Collaborative Pilot Dual PhD and/or Master Projects

between

Toulouse Computer Science Research Institute

&

Institute for Informatics and Automation Problems

on

High-Performance and Scalable Computing
AI for science & AI for HPC
Data-Intensive Simulation and Modeling
Serverless and Frugal Computing

Studies in Disaggregated Memory Systems (Prof. Daniel Hagimont, Dr. Camelia Slimani) 2

Adaptive, Uncertainty-Aware Hierarchical Inference for Communication- and Energy-Constrained Edge-Cloud-HPC Continuums 2

In-Memory Processing and Elastic Memory Management with RDMA for High-Performance Computing: Navigating Irregular Workloads in HPC and Serverless 4

Scalable and Secure Computing for Life Sciences Workflows (Dr. Ronan Guivarch,) 7

Confidential Serverless Computing for HPC Clusters 7

Frugal Computing Across the Cloud-Edge Continuum 9

Optimizing Deep Learning Storage Pipelines with Converged HPC and Cloud-Native Systems 10

Storage-Aware Pruning and Optimization of Large Neural Models for High-Performance Computing 11

Remote Sensing and Earth Observation Data Processing (Dr. Alena Kopanicakova) 12

Temporal Transfer Learning for Remote Sensing Time Series - Territory of Armenia 12

Enhancing Land Cover Classification: Comparative Analysis of Diverse Classification Methods and Ensemble Learning Application 14

2026

Studies in Disaggregated Memory Systems (Prof. Daniel Hagimont, Dr. Camelia Slimani)

Adaptive, Uncertainty-Aware Hierarchical Inference for Communication- and Energy-Constrained Edge-Cloud-HPC Continuums

Context

Deploying deep learning models across edge–cloud architectures introduce major challenges, particularly when communication between the edge and the cloud is constrained. In many real-world scenarios, the available bandwidth is highly limited. For example, LoRaWAN typically offers 250 bit/s–11 kbit/s versus 50–150 Mbit/s for 4G LTE, forcing a rethink of how data and computation are split between edge and cloud. Edge intelligence aims to bring inference closer to the data source to reduce communication overhead. However, edge devices usually have restricted computational and memory resources, making it difficult to run full-scale deep neural networks locally. Conversely, lightweight or tiny models are faster and more energy-efficient but often yield lower accuracy, especially in tasks involving large numbers of output classes. A common property in many machine learning datasets is their class imbalance: the majority of samples belong to a small subset of frequent classes. This observation motivates a hybrid strategy:

- Deploy a tiny model at the edge to handle the most frequent or "easy" classes locally,
- Offload the remaining cases to the cloud for more complex processing.

This selective inference mechanism acts as a filter, transmitting data to the cloud only when necessary, and can significantly reduce communication costs while maintaining acceptable accuracy. However, such ideal scenarios are not always realistic for the following reasons: the class distribution changes over time; connectivity is not always consistent and can become unreliable; and infrastructure may include more than two levels (edge, fog, cloud, HPC). A static partitioning of edge/cloud with a confidence level is unable to cope with the situation. This is what the thesis focuses on.

Objective

The project will explore the feasibility and performance gains of hierarchical or split inference architectures that combine edge processing and cloud offloading, with a particular focus on:

- Reducing data transfer volume;
- Improving inference latency;
- Balancing accuracy versus communication cost.

Methods

This study employs methodologies from deep learning, distributed computing, and optimization.

- Hierarchical Inference: The low-complexity edge model processes common or simple classes, whereas complex or rare instances are forwarded to the high-capacity cloud model.
- Confidence-Based Routing: A threshold or learned criterion determines whether a local prediction is accepted or escalated to the cloud model.
- Characterization of Workload: The characteristics and distribution of classes are analyzed to establish appropriate threshold
- Performance Modeling: Latency, communication overhead, and energy consumption are measured and simulated under varying bandwidth conditions.

- Drift detection and adaptation: online statistical drift-detection techniques combined with adaptive model retraining or re-thresholding, enabling the routing policy to track data distribution drift.
- Joint routing-compression design: multi-objective optimization (NSGA-II, for example) that adjusts routing criteria and model compression/early exit simultaneously instead of individually optimization.

Action Plan

- **Phase 1 (Design):** Define the edge/cloud model split, the routing criterion and characterize target datasets for class imbalance, and study drift scenarios and multi-tier network topologies.
- **Phase 2 (Model Development):** Train the tiny edge model and the cloud model, implement and tune the confidence-based routing mechanism and drift-aware recalibration for the routing threshold.
- **Phase 3 (Integration):** Connect edge and cloud components via a lightweight communication protocol and implement the end-to-end inference pipeline. Extend the pipeline to a multi-tier (edge–fog–cloud–HPC) testbed with an RL-based routing agent
- **Phase 4 (Evaluation):** Benchmark accuracy, latency, communication volume, and offload rate against centralized (cloud-only) and static (edge-only) baselines. Test sensitivity to bandwidth constraints and class-imbalance severity.
- **Phase 5 (Analysis):** Identify accuracy/communication trade-off points and derive practical guidelines for deploying hierarchical inference in bandwidth-constrained settings.

Software & Tools

- Deep Learning: PyTorch, TensorFlow, ONNX Runtime
- Edge AI: TensorFlow Lite, OpenVINO
- Communication: gRPC / REST APIs
- Simulation & Analysis: Python, NumPy, SciPy
- Reinforcement Learning / Optimization: Ray RLlib, PyGMO
  Network Simulation: ns-3

Datasets

CIFAR-10-LT, CIFAR-100-LT, ImageNet-LT (long-tailed benchmarks with built-in class imbalance), plus optional edge/IoT sensing data for realistic deployment testing.

Expected Contributions

- A working hierarchical edge–cloud inference architecture with a confidence-based routing mechanism.
- An empirical characterization of the accuracy–communication trade-off under class imbalance.
- A quantitative comparison against centralized and static-edge baselines across bandwidth regimes.
- A multi-tier (edge–fog–cloud–HPC) reinforcement-learning-based routing framework, extending beyond binary edge/cloud decisions.
- A joint optimization framework co-designing edge model compression and routing policy, evaluated against independently-tuned baselines.
- Practical deployment guidelines for selective inference in bandwidth-constrained environments.

In-Memory Processing and Elastic Memory Management with RDMA for High-Performance Computing: Navigating Irregular Workloads in HPC and Serverless

Context

With the increasing amount of data being processed today by applications such as machine learning workloads, databases, and high-performance computing (HPC), memory subsystems are under increasing stress. The primary source of this performance overhead is the data transfer between the processor and main memory, which accounts for most execution time and energy. To solve this problem, the concept of Processing In Memory (PIM) proposes moving computations closer to data. PIM is an emerging technology currently being developed by leading tech companies such as Samsung and UPMEM to enable computation within memory itself and minimise data transfer. While there have been many works that utilise PIM to accelerate compute-intensive or memory-bound kernels (graph analytics, matrix operations, and AI inference), PIM offers capabilities that extend beyond application acceleration. In particular, almost all current research on PIM assumes that it is used as a computational accelerator for each node rather than as a networked, shared memory pool. The aim of this thesis is to fill that gap: memory is often stranded or distributed unevenly between nodes, while each node maintains its own local swap space independently. There is currently no architecture that allows PIM to be treated as an additional memory layer accessible to different nodes via RDMA.

The goal is to utilize PIM as swap space as an alternative or supplement to the existing techniques of virtual memory management using secondary disk storage. This innovative technique stems from the following two main benefits:

- Latency and bandwidth optimization: Accessing PIM takes significantly less time than accessing regular secondary storage. Its utilisation as swap space would greatly diminish paging overheads.
- Computation capabilities of PIM: As PIM has computing units on board with memory cells, it will be able to perform compression/decompression operations on the data on board

Objective

Objective of the Project is to analyze the feasibility and performance of using the concept of Processing-In-Memory with In-Memory Compression & Decompression of Data as a Swap Method. This topic will cover software prototype development and performance analysis, INCLUDING.

- Whether RDMA-enabled PIM can act as an effective shared elastic memory tier rather than just a local memory swapping mechanism;
- What decisions need to be taken in regard to compression/eviction/migration based on RDMA latency, PIM compute capacity, and access type;
- Whether there is any benefit of using this kind of tiering technique for serverless applications, which have bursty memory requirements, over static memory allocation for each function; and
- Which features of irregular memory access make this technique advantageous over others.

Methods

- PIM-based swap prototyping: design a swap flow that would be optimised for PIM-capable memory (UPMEM DIMMs, for example) rather than, or in addition to, disk swap by performing interception of paging either at the OS or application-level.

- In-memory compression: perform compression/uncompression of swap pages on PIM processors with the help of efficient algorithms that are appropriate for computing resources available to PIM (LZ4-family or delta encoding, for example).
- Performance evaluation: assess latency, throughput, and memory capacity improvements provided by a PIM swap vs a typical SSD/HDD swap.
- Workload evaluation: examine performance under memory-intensive and non-uniform workloads (for example, graph applications) to identify where PIM swap is most beneficial.
- Disaggregation with RDMA: generalise the single-machine PIM swapping flow into an approach in which remote machines use RDMA verbs to utilise PIM resources as a shared cluster resource, rather than just a machine-level resource.
- Tiering policy for dynamic tiering: develop a policy decision-making algorithm (partially running on the PIM compute) which makes the choice between compress-and-store, move-and-not-compress, or evict, depending on the measured latency between RDMA round trip, PIM compute utilisation, and page access frequency.
- Serverless: integrate with a serverless environment (OpenFaaS, custom lightweight sandbox) to see whether it is possible to run a function that utilises RDMA-coupled PIM memory as elastic storage.

Action Plan

- **Phase 1 (Design):** Review PIM architectures (UPMEM, Samsung PIM) and their programming/memory-mapping models, and define the swap interception mechanism and the interface between host memory management and PIM.
- **Phase 2 (Prototype Development):** Implement a baseline PIM-as-swap path without compression and integrate a compression/decompression kernel running on PIM.
- **Phase 3 (Integration):** Validate the correctness of paging and compression under representative workloads, and tune compression parameters to balance PIM compute/memory trade-offs. Extend the prototype to multiple nodes via RDMA, enabling remote access to PIM capacity.
- **Phase 4 (Evaluation):** Benchmark paging latency, effective capacity, and processor offload against disk-based swap baselines, and analyse sensitivity to workload access pattern (regular vs irregular) and data compressibility.
- **Phase 5 (Analysis):** Identify workload classes and system configurations where PIM-as-swap is most beneficial.

Software & Tools

- PIM Platforms: UPMEM SDK, (Samsung PIM if accessible)
- Systems: Linux kernel memory management / user-space paging hooks
- Compression: LZ4, Zstd (as compute-budget references), custom lightweight kernels for PIM
- Performance Analysis: Python, perf, custom microbenchmarks

Datasets

Memory-intensive and irregular-access benchmarks representative of HPC and serverless workloads (e.g., graph analytics, sparse matrix operations, in-memory database working sets), used to stress paging behavior and evaluate compression effectiveness.

Expected Contributions

- A proof-of-concept implementation of PIM-based swap space augmented with in-memory compression.
- A quantitative comparison of PIM-as-swap versus traditional disk-based virtual memory under memory-bound and irregular workloads.
- Characterization of the latency, capacity, and processor-offload benefits achievable through in-situ compression.
- Design guidelines for incorporating PIM as an intermediate memory tier in HPC and serverless memory hierarchies.
- Architecture allowing PIM to serve as a cluster-level, elastic memory tier based on disaggregation and RDMA.
- Tiering strategy incorporating network latency, PIM computational slack, and access patterns in the decision-making process regarding compression, migration, and eviction.
- Evaluation of elastic memory powered by PIM for serverless applications with respect to performance improvement over the static function allocation.
- Framework for disaggregated compute-enabled memory tiering that can be applied beyond PIM in future near-memory-compute architectures.

Scalable and Secure Computing for Life Sciences Workflows (Dr. Ronan Guivarch)

Confidential Serverless Computing for HPC Clusters

The research investigates the feasibility and performance of integrating confidential computing paradigms within serverless frameworks deployed across high-performance computing (HPC) environments. The Trusted Execution Environments (TEEs) will be examined to enable secure, scalable, and fine-grained execution of sensitive workloads while maintaining strict data confidentiality and minimizing computational overhead. As biological research increasingly relies on large-scale computational pipelines, securing sensitive genetic information (e.g., patient genomes) and proprietary molecular structures becomes paramount. This study will focus on two real-world computational workflows:

- Variant Calling (Genomics): From raw DNA sequences to identifying mutations.
- Ligand Docking (Structural Biology): Predicting molecule–protein interactions using docking tools.

Objective

The core objective is to design and evaluate a confidential serverless computing framework that integrates with HPC resource managers such as SLURM and container runtimes like Apptainer. This includes developing and benchmarking workflow-specific confidential containers for genomics and structural biology applications. Beyond system development, the project aims to characterize the trade-offs between security guarantees (e.g., confidentiality, integrity) and computational performance. The study seeks scientific insight into the scalability limits, orchestration challenges, and design patterns that enable secure, policy-compliant, and efficient execution of sensitive workloads in multi-tenant HPC systems.

Scientific Methods and Models

- Secure Workflow Modeling: DAG-based task decomposition for pipelines using Nextflow.
- Confidential Computing Abstractions using Intel SGX enclaves.
- Allocation Algorithms: greedy heuristics, multi-objective optimization (NSGA-II), reinforcement learning.
- Security Modeling: Threat model formalization, confidentiality & integrity checks, TCB (Trusted Computing Base) minimization.

Action Plan

- Infrastructure Setup: Deploy FaaS over SLURM with confidential container runtime support (Apptainer).
- Pipeline Containerization: Genomics: BWA → GATK → Mutect2; Structural Biology: Structure Analysis (nglview+ProDy) -> Ligand sanitization (RDKit) → Docking (AutoDock Vina,etc.)
- Model Design: Implement stochastic simulators and scheduling algorithms to test workload behavior under real HPC trace patterns.
- Security & Performance Analysis: Evaluate TEEs for I/O latency, startup overheads, and data confidentiality.
- Multi-Scenario Simulation: Test workflows across different data sizes, resource configurations, and threat levels.

Software & Tools

- Workflow Engines: Snakemake, Nextflow
- Bioinformatics: BWA, GATK, Mutect2
- Docking Tools: AutoDock Vina
- Confidential Containers: Intel SGX SDK
- Serverless Platforms: OpenFaaS, Knative, Kubeless
- Schedulers: SLURM, Kubernetes
- Modeling/Simulations: Python (SimPy, NumPy, Scikit-Learn, RDKit, ProDy, nglview), R,
- Datasets: Genomics: 1000 Genomes, TCGA (public subset) and Structural Biology: PubChem, ChEMBL, BindingDB, PDB

Frugal Computing Across the Cloud-Edge Continuum

This research investigates the optimization of heterogeneous computing infrastructures for energy-efficient and scalable execution of biological workflows. It leverages ZLUDA, a compatibility layer that enables CUDA applications on non-NVIDIA GPUs (e.g., Intel, AMD), to extend bioinformatics pipelines beyond conventional GPU clusters. With growing demands in computational biology and increasing attention to sustainability, the study evaluates how lightweight, resource-aware workflows can maintain performance and accuracy across energy-constrained cloud-edge systems. The work targets on genomics (RNA-seq quantification, metagenomics) and structural biology (protein structure prediction), serving as representative cases for benchmarking computational and energy trade-offs in frugal computing environments.

Objective

To quantitatively evaluate the trade-offs between energy efficiency, performance, and biological accuracy when executing compute-intensive life science applications across cloud-edge platforms. Particular emphasis will be placed on the viability of ZLUDA as a middleware for broadening GPU support in high-performance bioinformatics pipelines. The study will also examine how workload characteristics (e.g., data size, model complexity, memory bandwidth usage) influence scheduling, offloading decisions, and placement strategies across the continuum. Ultimately, the goal is to derive adaptive workload placement strategies and profiling-driven models that inform optimal resource utilization without compromising scientific reproducibility or accuracy.

Implementation Plan

- Evaluate and master Intel VTune, NVIDIA Nsight, and ROCm-SMI, nvtop to collect runtime and energy metrics across diverse cloud-edge platforms.
- Develop statistical and machine learning models to relate workload features to power, latency, and temperature.
- Design adaptive scheduling strategies using reinforcement learning or heuristics for optimal task placement across cloud, edge, or hybrid nodes.
- Benchmark ZLUDA's performance by running CUDA-intensive bioinformatics workflows on Intel/AMD GPUs and comparing results with native NVIDIA systems.
- Analyze trade-offs between energy consumption, execution time, and scientific accuracy to optimize frugal deployments of biological pipelines.

Available tools

- Genomics Tools: Salmon, DESeq2, Kraken2, MetaSPAdes
- Structural Biology Tools: Boltz, AlphaFold3
- Middleware & Runtimes: ZLUDA, Docker, Apptainer, Kubernetes (for edge orchestration)
- Monitoring Tools: Prometheus, Grafana, Powercap APIs
- Datasets: SRA (e.g., PRJNA600892), NCBI RefSeq (for metagenomes), UniProt, CASP, AlphaFold DB, PDB Bank

Optimizing Deep Learning Storage Pipelines with Converged HPC and Cloud-Native Systems

Objective

To design, implement, and evaluate an optimized deep learning (DL) storage pipeline that leverages the strengths of both HPC parallel file systems (e.g., LustreFS) and cloud-native object storage (e.g., MinIO). The project aims to reduce I/O bottlenecks and improve training efficiency of state-of-the-art sequence models such as Mamba by employing dynamic optimization strategies across storage configurations.

Methods

- System Architecture Design - Develop a converged storage platform combining Lustre (high-speed parallel filesystem) with MinIO (scalable object storage). Introduce an interoperability layer that allows transparent data movement and caching between file systems and object stores.
- Workload Profiling and Tracing - Analyze the I/O characteristics of Mamba-based models (e.g., Mamba-S, Mamba-2) during training on representative datasets (e.g., ImageNet, The Pile, etc.). Trace file access patterns, block sizes, metadata calls, and concurrency characteristics.

Optimization Framework

- Implement a multi-armed bandit or Bayesian optimization-based system for dynamic tuning of storage parameters (e.g., striping, chunk size, replication factor, data placement).
- Use reinforcement learning to adapt strategies based on workload feedback (e.g., latency, throughput, model loss).

Benchmarking and Evaluation

- Use the MLPerf Storage benchmark suite and custom Mamba training pipelines to measure:
- Read/write throughput
- Metadata operation latency
- Checkpoint/resume performance
- Total model training time
- Compare with baseline setups: Lustre-only, MinIO-only, and static hybrid configurations.
- Scalability and Fault Tolerance Study
- Evaluate system behavior under scale-out settings, node failures, and I/O congestion.
- Test on both local supercomputers (e.g., Blue Banquise-based) and public cloud/HPC clusters (e.g., CloudLab, Chameleon).

Expected Results

- A converged storage architecture that outperforms traditional file or object-only systems for DL workloads.
- A dynamic optimizer that tunes storage hyperparameters based on workload feedback in real-time.
- Detailed profiling of Mamba-based training pipelines, identifying critical I/O bottlenecks and their correlation with storage backend design.
- A set of deployment guidelines for ML system architects integrating hybrid storage in HPC environments.
- Potential open-source release of tooling for storage optimization in AI workloads.

Storage-Aware Pruning and Optimization of Large Neural Models for High-Performance Computing

Description

This study investigates the promise of storage-aware structured pruning of large neural models, like Mamba and State Space Models (SSMs), in achieving inference efficiency and load minimization on hierarchical storage systems in HPC environments. Large transformer-based models and SSM-based models generate a lot of middle data during inference, which contributes to contention and latency on active storage levels like Lustre. By pruning duplicate computation paths and model structure compression at the block or channel level, we are able to reduce data accessed from active storage. We can also decrease cold-start inference latency and increase caching efficiency across storage tiers by synchronizing pruning strategies with actual storage access patterns.

Objective

To implement and evaluate structured pruning techniques for Mamba models that reduce inference latency and storage bandwidth usage, aligning model architecture with hierarchical storage behavior in HPC settings.

Scientific Methods and Models

- Channel- and block-level pruning, focusing on compressible segments within the Mamba residual stack.
- Latency-aware sparsification, guided by profiling of inference time and memory access cost.
- Knowledge-preserving fine-tuning, using distillation to maintain accuracy post-pruning.
- Storage-influenced pruning heuristics, linking chunk usage frequency with model component retention.
- Behavioral modeling of storage traffic during inference with different sparsity regimes.

Implementation Plan

- Apply structured pruning to trained Mamba models.
- Evaluate the impact on inference throughput, memory consumption, and I/O volume from Lustre+MinIO.
- Integrate model execution profiling with storage tracing to detect hotspots and access inefficiencies.
- Tune pruning strategies based on observed storage patterns and gateway read behavior.
- Benchmark under varying workload intensities to assess robustness and generality.

Software & Tools

- Model Optimization: PyTorch, Mamba, TorchAO
- Storage: Lustre, MinIO, orchestrated via intelligent I/O gateway
- Tracing: Nsight Systems, eBPF/BCC, gateway telemetry, LDMS
- Data Sources: Sequential, temporal, and structured datasets for inference benchmarking
- Deployment: SLURM with support for pruning-aware container placement (e.g., via Apptainer)

Remote Sensing and Earth Observation Data Processing (Dr. Alena Kopanicakova)

Temporal Transfer Learning for Remote Sensing Time Series

Context

Big Earth data refers to the vast and diverse collection of geospatial information from satellite imagery, aerial photographs, LiDAR, and other remote sensing technologies. This data provides comprehensive insights into the Earth's surface, atmosphere, and environment on a global scale. Researchers and professionals can use advanced techniques such as machine learning and data analytics to extract valuable patterns, trends, and actionable information from big Earth data to address various challenges, including environmental monitoring, disaster management, urban planning, agriculture, and climate research.

Remote sensing time series analysis involves the examination of sequential satellite or aerial observations of a specific area over time. While it offers a powerful means of tracking dynamic environmental changes, it also presents considerable complexity and challenges. Handling issues like varying sensor characteristics, atmospheric interference, cloud cover, and changes in lighting conditions can introduce noise and artifacts into the data. Accurate aligning and registering time series data and selecting appropriate temporal resolutions for analysis require meticulous preprocessing and advanced algorithms. The multidimensional nature of time series data further amplifies the intricacies, demanding specialized methods to unveil meaningful trends and patterns amidst the intricate interplay of temporal and spatial dynamics.

Transfer learning is a machine learning technique where a pre-trained model developed for one task is utilized as a starting point for another related task. By leveraging the knowledge and learned features from the source task, transfer learning aims to enhance the model's performance on the target task, mainly when data for the target task is limited. Implementing transfer learning for time series analysis in remote sensing faces challenges such as aligning time series data across years with varying sensor characteristics, requiring precise preprocessing. Handling missing or irregular data demands robust interpolation. Noise and artifacts in time series must be managed to prevent negative transfers. Selecting a relevant source year is crucial to ensure the information is relevant. Adapting the model for multi-dimensional time series data and evolving patterns is a complex optimization challenge. To further improve model efficiency and accuracy, the project will dynamically explore the Incremental Data Selection (IDS) problem, selecting the most informative data samples from a continuous stream of remote sensing time series. IDS will help build more effective training datasets yearly by prioritizing samples contributing most to model generalization and minimizing redundancy. Implementing IDS techniques will enable the model to adapt more efficiently to evolving environmental conditions and sensor variability, reducing training time and computational costs.

Aside from the aforementioned problems, two additional research directions extend the thesis's scientific boundaries. The first direction concerns the development of the scientific foundation for vision, exemplified by Meta's Segment Anything Model (SAM), which has proven effective in zero- or few-shot image segmentation for natural images but has yet to demonstrate transferability to remote sensing imagery (multi-spectral channels, different resolutions, and shifted textures). Including SAM (or its remote-sensing adaptation, such as SAM-based algorithms adapted for satellite imagery) in the project as the source of spatial information is a good complement to or alternative to annual temporal transfer learning. Finally, with the increase in training dataset size due to multiple years' worth of data, the problem of scaling the

training process will be solved by implementing Incremental Data Selection (IDS), i.e., dynamically selecting the most informative samples from the stream of remote sensing time series.

Objective

This project aims to harness the power of transfer learning, foundation-model-based spatial representations, and incremental data selection (IDS) to enhance the accuracy and efficiency of remote sensing data analysis, focusing on adapting models year by year. The project seeks to improve the understanding and interpretation of evolving spatial and temporal patterns in remote sensing datasets by leveraging pre-existing knowledge from one year's data and transferring it to subsequent years. The objectives are:

- To develop transfer learning service based on Sits package and IDs sampling: Sits package is an open source software for analysis, visualization and classification of satellite image time series data (more http://www.brazildatacube.org/en/sits-satellite-image-time-series-2/). Sits has been designed to work with big satellite image data sets and fully support all land use and land cover classification workflow steps, including sampling selection, time series clustering, machine learning model training and validation, classification, and maps post-processing.
- To investigate the applicability of the SAM to remote sensing imagery, evaluating whether SAM-derived features or SAM-based segmentation (zero-shot or fine-tuned) can serve as a transferable spatial prior that complements or improves temporal transfer learning across years.
- To handle missing data: Devise and apply advanced interpolation methods to effectively handle missing or irregularly sampled data points within time series, ensuring the continuity of temporal sequences.
- Experiments and evaluation: Address the intricate optimization challenges involved in adapting models year by year. Utilize comprehensive evaluation metrics to assess the performance gains achieved through transfer learning and from SAM-based spatial transfer.

Methods

- Temporal transfer learning: Training a model trained on one year’s worth of data to future years, overcoming sensor drift and changing spatio-temporal dynamics.
- Incremental Data Selection (IDS): Online selection of the most valuable samples out of an ongoing stream of time series data, focusing on the samples that maximise generalisation without introducing redundancy, thus saving training time and computing resources.
- Missing data interpolation: Techniques for interpolating missing values in irregularly or incompletely sampled time series.
- SITS-based pipeline: Sampling, time series clustering, model training and validation, classification, and map post-processing using the SITS software.
- Spatial transfer learning via SAM: Evaluation of SAM (out of the box or fine-tuned on remote sensing data) as a source of segmentation priors or embeddings in comparison with temporal transfer learning.

Action Plan

- Study the SITS package and develop a small demo to illustrate its workflow.
- Study SAM and existing remote-sensing adaptations of it; run a small feasibility demo on a subset of Armenian imagery.

- Design a transfer learning framework based on SITS and, in parallel, a SAM-based spatial transfer baseline.
- Gather multi-year remote sensing datasets for the territory of Armenia.
- Investigate interpolation techniques suitable for missing/irregularly sampled data.
- Incorporate IDS methods to dynamically select informative samples from continuously arriving data.
- Implement fine-tuning mechanisms that gradually adapt the pre-trained model to each target year's data.
- Fine-tune or adapt SAM for the relevant land cover classes and imagery characteristics; compare its transferability against the temporal transfer learning pipeline.
- Optimize hyperparameters (learning rate, batch size, regularization).
- Experiment with different optimization algorithms for effective convergence.
- Define evaluation metrics to measure performance improvements from transfer learning and from SAM-based spatial transfer, including a direct comparison between the two paradigms.

Software & Tools

- Remote Sensing: SITS package, Segment Anything Model (SAM) / SAM variants for remote sensing
- Deep Learning: PyTorch, TensorFlow
- Data Processing: R (SITS native environment), Python, NumPy

Expected Contributions

- A transfer learning framework combining SITS-based temporal adaptation with IDS sampling for multi-year remote sensing analysis.
- Robust interpolation methods for handling missing/irregular time series data.
- An empirical evaluation of SAM's transferability to remote sensing imagery, and its comparative value against year-over-year temporal transfer learning.
- A demonstrated, evaluated pipeline applied to real-world Armenian remote sensing datasets.

Enhancing Land Cover Classification: Comparative Analysis of Diverse Classification Methods and Ensemble Learning Application

Context

Land cover classification is the process of identifying and classifying different types of land cover from remotely sensed data, such as satellite images. Land cover is the physical material that covers the Earth's surface, such as forests, grasslands, water bodies, and urban areas. Land cover classification is a valuable tool for a variety of applications, including environmental monitoring, natural resource management, disaster management, etc. There are many different methods for conducting land cover classification, with a selection of prevalent methods including unsupervised and supervised classifications. The unsupervised classification method classifies pixels based on their spectral similarity. No prior knowledge of the land cover classes is required. The supervised classification method classifies pixels based on their spectral similarity and their association with known land cover classes. This method requires a training dataset of labeled pixels.

Ensemble learning is a machine learning technique that combines the predictions of multiple models to improve overall performance. The idea is that combining the predictions of several different models reduces the variance and improves the accuracy of the predictions.

Most classical land cover classification pipelines treat each image (or image composite) as a static snapshot, which discards the temporal information available in dense satellite time series — often the strongest signal for distinguishing land cover types with similar spectral signatures but different seasonal behavior (e.g., crops vs. natural vegetation). This motivates extending the comparison beyond static-image classifiers to satellite image time-series (SITS)-based classification, using both traditional ML and deep learning models, positioning this project as a natural complement to Topic 1's focus on temporal transfer learning while remaining self-contained.

Objective

- Comprehensively investigate a diverse set of base classifiers, such as Decision Trees, Random Forests, Support Vector Machines, and Neural Networks, among others, within the context of land cover classification.
- Compare the results of the methods and identify the most optimal model for effectively addressing the given problem.
- Leverage Ensemble learning techniques to enhance the precision of land cover classification using earth observation data.
- Extend the comparison to SITS-based classification, evaluating both traditional ML models (e.g., Random Forest, SVM on time-series features) and deep learning architectures suited to sequential data (e.g., 1D-CNN, LSTM/GRU, Transformer-based time-series classifiers), to determine whether temporal information meaningfully improves classification accuracy over single-date imagery.
- Assess whether ensemble methods combining static-image and time-series-based classifiers outperform either approach alone.

Methods

- Static classification: Training and evaluating Decision Trees, Random Forests, SVMs, and Neural Networks on single-date or composite imagery.
- Ensemble learning: Combining multiple base classifiers (e.g., via bagging, boosting, or stacking) to reduce variance and improve robustness.
- SITS-based classification: Using dense time-series inputs per pixel/parcel rather than single-date composites, applying both feature-engineered ML models and deep sequence models (1D-CNN, RNN/LSTM, Transformer) to capture seasonal/temporal land cover signatures.
- Comparative evaluation: A unified benchmark comparing static classifiers, SITS-based classifiers, and their ensembles on the same Armenian dataset, using consistent accuracy, F1, and confusion-matrix-based metrics.

Action Plan

- Study state-of-the-art solutions for land cover classification, including both static-image and SITS-based approaches.
- Problem analysis and dataset preparation for the territory of Armenia.
- Selection of diverse classifiers and training (static-image models).
- Construction of a SITS-based training pipeline (potentially reusing the SITS package from Topic 1) and training of time-series ML/DL classifiers.
- Model evaluation and comparison across static and time-series-based approaches.

- Ensemble model design, implementation, and training — including hybrid ensembles combining static and SITS-based classifiers.

Software

- Machine Learning: scikit-learn, Random Forest, SVM implementations
- Deep Learning: PyTorch or TensorFlow (for SITS-based sequence models)
- Remote Sensing: SITS package (for time-series data handling and preprocessing)
- Ensemble Frameworks: e.g., XGBoost, LightGBM, or custom stacking implementations

Expected Contributions

- A comprehensive comparative analysis of classical classifiers for land cover classification in the Armenian context.
- An ensemble learning model demonstrating improved classification precision over individual base classifiers.
- A comparative evaluation of static-image versus satellite image time-series (SITS)-based classification, quantifying the benefit of temporal information for land cover discrimination.
- A hybrid ensemble framework combining static and time-series-based classifiers, assessed against single-paradigm baselines.