Enhancing GNUstep with Machine Learning Support

Group 10
Cross-Platform ML Integration Layer Proposal

Group 10

https://youtu.be/baR37Gsg1e4

Group 10

Chuka Uwefoh – 22khl2@queensu.ca - Team Lead

Lore Chiepe – 21lpc2@queensu.ca - Presenter

Henry Chen – 19hc38@queensu.ca

James Wang – 22jw16@queensu.ca - Presenter

Ryan Jokhu – 21raaj@queensu.ca

Zachary Stephens – 20zs46@queensu.ca

Previews

1. Motivation for New Feature
2. Proposed Enhancement
3. Benefits of libs-ml
4. Implementation Approach 1: Direct Integration
5. Implementation Approach 2: Plugin Based
6. SAAM Analysis
7. Effects on Architecture
8. Use Case 1: Real-Time Image Classification
9. Use Case 2: Predictive Text Input
10. Risks, Concurrency, Future
11. Plans for Testing
12. Lessons Learned
13. Conclusion

Motivation for New Feature

Current Challenges:

GNUstep lacks built-in machine learning (ML) capabilities

Why it matters:

Growing demand of cross-platform desktop ML apps

No mainstream Objective-C support for ML

Goal: Enhance GNUstep with a native ML layer to improve usability

Proposed Enhancement

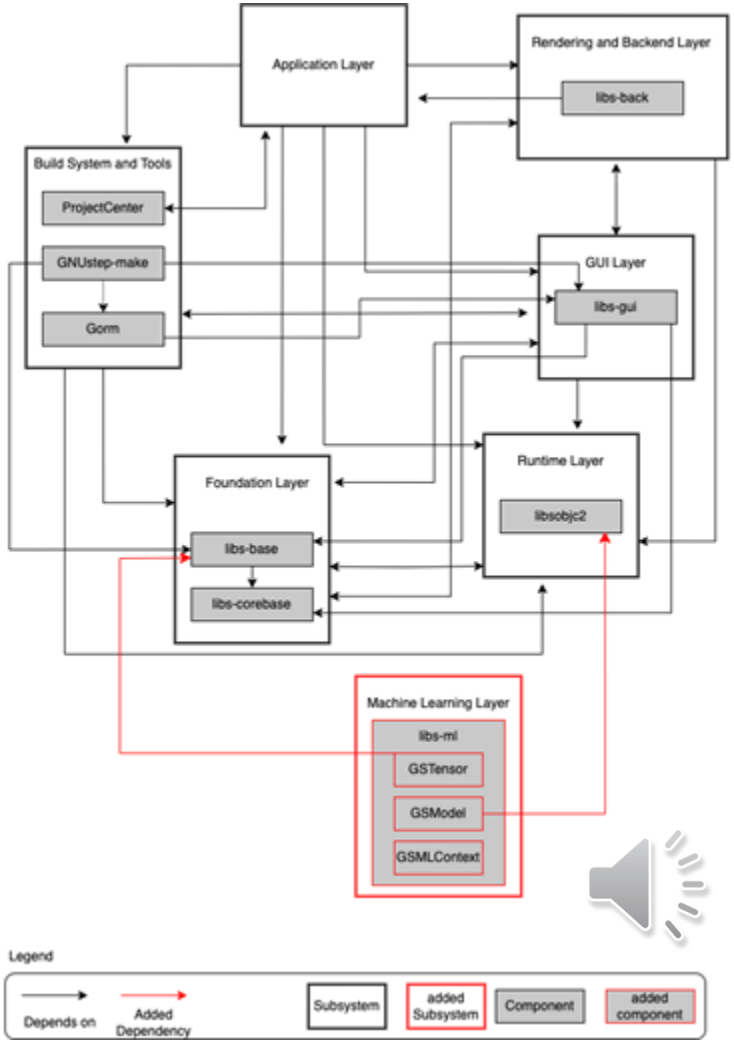
Name: libs-ml – A new ML subsystem

Key Components:

GSModel: Unified ML model interface.

GSTensor: Optimized numerical data structures.

GSMLContext: Hardware-aware inference management.



Benefits of libs-ml

Key Advantages:

Cross-Platform Compatibility - Addresses compatibility needs

On-Device Privacy - No dependency on cloud service for data safety

Hardware Acceleration - Utilizes GPU for fast processing

Ease of Use - Simplified ML commands enhance productivity.

Design Goals:

Modular, plug-in based

Optional runtime integration

Hardware abstraction

Minimal legacy disruption

Implementation Approach 1: Direct Integration

Integrate existing GNUstep structure to use new ML features with slight alterations

- Libs-ml as a core subsystem providing the functionality of the model
- Key features:
  - GSModel: Objective-C wrapper for ML model
  - GSTensor: Data structure for numerical operations
  - GSMLContext: Manages hardware acceleration

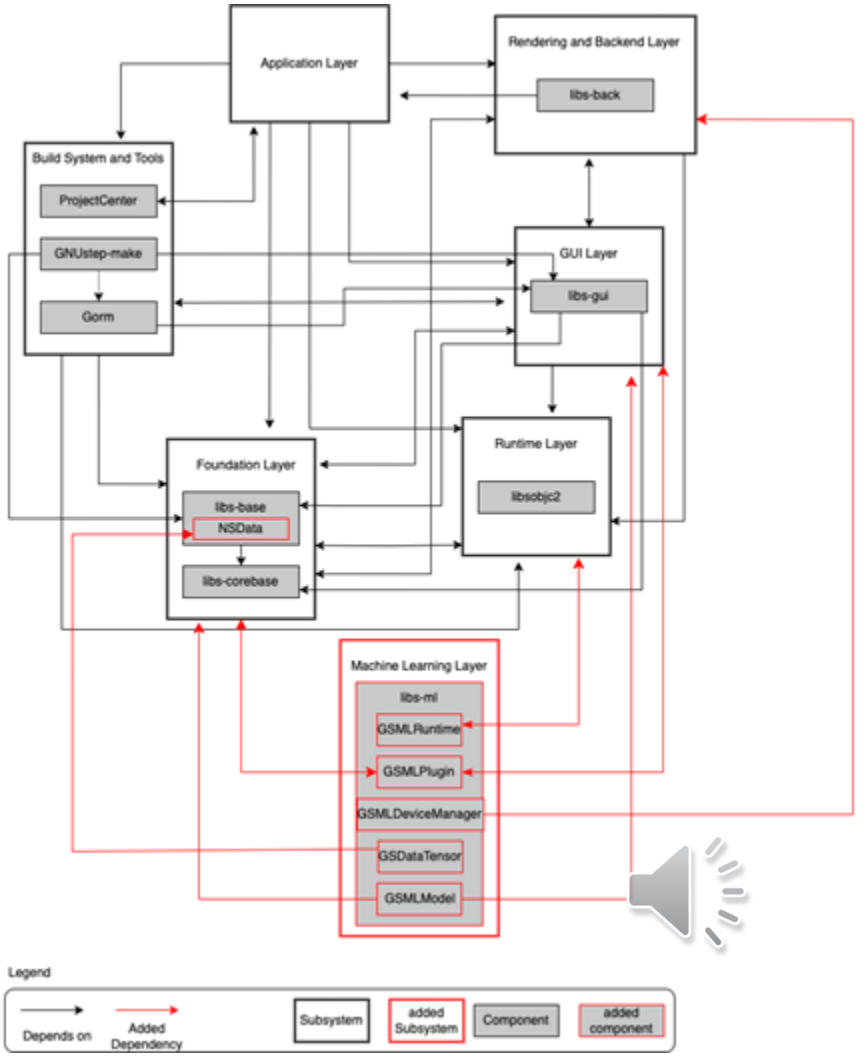
- Modified:
  - libs-base: Add tensor support to NSData
  - libs-gui: Optional ML-powered UI enhancements
  - GNUstep-make: Add ML framework dependencies

Implementation Approach 2: Plugin-Based Execution

Create a plugin based execution for backend switching with efficient tensor operation and hardware-aware inference.

- Libs-ml: main library providing ML Functionality
- GSMLRuntime: Unified runtime interface for executing ML models
- GSMLPlugin: Abstract plugin interface for different ML frameworks
- GSMLDeviceManager: Handles CPU/GPU/NPU inference selection
- GSDataTensor: Independent tensor structure optimized for ML data
- GSMLModel: Base class for all ML models

- Modified:
  - libs-base: Interfaces with GSDataTensor numerical operations
  - libs-gui: Optional ML-powered UI enhancements



SAAM Analysis: Stakeholders and Concerns

Major Stakeholders:

- GNUstep Developers: Seek seamless integration with minimal disruption.
- Application Developers: Desire easy access to ML capabilities without platform-specific complexities.
- End Users: Expect efficient, privacy-preserving applications powered by on-device ML.
- System Maintainers: Prefer modular solutions that are easy to debug.
- Hardware Vendors: Benefit from optimized performance utilizing their hardware.

Non-Functional Requirements:

- Backward Compatibility: Essential to avoid disruptions.
- Simplicity: Interfaces must be intuitive.
- Performance: Focus on low latency and resource optimization.
- Maintainability: New components must integrate smoothly.

Effects on Architecture

Architecture Style: The integration maintains a layered architecture approach, allowing modular enhancements without significant disruption

Architectural Impact:

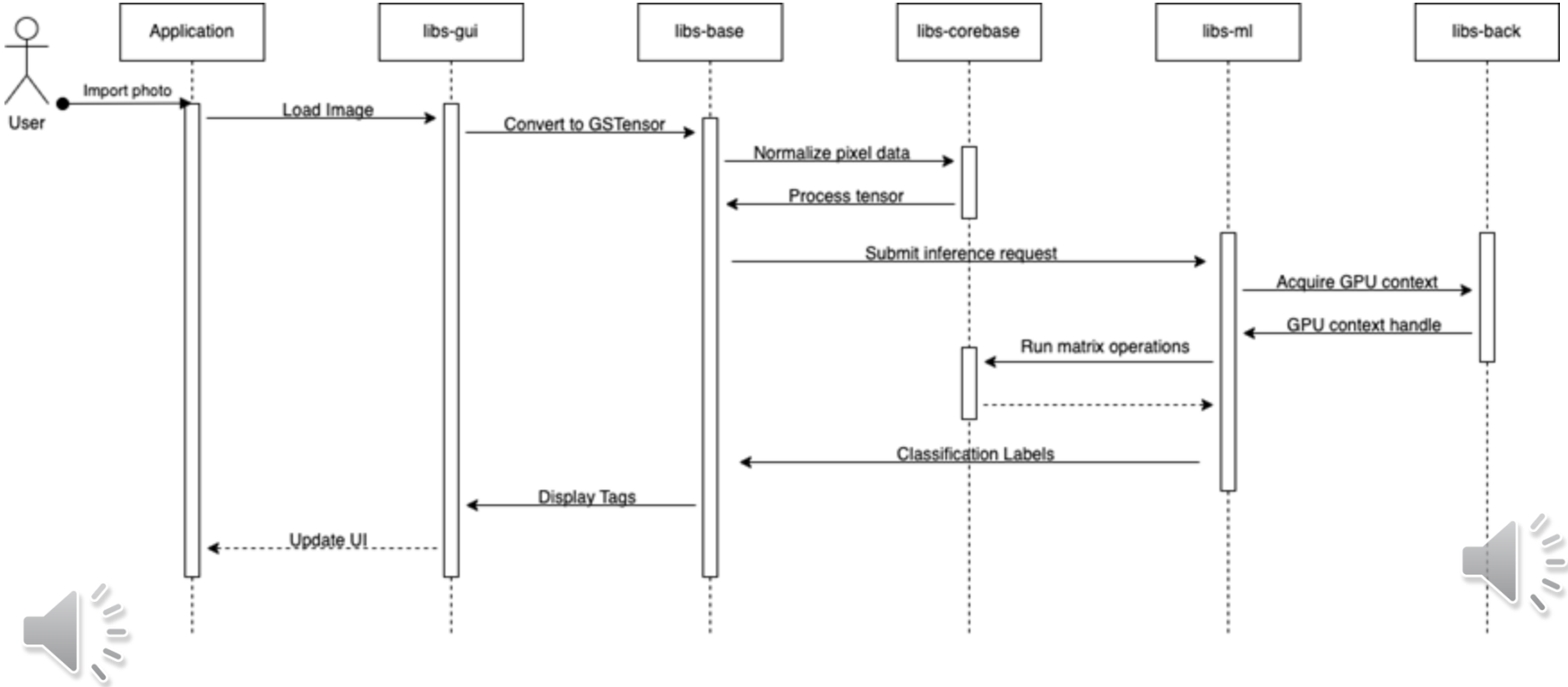
Maintainability: By isolating the libs-ml subsystem from core components, the overall system easy to maintain

Evolvability: The structure allows for easy addition of new features, plugins, etc...

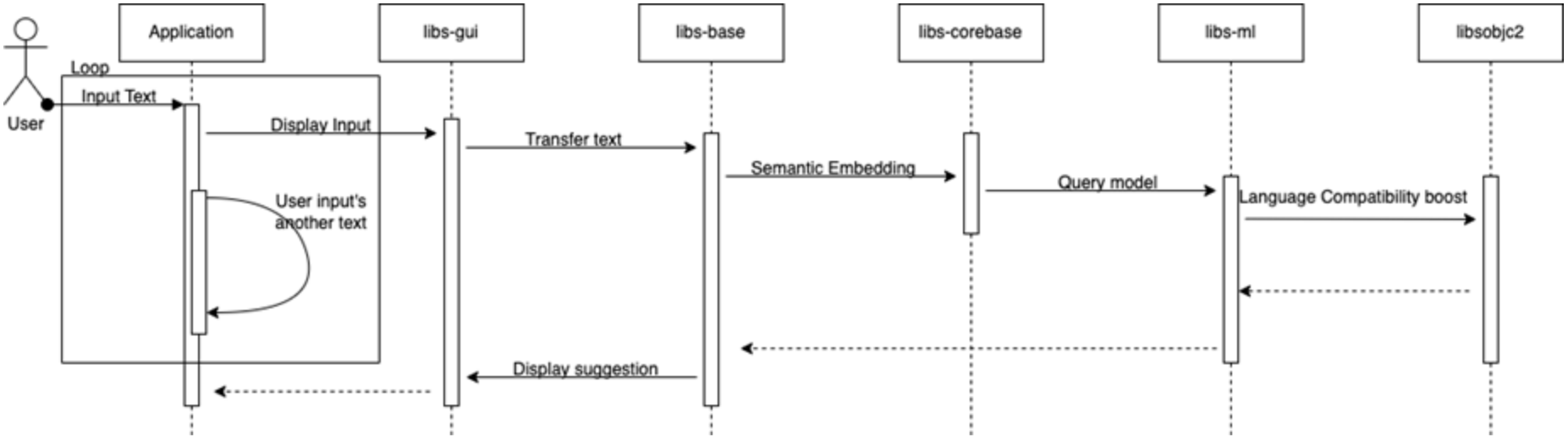
Performance Efficiency: The new ML features leverage hardware acceleration

Testing and Integration: Simplified testing for individual components due to clear boundaries

Use Case 1: Real-Time Image Classification



Use Case 2: Predictive Text Input



Risks, Concurrency, and Future Directions

Risks:

- Increased system complexity
- Potential performance bottlenecks (GPU overuse)
- Security concerns
- Risk of architectural bloat

Concurrency:

- Potential race conditions or memory leaks
- Need for thread-safe implementations

Future Directions:

- Further optimization
- Expanding support for more ML frameworks

Plan for Testing

Testing Strategy:

- Component Tests for GSModel, GSTensor, GSMLContext.
- Cross-Platform Testing to ensure consistency.
- Performance Testing with benchmarks for GPU and CPU inference.
- Load and Stress Testing to identify race conditions.

Lessons Learned

- Learned to integrate ML features while maintaining GNUstep's lightweight design.
- Abstracting low-level details simplifies development and enhances usability.
- Combining insights from various fields leads to more effective modernization of legacy frameworks.

Conclusion

Integration Benefits:

- **Aligns with GNUstep's goals of maintaining simplicity and modularity.**
- **Enhances usability for developers while ensuring performance and privacy.**

Future of GNUstep:

- **The libs-ml enhancement prepares GNUstep for AI-driven applications without compromising foundational integrity.**