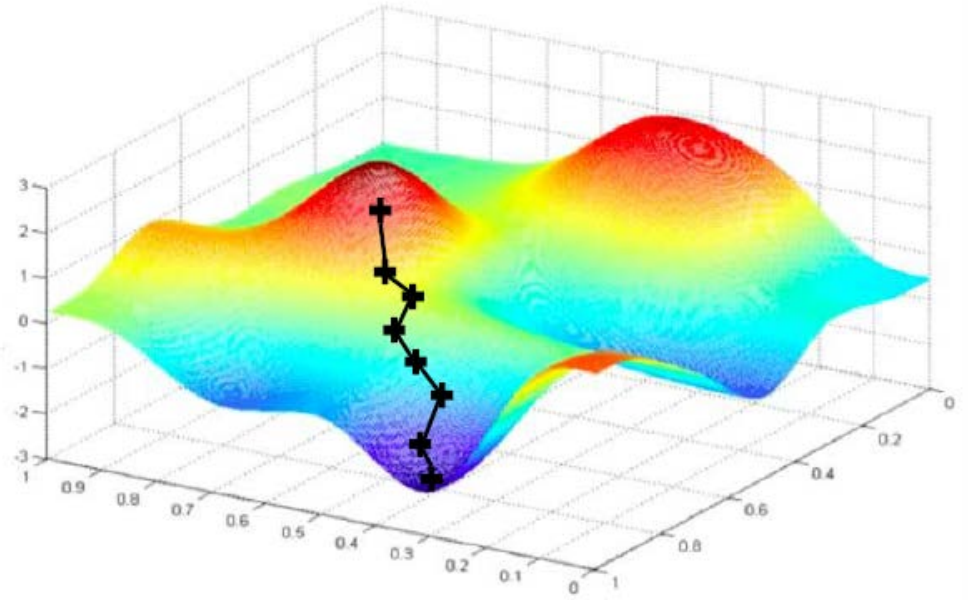
BareSoil (57)	Corn-MinTill (1049)	Hay (1128)	Soybeans-NS (1110)	Soybeans-NaTill (2157)
Buildings (17195)	Corn-MinTill-EW (5629)	Hay (pre.) (2185)	Soybeans-CleanTill (5074)	Soybeans-NaTill-EW (2533)
Concrete/Asphalt (69)	Corn-MinTill-NS (8862)	Hay-Alfalfa (2258)	Soybeans-CleanTill (pre.) (2726)	Soybeans-NaTill-NS (929)
Corn (17783)	Corn-Natill (4381)	Lake (224)	Soybeans-CleanTill-EW (11802)	Soybeans-NaTill-Drilled (8731)
Corn (presumed) (158)	Corn-Natill-EW (1206)	NotCropped (1940)	Soybeans-CleanTill-NS (10387)	Swampy Area (583)
Corn-EW (514)	Corn-Natill-NS (5685)	Oats (1742)	Soybeans-CleanTill-Drilled (2242)	River (3110)
Corn-NS (2356)	Fescue (114)	Oats (pre.) (335)	Soybeans-CleanTill-Weedy (543)	Trees (pre.) (580)
Corn-CleanTill (12404)	Grass (1147)	Orchard (39)	Soybeans-MinTill (15118)	Wheat (4979)
Corn-CleanTill-EW (26486)	Grass/Trees (2331)	Pasture (10386)	Soybeans-Drilled (2667)	Woods (63562)
Corn-CleanTill-NS (39678)	Grass/Pasture-mowed (19)	Pond (102)	Soybeans-MinTill (1832)	Unknown (144)
Corn-CleanTill-NS-Irrigated (800)	Grass/Pasture (73)	Soybeans (9391)	Soybeans-MinTill-Drilled (8098)	
Corn-CleanTill-NS (pre.) (1728)	Grass-runway (37)	Soybeans (pre.) (894)	Soybeans-MinTill-NS (4953)	

MINI-BATCH GRADIENT DESCENT

Batching the data into mini-batches

- Tradeoff between **BGD** and **SGD**
- Gives an estimate of the true gradient by averaging the gradient from each of the B points
- Minibatch sampling: implemented by shuffling the dataset S, and processing that permutation by obtaining contiguous segments of size B from it

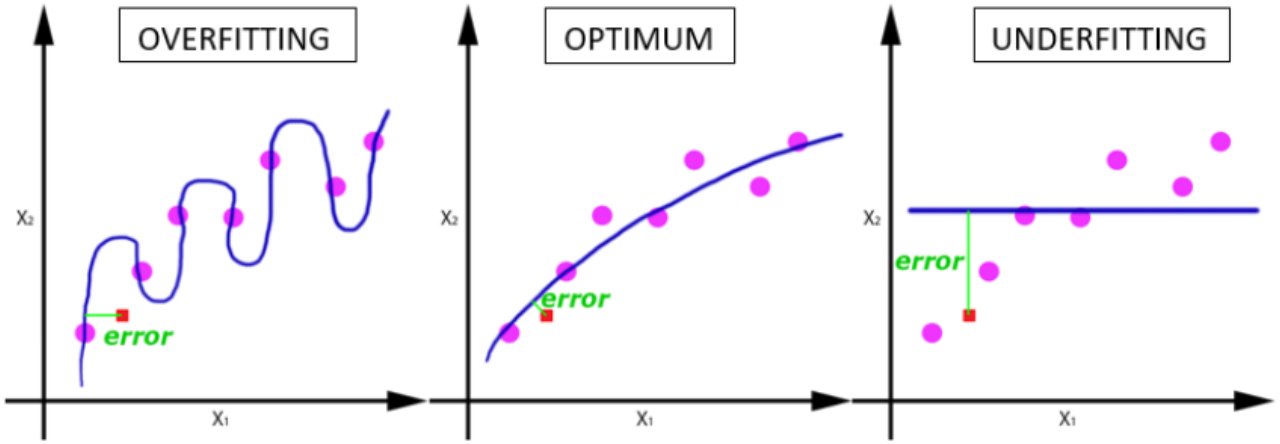
1. Initialize weights randomly $\sim \mathcal{N}(0, \sigma^2)$
2. Loop until convergence
3. **Pick batch of B data points**
4. Compute gradient $\frac{\partial \mathcal{L}_i(\boldsymbol{W})}{\partial \boldsymbol{W}} = \frac{1}{B} \sum_{k=1}^{B} \frac{\partial \mathcal{L}_i(\boldsymbol{W})}{\partial \boldsymbol{W}}$
5. Update weights $\boldsymbol{W} \coloneqq \boldsymbol{W} - \eta \frac{\partial \mathcal{L}(\boldsymbol{W})}{\partial \boldsymbol{W}}$
6. Return weights

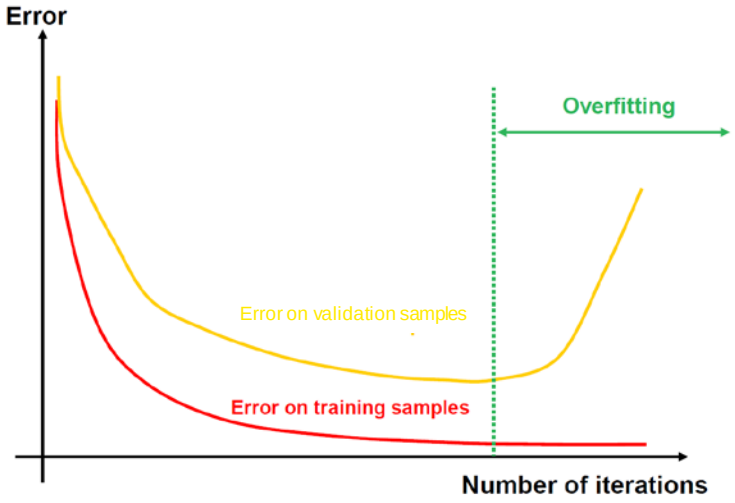


Fast to compute and a much better estimate of the true gradient!

EPOCHS AND ITERATIONS

- **1 Epoch**: **entire training** set passed forward and backward through the network in once
  - The training set is divided in batches since the data can be too large
- **1 iteration**: **entire batch** passed forward and backward through the network in once
  - If 1000 training samples and batch size set to 500, it means 2 iterations to complete 1 Epoch





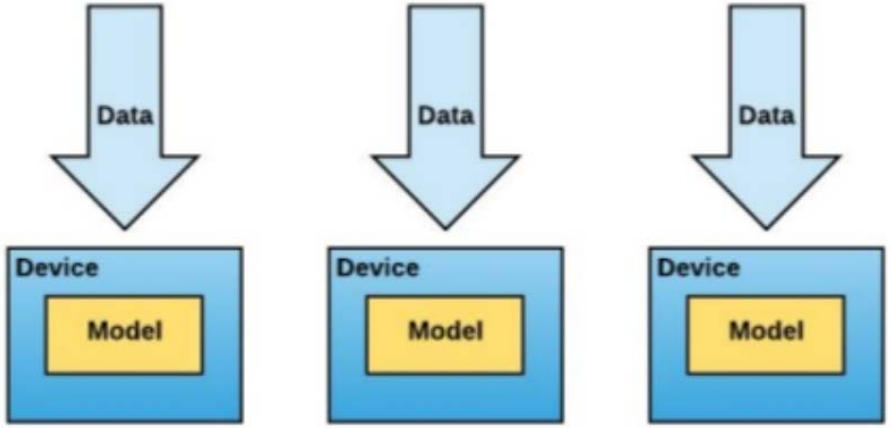
What is the right numbers of epochs?

DISTRIBUTED TRAINING

With Data Parallelism

- **Mini-Batch Gradient Descent:**
  - More accurate estimation of gradient and smoother convergence
  - Allows for larger learning rates (i.e., trust more the gradient , training faster)
  - Can **parallelize computation** and achieve significant speed increases

Send batches across the GPUs, compute their gradient simultaneously and aggregate them back

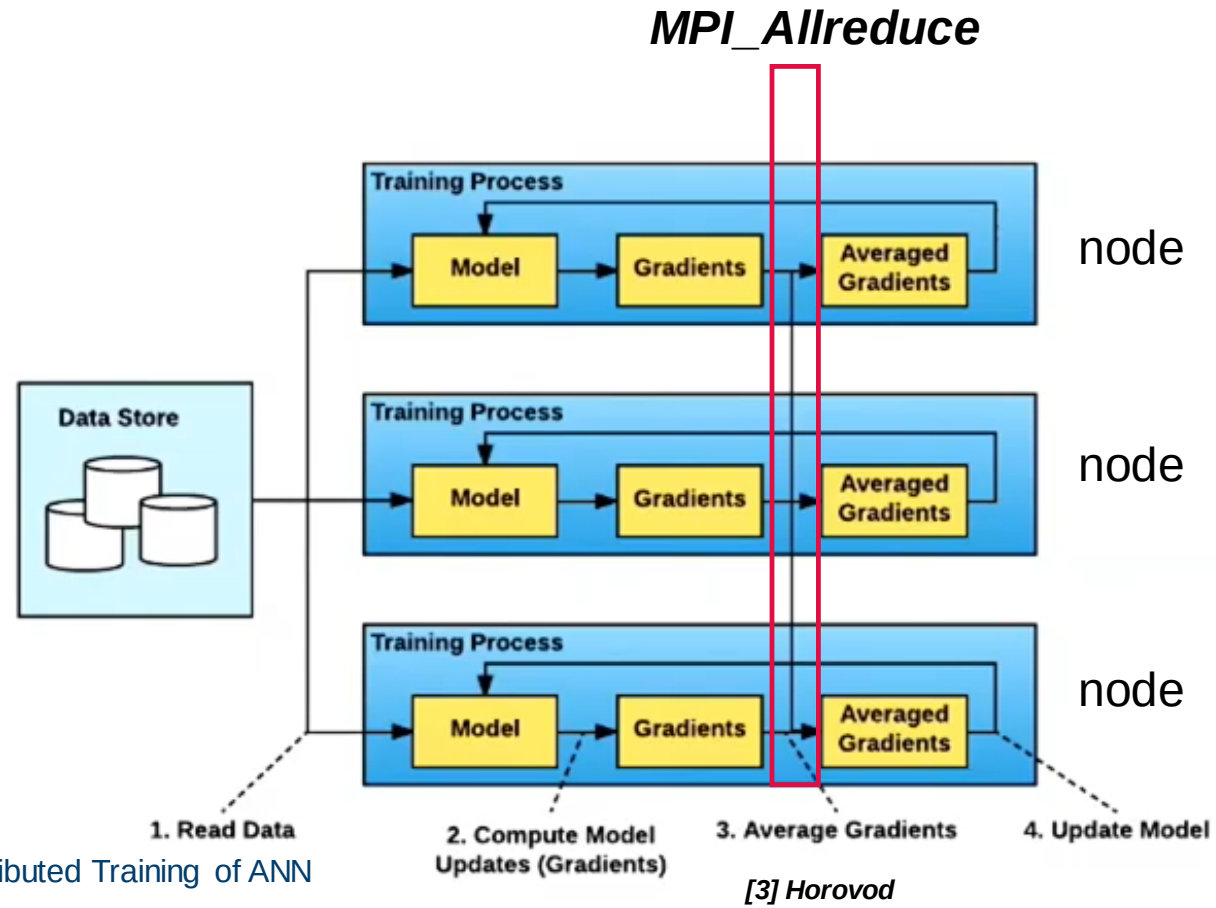


[3] Horovod

DISTRIBUTED TRAINING

With Data Parallelism

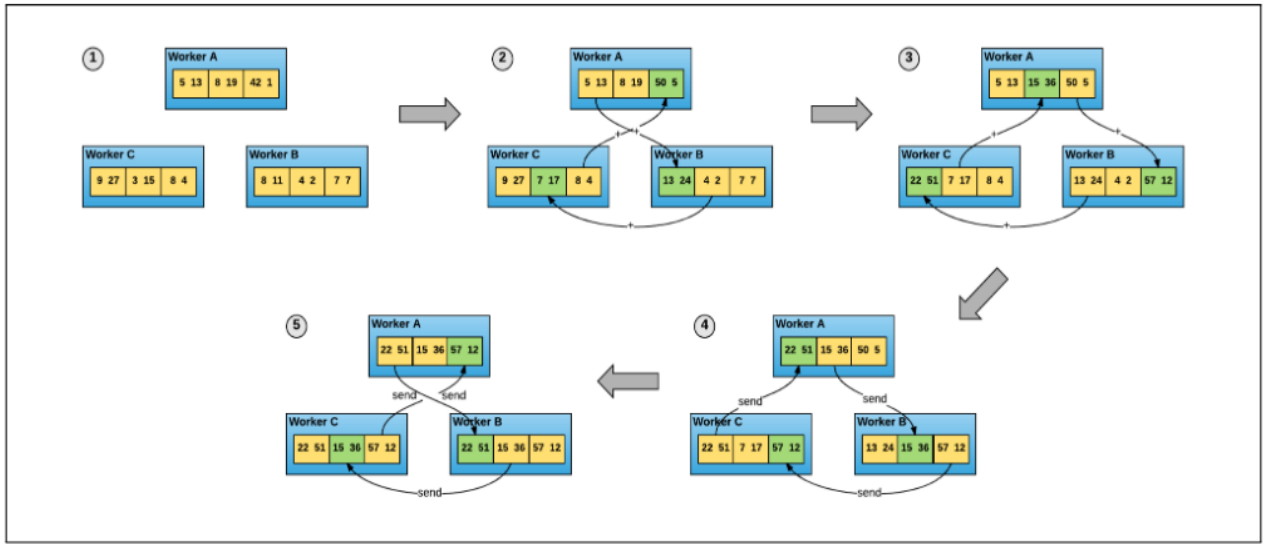
- The gradients for different batches of data are calculated separately on each node
- But averaged across nodes to apply consistent updates to the model copy in each node



[3] Horovod

HOROVOD

- Horovod is a data distribution framework
- Supports Keras, Tensorflow, PyTorch
- Based on MPI operationts and NCCL library
- No need for a parameter server
- Implemented using the Ring-AllReduce operation for the exchange of the gradient between the different actors
- Other frameworks for data parallelization exist (e.g. Distributed Tensorflow)



[4] Horovod: Ring-AllReduce

HOROVOD

How to use it

1. Initialize Horovod

hvd.init()

2. Pin GPU to each process (local rank)

config = tf.ConfigProto()

config.gpu_options.allow_growth = True

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

K.set_session(tf.Session(config=config))

[5] Fahad Khalid

HOROVOD

How to use it

3. Distribute the optimizer of your choice

opt = hvd.DistributedOptimizer(opt)

4. Broadcast initial variable states from rank 0 to all other processes to ensure consistent initialization of all workers when training is started with random weights or restored from a checkpoint

callbacks = [hvd.callbacks.BroadcastGlobalVariablesCallback(0),]

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

if hvd.rank() == 0:

callbacks.append(keras.callbacks.ModelCheckpoint('checkpoints/checkpoint-{epoch}.h5'))

- Remember to pass the callback as an argument to the fit function!

[5] Fahad Khalid

HOROVOD

How to use it

- When using Horovod, it is up to the developer to distribute the data among the workers (e.g. using sharding in TensorFlow)

- In this simple example, the same data are loaded on each GPU
  - However, more complex approaches are needed for a real training distribution

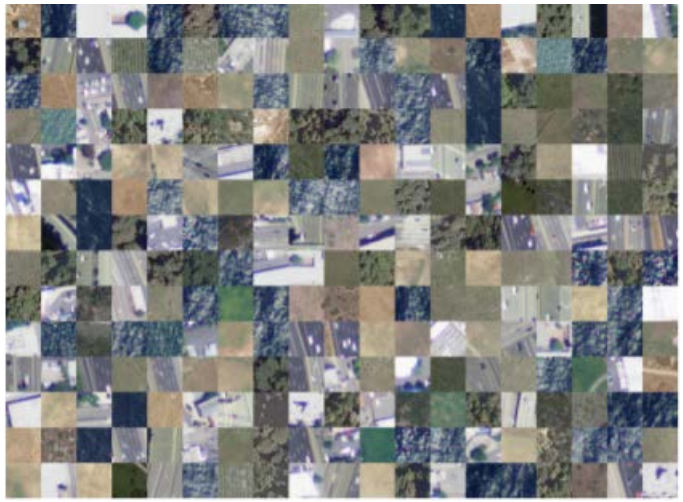
- Each GPU processes a different subset of training data

- Have a look at "Accurate, Large Minibatch SGD:Training ImageNet in 1 Hour" by Goyal et al., 2017 to get some insight (learning rate scheduler & other tips)

DEEPSAT-SAT6

Classification problem

- Images were extracted from the National Agriculture Imagery Program (NAIP) database
- 4 bands - red, green, blue and Near Infrared (NIR)
- 28x28 pixels
- The training and test labels are one-hot encoded 1x6 vectors (6 agricultural land cover classes)
- 324,000 training images
- 81,000 test images



[6] DEEPSAT-SAT6

REFERENCES

- [1] Rasti, Behnood, “Sparse Hyperspectral Image Modeling and Restoration”, 10.13140/RG.2.1.4097.4247, 2014.

  Online: https://www.researchgate.net/publication/277004219_Sparse_Hyperspectral_Image_Modeling_and_Restoration
- [2] Hyperspectral Images

  Online: https://engineering.purdue.edu/~biehl/MultiSpec/hyperspectral.html
- [3] Horovod: Uber’s Open Source Distributed Deep Learning Framework for TensorFlow

  Online: https://www.slideshare.net/databricks/horovod-ubers-open-source-distributed-deep-learning-framework-for-tensorflow
- [4] Horovod: Ring-AllReduce

  Online: https://towardsdatascience.com/distributed-tensorflow-using-horovod-6d572f8790c4
- [5] Tutorial on distributed ML by Fahad Khalid, FZ Juelich:

  Online: https://gitlab.version.fz-juelich.de/hpc4ns/dl_on_supercomputers
- [6] DEEPSAT-SAT6 dataset and paper

  Online: https://www.kaggle.com/crawford/deepsat-sat6 and https://arxiv.org/pdf/1509.03602.pdf