Week	Lectur	Title	Contents

1	1	Introduction to the course	- Introduction to the course, objectives, and expectations.
			- Describe "Changing hats" sessions (and lottery)
			- Describe "Hands-on" sessions
			- Describe "Invited lecture"
			- Describe Program
			- Introduction to HPC
			  - Overview of high-performance computing, its evolution, and its impact on various domains.
			  - Importance of HPC in scientific research, simulations, and data-intensive applications
			- Introduction to Distributed AI

	2	Introduction to HPC	- Brief history of supercomputers - Measuring capacity: FLOPs and memory - Comparing resources: from your laptop to the World's TOP10 from TOP500 - Requirements (cooling, space, etc.) and "the first" big issue: scheduling
	3	Fundamentals of HPC Architecture	- Lookback into 102.2 - Basics of parallel processing and its role in achieving high performance. - Overview of supercomputers, clusters, and distributed memory systems. - Introduction to GPU acceleration and its applications. - Mention TPU briefly - Brief intro to DISCO (w/focus on hardware)

	4	Scheduling and environment: from PC to HPC	- Lookback into unit 202.2 (OS and concurrence)
			- OS scheduling vs. HPC scheduling
			- HPC scheduling software & SLURM (Simple Linux Utility for Resource Management):
			  - Define the role of job scheduling in optimizing resource utilization
			  - Highlight the challenges of managing computing resources in a multi-tenant environment.
			  - Introduce SLURM as a solution for efficient job scheduling.
			- Preparing environments on HPC & containers

2	5	SLURM	- Architecture of SLURM
			  - Explain the purpose and functionality of SLURM's control daemon (slurmctld)
			  - Describe the role of SLURM's job scheduler (slurmd) on compute nodes.
			  - Outline the interaction between SLURM components in managing job submissions and resource allocation
			  - Introduce the concept of partitions
			- Deployment of SLURM
			  - Discuss the SLURM configuration file (slurm.conf) and its key parameters.
			  - Demonstrate how to define partitions and resources within the configuration
			  - Discuss some optimization options depending on different cluster architecture use-cases
			- Job submission and management:
			  - Submitting jobs with sbatch and specifying job requirements
			  - Introduce a typically underestimated utility: job arrays
			  - Explain how to monitor job status, resource usage, and job dependencies using various SLURM commands (squeue, scontrol)
			  - Interative sessions on nodes (srun/sdev)
			- Advanced SLURM & best practices
			  - Strategies to parallelize across nodes (multi-node job execution)
			  - Extend job arrays in parallelization
			  - Mention the existence of plugins and showcase some scripting extensions (e.g., launcher)
			  - Optimizing job performance and minimizing queue latencies
			  - Common issues and errors (e.g., execution on login node) in SLURM usage and "good citizenship" (e.g., "nice" config)
			- Provide a perspective on SLURM alternatives (Torque, PBS Pro, HTCondor, SGE)

	6	Hands-on: playing with DISCO	- A practical session where prepared (and small) programs will be launched by the students, to then track the execution by the cluster. - SLURM tools for post-investigation of the fate of programs. Some of the examples will intendendly be created to fail (e.g., allocating excessive memory), letting the students to investigate the problem. - Etiquette when using multi-tenant systems such as HPCs - login nodes, data-transfer nodes and compute nodes.

	7	HPC vs Cloud	- Define the fundamental concepts of HPC and Cloud Computing, highlighting differential points.
			- Highlight the primary differences in architecture, resource provisioning, and usage models.
			- Infrastructure Ownership
			  - Discuss how HPC involves dedicated, on-premises infrastructure owned by an organization.
			  - Contrast this with Cloud Computing, where infrastructure is typically owned and managed by a third-party cloud service provider.
			- Resource Provisioning
			  - Explain the on-demand nature of resource provisioning in Cloud Computing, allowing for flexibility and scalability.
			  - Describe the static allocation of resources in HPC, where users typically have dedicated access to specific hardware.
			- Security
			  - Discuss security considerations in HPC, where isolated, tightly controlled environments are preferred for sensitive research and data.
			  - Highlight traditional HPC practices that prioritize physical and network security within controlled environments.
			  - Address security considerations in Cloud Computing, emphasizing shared infrastructure and virtualization challenges.
			  - Discuss the role of cloud providers in ensuring security and the importance of robust access controls.
			- Describe scenarios where each option is most efficient / useful
			  - Identify scenarios where HPC is most efficient, such as complex scientific simulations, numerical modeling, and large-scale data processing.
			  - Discuss the advantages of specialized hardware for compute-intensive tasks.
			  - Highlight scenarios where Cloud Computing excels, including variable workloads, rapid scalability, and global accessibility.
			  - Discuss the advantages of outsourcing infrastructure for cost savings and flexibility.
			- Highlighting the cost, performance, and scalability implications of choosing between HPC and cloud solutions.
			  - Break down the cost implications of HPC, including initial infrastructure investment and ongoing maintenance.
			  - Contrast this with the pay-as-you-go model of Cloud Computing, where costs are tied to actual resource usage.
			  - Discuss the performance benefits of HPC, where specialized hardware can deliver high throughput and low latency.
			  - Acknowledge the performance trade-offs in Cloud Computing, where shared resources may lead to variability in performance.
			  - Explore scalability considerations in both HPC and Cloud Computing.
			  - Discuss how HPC scales vertically with powerful hardware, while Cloud Computing scales horizontally by adding more virtual instances.
			- Deploying environments: Cloud vs. HPC
			  - Detail the process of deploying environments in HPC, emphasizing custom configurations and direct hardware access.
			  - Discuss how HPC environments are tailored for specific applications and workloads.
			  - Outline the ease of environment deployment in Cloud Computing using virtual machines or containers.
			  - Introduce containerization technologies like Docker and Kubernetes that facilitate consistent environment deployment across clouds.
			  - Discuss the security challenges associated with containerization, particularly in multi-tenant environments.
			  - Address concerns related to container isolation, shared kernel vulnerabilities, and container escape.
			  - Introduce security-focused containerization solutions in HPC, such as Singularity (now known as Apptainer).
			  - Discuss why HPC traditionally resisted Docker due to security concerns and how Singularity addresses these challenges.

Week	Lectur	Title	Contents
	8	Changing hats: docker / containers basics	- Student 1: Introduction to Docker - Introduction - Briefly explain the concept of containerization and its significance in modern software development. - Introduce Docker as a popular containerization platform. - Docker Installation Steps - Provide step-by-step instructions for installing Docker on different operating systems (Windows, macOS, Linux). - Anticipate potential installation issues on various platforms and offer troubleshooting tips. - Checking Docker Installation - Guide participants on how to verify a successful Docker installation. - Demonstrate basic Docker commands like docker --version and docker info. - Troubleshoot common issues such as permission errors and missing dependencies. - Running Hello World Image - Explain the concept of Docker images and containers. - Walk through the process of running the "Hello World" Docker image. - Address potential challenges such as image download errors or connectivity issues. - Student 2: Understanding Docker Concepts - Introduction - Advantages of containers. - Introduce the core concepts: images, containers, and how they differ from virtual machines. - Differences Between Images and Containers - Define Docker images and containers and explain their relationship. - Illustrate with examples the distinction between a static image and a running container. - Differences with Virtual Machines - Compare and contrast containers with virtual machines. - Discuss resource efficiency, startup times, and resource isolation. - Address common misconceptions and potential challenges in transitioning from VMs to containers. - Layers and Image Build - Explain the concept of Docker image layers. - Guide participants through a functional and easy image build using a provided Dockerfile. - Anticipate issues related to Dockerfile syntax errors and best practices in image creation. - Introduce the concept of remote image repositories like Docker Hub. - Student 3: Advanced Docker Usage - Executing containers - Execution permissions of the docker daemon - Mounting filesystems points and their permissions - User and group binding - Interactive Execution of Containers - Container Management - Introduce basic container management commands (docker ps, docker stop, docker rm). - Discuss the significance of container names and IDs. - Anticipate challenges with container management, such as conflicts with container names. - Detaching and Reattaching Containers - Discuss scenarios where detaching and reattaching containers are useful. - Demonstrate how to detach from a running container without stopping it. - Address potential challenges with detaching and reattaching, such as terminal disconnections.
3	9	Consolidate "changing hats" session	- Consolidate the changing hats session: - Briefly recap the key points covered by each student in the previous Changing Hats session. - Discuss difficult points and bring up elements that were not covered in sufficient depth or with clarity - Introduction to containerization technologies (Docker, Singularity/Apptainer) and their role in deploying environments. - Benefits of containerization for reproducibility, portability, and ease of deployment. - Security: docker/kubernetes vs singularity (apptainer)
	10	Deploying Environments with Containers	- Consolidate the changing hats session - Introduction to containerization technologies (Docker, Singularity/Apptainer) and their role in deploying environments. - Benefits of containerization for reproducibility, portability, and ease of deployment. - Security: docker/kubernetes vs singularity (apptainer) - Developing/debugging with containers - example with Python
	11	Parallel and distributed computing on HPC (1)	- Lookback into unit 202.2 (OS and concurrence) - Shared vs. distributed memory (parallel vs distributed) - Architectures - Embarrasingly parallel solutions across nodes (continuation from Lecture 5)
	12	Parallel and distributed computing on HPC (2)	- Overview of parallel programming models (MPI, OpenMP, CUDA). - Hands-on examples and exercises to reinforce programming concepts. - Discussing challenges and best practices in HPC programming.
4	13	Hands-on: peer-programming	- Use DISCO and launch a parametric job with launcher and with job-arrays - Create and fine tune the SLURM configuration the sbatch file for a particular application - Use DISCO to benchmark I/O performance on an HPC system. - Deploy a sample application on both HPC and a cloud platform. - Create a containerized environment using Docker and Singularity. - Develop a parallelized algorithm using MPI. - Implement a parallelized task using OpenMP. - Build a Docker image to execute Jupyter notebooks with an AI environment (e.g., with PyTorch) - Review and troubleshoot Dockerfiles from previous sessions.
	14	Managing Containers at Scale with Kubernetes	- Overview of Kubernetes and its role in orchestrating containerized applications. - Discussing how Kubernetes addresses challenges in deploying and managing distributed systems. - Practical exercises on deploying and scaling applications with Kubernetes. - Monitoring and Managing Resources - Touch on the importance of monitoring resources in a Kubernetes cluster. - Introduce tools and practices for managing resources effectively.
	15	Intro to ML through scikit-learn	- Lookback/sync with 301 (ML briefing) - Introduction and general design considerations - Navigating the documentation - Core functionalities and key modules. - Essential utilities for data manipulation and model evaluation. - Supervised and unsupervised learning algorithms. - Model evaluation using metrics. - Showcasing guided exercises from the documentation
	16	Changing hats: scikit-learn	- Student 1: Feature Engineering in scikit-learn. - Describe feature engineering techniques available in scikit-learn. - Discuss the impact of feature engineering on model performance. - Provide practical examples and case studies showcasing effective feature engineering. - Student 2: Hyperparameter Tuning with Grid Search and Random Search. - Dive into hyperparameter tuning using Grid Search and Random Search in scikit-learn. - Compare the advantages and limitations of each approach. - Demonstrate how to implement hyperparameter tuning on a specific machine learning model. - Student 3: Ensemble Learning in scikit-learn. - Introduce ensemble learning concepts and algorithms available in scikit-learn. - Compare and contrast different ensemble methods (e.g., RandomForest, AdaBoost). - Present real-world applications and use cases where ensemble learning is beneficial.
	17	Using scikit-learn (consolidating "changing hats")	- Consolidate the changing hats session: - Briefly recap the key points covered by each student in the previous Changing Hats session. - Discuss difficult points and bring up elements that were not covered in sufficient depth or with clarity - Feature engineering and preprocessing - Building data pipelines - Cross-validation and nested cross-validation - Parallelization - Example combining a pipeline and comparing cross-validation vs. nested cross-validation

Week	Lectur	Title	Contents
5	18	Ensemble Learning (with scikit-learn) and Distributed ML	- Define Ensemble Learning and its motivation. - Ensemble Methods: - Bagging (Bootstrap Aggregating): Reduce variance with methods like Random Forest. - Boosting (AdaBoost, Gradient Boosting): Correct model errors iteratively. - Stacking: Combine predictions from diverse models. - Implementation and Optimization - Considerations for setting up ensemble models. - Strategies for optimizing ensemble hyperparameters. - Metrics and techniques for assessing ensemble performance. - Introduce Canonical Ensemble Learning as a precedent of energy-based models introduced in lecture 49. - Discuss how ensemble methods can be adapted and optimized for distributed machine learning environments. - Case Studies and Examples
	19	Hands-on: peer-programming	- Implement a Random Forest classifier using bagging. - Employ AdaBoost ensemble for a classification problem. - Build a stacking ensemble using diverse base models - Extend a basic ensemble training script for distributed computing - Modify an ensemble approach for handling large-scale datasets - Develop a nested cross-validation script for hyperparameter tuning of an ensemble model - Implement a mechanism to dynamically adjust the size of an ensemble during training - Integrate feature engineering techniques into the ensemble learning pipeline. Explore how feature transformations impact the overall performance of the ensemble. - Scale ensemble learning for a distributed computing cluster - Replicate and extend a case study presented in the previous session
	20	Data pipelines and HPC	- Definition and significance of data pipelines in scientific computing. - Overview of challenges and benefits in deploying data pipelines in HPC. - Lookback to scikit-learn's pipelines - Introduction to key concepts: - data provenance, - pipeline synchronization/communication, - resource management and allocation, - job dependency matrix, - error handling and fault tolerance, - reproducibility, - scalability. - Deploying data pipelines on HPC - Understand the overlap and collaboration between data pipelines and HPC scheduling
6	21	Nextflow: A Scalable and Reproducible Data Workflow Engine	- In-depth exploration of Nextflow for building scalable and reproducible workflows. - Hands-on demonstration of creating a simple data pipeline with Nextflow. - Discussion on features, syntax, and best practices.
	22	Python's Dask for Parallel Computing and Scalable Data Analytics	- Introduction to Dask and its role in parallel computing and data analytics. - Hands-on exercises demonstrating parallel computing with Dask. - Discussion on integrating Dask with HPC environments.
	23	Nipype: A Python Framework for Neuroimaging Pipelines	- Overview of Nipype and its application in neuroimaging pipelines. - Practical examples of building and executing neuroimaging workflows. - Discussion on interfacing with existing tools and ensuring reproducibility.
	24	"Changing hats" session (data pipelines)	- Student 1: A data pipeline with Nextflow, deployed on DISCO - Student 2: A data pipeline with Dask, deployed on DISCO - Student 3: A data pipeline with Nipype, deployed on DISCO
7	25	Comparative Analysis of Data Pipeline Engines	- Consolidate the changing hats session: - Briefly recap the key points covered by each student in the previous Changing Hats session. - Discuss difficult points and bring up elements that were not covered in sufficient depth or with clarity - Comparative review of Nextflow, Dask, Nipype, and other notable engines. - Strengths, weaknesses, and use cases for each engine. - Learn how to introspect the pipeline and understand the bottlenecks of execution - Discussion on choosing the right engine based on specific requirements.
	26	Hands-on session	- Build a basic Nextflow workflow: Create a simple Nextflow workflow with two processes. - Parallelize data processing with Dask: Use Dask to parallelize a data processing task. - Nipype for neuroimaging pipeline: Construct a Nipype pipeline for a basic neuroimaging task. - Integrate Nextflow with HPC resources: Modify a Nextflow workflow for an HPC cluster. - Data pipelines for machine learning: Create a data pipeline for ML data preprocessing. - Distributed computing with Dask: Design a Dask computation for distributed computing. - Enhance Nipype workflow with custom interfaces: Extend a Nipype workflow with custom interfaces. - Data pipeline optimization challenge: Identify and optimize a pipeline bottleneck. - Data provenance and reproducibility: Implement data provenance tracking in a pipeline. - Error handling and resilience in pipelines: Introduce errors and implement error handling strategies.
	27	Foundations of Deep Learning	- Introduction to Deep Learning: - Definition and significance in modern machine learning. - Brief historical overview. - Neural Network Basics: - Detailed discussion on neural network architecture. - Explanation of layers, neurons, and their interconnections. - Activation functions and their role in introducing non-linearity. - Illustration of forward and backward passes in a neural network.
	28	Training Deep Neural Networks	- Overview of the training process in deep learning. - The role of data in supervised learning. - The importance of differentiability and its role in optimization - Training Deep Neural Networks and key training concepts: - Batches: Definition, significance, and impact on training. - Epochs: Understanding the concept of one complete pass through the entire dataset. - Learning rate: Importance, tuning strategies, and common challenges. - Loss functions: Overview and selection based on task. - DL frameworks: TensorFlow, PyTorch, Keras, etc.
8	29	Introduction to PyTorch Fundamentals	- Overview of PyTorch and its role in deep learning. - Installation and setup of PyTorch. - Introduction to PyTorch tensors and basic operations. - Building a simple neural network using PyTorch.
	30	Deepening PyTorch Concepts	- Exploring PyTorch modules and autograd for automatic differentiation. - Defining and training more complex neural network architectures. - Guided exercises to reinforce PyTorch fundamentals. - Discussion on best practices for PyTorch development. - Debugging with PyTorch: gathering, understanding and processing failure evidence
	31	PyTorch for Computer Vision and Natural Language Processing	- Introduction to PyTorch's torchvision and torchaudio libraries. - Building and training models for computer vision tasks. - Utilizing pre-trained models for transfer learning. - Basic natural language processing tasks using PyTorch.
	32	"Changing hats" session (PyTorch)	Student 1: Monitoring Training in PyTorch - Explore different techniques for monitoring and visualizing training progress in PyTorch. - Discuss the use of TensorBoard, PyTorch Ignite and Lightning, and - Present best practices for tracking metrics, loss curves, and model performance during training. Student 2: Transfer Learning with PyTorch - Provide an in-depth overview of transfer learning techniques using PyTorch. - Discuss pre-trained models, feature extraction, and fine-tuning strategies. - Demonstrate how to implement transfer learning on a practical project or dataset. Student 3: Convolutional Networks in PyTorch - Dive into the fundamentals of convolutional neural networks (CNNs) in PyTorch. - Explain the architecture of a basic CNN and its components (convolutional layers, pooling layers). - Walk through the implementation of a simple image classification task using a CNN in PyTorch.

Week	Lectur	Title	Contents

9	33	Consolidate "changing hats" session	- Consolidate the changing hats session:
			  - Briefly recap the key points covered by each student in the previous Changing Hats session.
			  - Discuss difficult points and bring up elements that were not covered in sufficient depth or with clarity
			
			- Introduction to Transfer Learning
			  - Definition and motivation behind transfer learning.
			  - Explanation of pre-trained models and their role in transfer learning.
			  - Demonstration of how to implement transfer learning in PyTorch.
			
			- Distributed Hyperparameter Optimization in PyTorch with Ray Tune

	34	Hands-on: peer-programming	- Implement a simple CNN for image classification using the CIFAR-10 dataset. - Apply transfer learning with a pre-trained CNN for image classification. - Develop an RNN for text classification using a small or synthetic dataset. - Extend an image classification project to incorporate data parallelism. - Generate images using a GAN and implement model parallelism for efficiency. - Combine transfer learning and model parallelism for image classification. - Implement an RNN for time series prediction using synthetic or real data. - Apply data parallelism to a regression task and compare performance. - Create an image captioning model using a sequence-to-sequence architecture. - Implement model parallelism in a project tailored to a custom dataset.

	35	Parallel programming on HPC requiring GPU resources	- (Big) Data and Model Size - Why Distributing the Workload?
			  - Explore the challenges posed by large datasets and complex models in traditional computing environments.
			  - Understand the need for distributing workloads to efficiently process extensive data and intricate models.
			  - Discuss the limitations of sequential processing and how parallelization can address these challenges.
			- Parallelization Within a Node with Several GPUs
			  - Dive into the architecture of a single computing node equipped with multiple GPUs.
			  - Explore techniques for parallelizing computations within a node, leveraging the parallel processing capabilities of individual GPUs.
			  - Understand the nuances of load balancing and optimizing performance when multiple GPUs collaborate within a single node.
			- Parallelization Across GPU Nodes
			  - Extend the discussion to parallelization across multiple nodes, forming a distributed GPU computing environment.
			  - Examine the communication challenges and strategies for coordinating computations across GPU nodes.
			  - Discuss frameworks and libraries that facilitate distributed GPU computing, ensuring efficient parallel processing at scale.

	36	Changing hats: HPC parallelization	- Student 1: Diving into MPI, OpenMP, and CUDA for parallelization
			  - Present the implementation of a parallelized algorithm using MPI for distributed computing.
			  - Explain shared-memory parallelization using OpenMP on an HPC node.
			  - Showcase GPU-accelerated code using CUDA for efficient parallel processing.
			  - Mention challenges and considerations in parallelizing algorithms with different frameworks.
			
			- Student 2: Parallelization with TensorFlow
			  - Present the implementation of a parallel machine learning model using TensorFlow.
			  - Explain distributed training strategies with TensorFlow.
			  - Discuss optimization techniques for parallelizing computations in TensorFlow.
			  - Enumerate some challenges and best practices in parallelizing deep learning models.
			
			- Student 3: Parallelization with PyTorch
			  - Present the implementation of a parallel machine learning model using PyTorch.
			  - Explain PyTorch's native support for parallel computing.
			  - Discuss strategies for optimizing and scaling PyTorch-based applications.

10	37	Consolidate "changing hats" session	- Consolidate the changing hats session:
			  - Briefly recap the key points covered by each student in the previous Changing Hats session.
			  - Discuss difficult points and bring up elements that were not covered in sufficient depth or with clarity
			
			- Describe the reasons to distribute machine learning models:
			  - Sizing the problem dimensions (data volume and model size)
			  - Privacy/security/access
			  - Examples: model ensembles and federated learning.
			- Understanding what can be paralellized:
			  - Model training: distributing the workload across multiple devices or nodes
			  - Model inference: parallelizing the prediction process
			  - Data processing: expedite model training by parallelizing feature extraction and engineering
			- Challenges of parallelizing ML applications: complexity, I/O overhead, synchronization, data consistency, scalability, fault tolerance.

	38	Monitoring and Control of HPC, with a focus on distributed ML models	- Monitoring in HPC - introduction and tools
			  - Explain the critical role of monitoring in ensuring the efficiency, reliability, and performance of HPC systems.
			  - Discuss the impact of monitoring on resource utilization, fault detection, and overall system health.
			  - Introduce popular monitoring tools in HPC, such as Ganglia, Nagios, and Prometheus.
			  - Discuss key metrics for monitoring, including CPU utilization, memory usage, network traffic, and disk I/O.
			  - Relevance of real-time monitoring for identifying bottlenecks and optimizing resource allocation.
			  - Live Demonstration using a monitoring tool to showcase real-time data visualization and analysis on DISCO
			
			- Monitoring Distributed ML Models on HPC
			  - Challenges in Monitoring ML Workloads
			  - Monitoring Frameworks for ML Models, such as TensorFlow's TensorBoard and PyTorch's built-in monitoring capabilities.
			  - Discuss the integration of these frameworks with HPC systems for comprehensive monitoring.
			
			- Case Study highlighting the successful implementation of monitoring strategies for distributed ML models on HPC.
			
			- Control and Optimization Strategies
			  - Dynamic Resource Allocation: introduction and tools
			  - Discuss how monitoring data can inform adjustments to hyperparameters for optimal model performance.

	39	PyTorch's distributed package	- Overview of PyTorch distributed AI capabilities - Introduce DDP, RPC, and c10d - Recap of data parallelism and its applications - Recap of model parallelism and its applications - Recap of inference parallelism and its applications
	40	Data parallelism in PyTorch	- Running single-machine, multi-GPU DataParallel and the role of CUDA parallelizing data splits - Running multi-machine DistributedDataParallel to scale across nodes - Introduce how the model is kept synchronized across workers that are training on different data splits - Introduce "sharding" and the FullyShardedDataParallel training
11	41	Model parallelism in PyTorch	- Running simple PyTorch models with several GPUs in parallel - Pipelining inputs to optimize model parallelization - Pipeline parallelism using multiple GPUs (torch.distributed.pipeline) - Model parallelism across compute nodes - introduction to the Distributed RPC framework
	42	Data & model parallelism in PyTorch	- Consolidate the concepts of the two previous lectures - Introduce torch.distributed.elastic for fault-tolerant and dynamic allocation training - Mixed-precision training using torch.cuda.amp - Gradient accumulation for larger effective batch sizes - Wrap-up overview of distributed training - Explore paramount real-world examples of parallel techniques for training large models
	43	Parallelizing inference in PyTorch	- CPU parallelization - GPU parallelization - Edge devices and distributed inference - Federated learning as a combination of large models that require parallelization and AI on edge devices
	44	Hands-on session: PyTorch	- Create and run a simple PyTorch model on a single machine with multiple GPUs using DataParallel. - Create and run a simple PyTorch model using DistributedDataParallel across multiple machines. - Implement a mechanism to keep a model synchronized across workers training on different data splits. - Experiment with sharding techniques and implement FullyShardedDataParallel training. - Run a PyTorch model on a machine with several GPUs in parallel, exploring GPU parallelization. - Implement input pipelining techniques to optimize model parallelization. - Extend model parallelism to operate across compute nodes, using the Distributed RPC framework. - Introduce and experiment with torch.distributed.elastic for fault-tolerant training - Implement mixed-precision training on an existing model using torch.cuda.amp for faster computation. - Experiment with gradient accumulation techniques to achieve larger effective batch sizes, optimizing training efficiency.
	45	Generalizability of Inference and Domain Shifts	- Definitions and background: generalizability and domain shift - Sources of bias that set boundaries to generalizability - The relationship between the training dataset and biases - Mitigating biases and ensuring the diversity and representativeness of the dataset - Algorithmic fairness and ethical implications

Week	Lectur	Title	Contents
12	46	Cross-Validation and Nested Cross-Validation	- Re-introduce cross-validation as a robust model evaluation technique - K-fold cross-validation - Stratification and operation on "imbalanced datasets" - Nested cross-validation for robust model selection and evaluation - Model selection as a source of bias - Showcase examples where nested cross-validation demonstrates model selection biases - Batch-effects (genomics), site-effects (other fields)
	47	Domain Adaptation and Transfer Learning	- Briefly recap theory on generalizability and domain shifts - Zero-shot learning, domain generalization, test-time adaptation and examples - Objectives of domain adaptation - Adversarial training as a technique for domain adaptation by minimizing domain discrepancy - Concept of task, multi-task learning, and lifelong learning - Transfer learning: practical implementation
	48	Transfer Learning with PyTorch	- Navigate the pre-trained models avalable within torchvision and their applications - Investigate fine-tuning procedures using PyTorch - Tutorial: transfer learning of a torchvision model

13	49	Generative Models 1	- From discriminative to generative models.
			- Introduce probabilistic graphical models as a foundation for generative models.
			  - Latent variables
			- Classical generation: energy-based models (EBMs)
			- Generative Adversarial Networks (GANs)
			  - Adversarial training process in depth

	50	Generative Models 2	- Explore Variational Autoencoders (VAEs) and their use in generative modeling. - Encoder-decoder architectures - Flow-based generative models and differences with VAEs and GANs - Autoregressive models (ARMs) - Showcase applications of generative models in image synthesis, text generation, etc.

	51	Transformers	- Precedents: Recurrent Neural Networks, long short-term memory (LSTM) networks
			- Introducing ML attention
			  - Implementation of the self-attention mechanism
			- Transformers and their revolutionary impact on natural language processing.
			- Architecture of transformers:
			  - Encoder-decoder and attention layers
			- Application of transformers on NLP

	52	Graph Learning	- Introduction to graph learning and the model representation of graph data - Foundations of graph neural networks (GNNs) - Message passing and other GNNs operations - Showcase social network analyses
14	53	Diffusion Models	- Recap of flow-based models seen in lecture 50 - Diffusion models and their fundamental principles. - Stochastic processes and how diffusion models capture data evolution. - Explore how diffusion models can be applied to denoise images and image restoration. - Super-resolution, its importance in image processing, and its implementation with diffusion models - Introduce latent diffusion as a powerful approach for generative modeling.
	54	Topic hold and/or Advanced optimizations and tools	- Recover topics that may require an additional recap - Mention torch_xla to execute PyTorch on XLA devices (e.g., TPUs) - Introduce torch.cuda.amp to implement mixed-precision training - Introduce Google's JAX - Wandb (weights & biases) developer platform
	55	Hands-on sessions	- Dataset audit: find biases in openly available datasets - Using pre-processing techniques to mitigate biases in a given dataset. - Document ethical implications related to biases in machine learning. - Implement K-fold cross-validation on a dataset for model evaluation. - Apply stratification techniques for handling imbalanced datasets. - Perform nested cross-validation for robust model selection and evaluation. - Investigate examples showcasing biases in model selection using nested cross-validation. - Explore batch-effects in genomics and site-effects in other fields. - Implement transfer learning using pre-trained models from torchvision. - Fine-tune a pre-trained torchvision model using PyTorch for a specific task.
	56	Federated Learning (FL) 1	- Significance of FL in the context of distributed machine learning. - Discuss the motivation behind FL and its applications. - Contrast centralized vs. decentralized learning approaches. - Training models on decentralized devices - Components and their communication: server, clients, and FL algorithm - Privacy concerns in traditional machine learning models. - Keeping data local: privacy-preserving applications. - Secure aggregation techniques - Applications on healthcare, finance, and edge computing
15	57	FL 2	- Optimization techniques specific to FL. - Reducing communication overhead with quantization and sparsification - Ultra-light DNN architectures - FL frameworks (e.g., TensorFlow Federated, PySyft). - Federated averaging and model aggregation. - Challenges: non-IID data, communication bottlenecks, and model heterogeneity. - Technical limitations
	58	Tutorial: implementing FL with PySyft	- PySyft and its purpose in the context of FL. - Installation and compatibility with PyTorch - PySyft workers: concept, initialization and communication - Secure multi-party computation, encrypted tensors, and computation on encrypted data - Showcase an application using PySyft
	59	Advanced Topics in Distributed AI	- Apply reinforcement learning concepts to distributed scenarios. - Discuss emergent behavior in multi-agent systems. - Explore how local interactions lead to global patterns. - Explore emerging trends in distributed AI such as edge computing
	60	Exam preparation	- A sessions where students can ask questions about the contents and about the exam - Some parts of the exam will be hinted, so that the students can understand the priorities over the content
16	61	Invited lecture	Fabian Pedregosa (JAX, Google Inc) or Chris Gorgolewski (Bard, Google Inc.)
	62	Exam	
	63	Exam revision	
	64	Recap and evaluation	- Open format where students can ask about any of the contents seen over the course