The ML 1 and ML 2 Team Leads are responsible for guiding their respective teams through all phases of machine learning model development. They provide technical leadership and mentorship to ensure team members have the necessary resources and support to perform effectively. Task delegation is done based on individual strengths, and the leads manage timelines, address conflicts, and maintain overall team efficiency. In terms of project planning and execution, the team leads prioritize tasks, track progress against key milestones, and proactively identify risks to implement solutions. They ensure the quality of technical outputs remains high by promoting best practices in coding, documentation, and version control. Innovation is also a key focus, encouraging teams to explore new algorithms and techniques, particularly in EEG signal classification.

Strong communication skills are essential for these roles. The leads ensure coordination between the ML 1 and ML 2 teams, maintaining consistency and integration across their efforts. They serve as the primary point of contact for stakeholders, providing regular updates on progress and challenges. Additionally, detailed documentation of processes, decisions, and outcomes is maintained to ensure transparency and accountability. The team leads are also responsible for producing project documentation, including reports on algorithm exploration, benchmark studies, and model performance metrics. They supervise the development and fine-tuning of machine learning model architectures tailored to EEG data, overseeing the evaluation process to guide model refinement and selection. Final deliverables include comparative analyses, experimental records, and the justified selection of models for deployment.

To keep the teams motivated and aligned with project goals, the team leads establish clear, measurable objectives (SMART goals) and hold regular meetings to track progress and resolve issues. They foster a collaborative atmosphere by promoting knowledge sharing and cross-team efforts, while recognizing and celebrating team successes to boost morale. By offering opportunities for professional growth and encouraging innovation, they help maintain high motivation and job satisfaction. Efficient management through agile methodologies and the use of task management tools ensures flexibility and accountability. Additionally, flexible scheduling and wellness initiatives support work-life balance and overall team well-being. Through their leadership and management, the ML 1 and ML 2 Team Leads ensure their teams contribute effectively to the project’s success.

Successful project:

Example Project : EEG controlled RC car

GUI creation

- Teams involved: Interface Design Team, Real Time Signal Acquisition, Signals and ML Integration
- Design a GUI that instructs the user to imagine moving a RC car
    - Implement clear instructions within the GUI to guide users
        - Display concise, easy-to-understand instructions guiding the user on how to perform the imagined movements.
        - Use bullet points or numbered steps for clarity.

    - Arrows will prompt the user which direction they should imagine (Left, Right, Up, Down)
        - Color-code the arrows (e.g., Left - Blue, Right - Green) for quick recognition.
    - Incorporate icons or images of the RC car moving in the prompted direction.
        - Use simple animations to demonstrate the movement.
    - Top right corner should let the user know how many more samples they need to collect
    - Progress bar should inform the user how much time is left in the sample
    - Provide an option to view real-time EEG signals in a graph format.
        - Implement real-time graphical display of EEG signals for immediate assessment.
    - Allow users to select which channels to display and adjust the scaling.
- Session Summary
    - At the end of a session, display a summary with total samples collected, success rate, and signal quality statistics.
    - Include visual charts or graphs for better understanding.

- Signals team should implement functionality to collect data in real time and store the data in a chosen database after each sample
    - Implement buffering to handle any latency or data spikes.
    - There should be documentation explaining how the data is being collected and how the data is to be labeled
        - Automatically label collected data with the corresponding imagined movement (Left, Right, Up, Down).
        - Include timestamps and session IDs for reference.
    - Manage privacy, confidentiality, and consent for any subjects the data comes from

- 3D Brain Maps:
    - Offer advanced users the option to view EEG activity on a 3D brain model.
    - Use color gradients to represent different levels of activity.
- Customizable Dashboards:
    - Let users create personalized dashboards showing the data and metrics they care about.
    - Include widgets for quick access to specific functions.

Data collection

- Teams Involved: Real-Time Signal Acquisition Team, Signals and ML Integration Team
- Impedance Checks
    - Perform impedance measurements to confirm good contact between electrodes and the scalp.

- Baseline Recording Verification
    - Conduct initial baseline recordings to verify signal clarity and stability.
    - Check for any artifacts or noise in the baseline signal.

- EEG Data Capture:
    - Set up and calibrate EEG devices for accurate signal acquisition.
    - Monitor data quality in real-time to identify and mitigate artifacts or noise.
    - User Preparation Guidelines
        - Provide clear instructions to users on pre-session preparations (e.g., wash hair without conditioner, avoid caffeine).
        - Supply a checklist to confirm readiness before starting the session.

- Data Synchronization:
        - Align EEG data streams with GUI prompts to ensure accurate labeling.
    - Quality Assurance:
        - Validate the completeness and integrity of collected data.
        - Implement preliminary checks to identify and flag any anomalies or inconsistencies in the data.

- Artifact Detection and Mitigation
    - Employ algorithms to automatically detect common artifacts like eye blinks, muscle movements, and line noise.
        - Provide real-time notifications to the operator when artifacts are detected.
        - Use adaptive filtering techniques to minimize the impact of artifacts on the collected data.

- Signal Integrity Alerts

        - Set thresholds for acceptable signal levels and notify the operator when signals fall outside these ranges.
            - Pause data collection automatically if severe artifacts or signal loss is detected.

- Event Markers in EEG Streams
    - Insert digital markers directly into the EEG data stream at the onset and offset of each GUI prompt.
    - Use unique codes for different imagined movements to facilitate automatic labeling.
- Validation of Synchronization Accuracy
    - Perform test runs comparing expected event times with recorded EEG markers.
    - Calculate synchronization error margins and adjust protocols accordingly.

- Error Handling
    - Implement checks to identify mislabeled or unlabeled data segments.
    - Provide logs detailing any discrepancies found during the labeling process.

- Labeling Algorithms
    - Develop scripts to automatically associate EEG data segments with the correct imagined movement labels based on event markers.
    - Include functionality to handle overlapping events or missed markers.

- Metadata Inclusion
    - Attach additional metadata such as participant ID, session number, and timestamp to each data file.
        - Record environmental conditions (e.g., room temperature, noise levels) if relevant.

- Documentation and Reproducibility
    - Create detailed documentation outlining synchronization steps, tools used, and troubleshooting tips.
        - Ensure protocols are reproducible by other team members or for future projects.
- Channel Data Integrity
    - Complete Channel Recording
    - Confirm that data from all EEG channels have been successfully recorded without any loss.
    - Cross-verify the recorded channels against the expected list to ensure none are missing.
- Signal Continuity
    - Scan for any interruptions or discontinuities in the recorded signals.
    - Use automated scripts to detect gaps or abrupt changes in the data streams.
- Session Duration Confirmation

    - Time Alignment
        - Ensure that the total duration of the recorded data matches the expected session length.
        - Compare timestamps from EEG data with the GUI prompts to verify synchronization.
    - Event Marker Verification
        - Check that all expected event markers (e.g., movement prompts) are present and correctly time-stamped.
        - Validate the consistency of event markers throughout the session.

- Post-Session Data Checks
    - Verify that data from all EEG channels has been recorded successfully.
    - Ensure that the duration of recorded data matches the expected session length.
    - Corrupted File Detection
        - Use checksums or hash functions to detect corrupted files immediately after data capture.
            - Implement automatic data recovery procedures where possible.
- Data Integrity Checks
    - Checksum Generation
        - Generate checksums (e.g., SHA-256) for each data file immediately after recording.
    - Compare generated checksums with originals to detect any file corruption.
- Automatic Data Recovery Procedures
    - Redundant Storage
        - Implement redundant data saving mechanisms, such as RAID configurations or cloud backups.
    - Schedule regular backups during and after data collection sessions.

Data processing

- Teams Involved: Signals and ML Integration Team
- Anomaly Detection and Flagging
    - Statistical Analysis
        - Descriptive Statistics Computation
            - Compute Key Metrics
                - Calculate mean, median, variance, standard deviation, skewness, and kurtosis for each EEG channel.
                - Identify channels with abnormal statistics indicating potential issues.
            - Outlier Detection
                - Use z-scores or interquartile ranges to detect outliers in the data.

                - Flag data points that deviate significantly from normal ranges.
            - Time-Frequency Analysis
                - Spectral Analysis
                    - Perform power spectral density (PSD) analysis to examine frequency content.
                    - Identify unusual frequency components that may indicate artifacts.
    - Quality Metrics Reporting
        - Metric Calculation
            - Signal-to-Noise Ratio (SNR)
                - Calculate SNR for each channel to assess signal quality.
        - Artifact Prevalence
            - Quantify the percentage of data affected by artifacts.
        - Channel Reliability
            - Evaluate the reliability of each channel based on signal consistency.
        - Develop scripts to generate reports summarizing quality metrics after each session.
    - Feedback Loop to Acquisition Team
        - Communicate any identified issues back to the Real-Time Signal Acquisition Team promptly.
        - Collaborate on solutions to recurring problems, updating protocols as necessary.
    - Documentation
        - Maintain a log of identified issues, actions taken, and outcomes.
        - Share best practices and lessons learned with the team.
- Preprocessing:
    - Band-Pass Filtering
        - - Define Frequency Ranges
            - - Apply band-pass filters (e.g., 0.5 Hz to 40 Hz) to retain frequencies relevant to EEG signals.
    - Filter Design
        - Use Butterworth, Chebyshev, or elliptic filters as appropriate.
        - Ensure filters are zero-phase to prevent signal distortion.
    -
    - Notch Filtering
        - Implement notch filters at 50 Hz or 60 Hz to eliminate power line interference.
    - Consider adaptive filters for varying noise conditions.
        - High-Pass and Low-Pass Filtering
        - Use high-pass filters (>0.5 Hz) to remove slow drifts.
    - High-Frequency Noise Reduction

        - Apply low-pass filters (<40 Hz) to eliminate muscle artifacts and high-frequency noise.
    - Artifact Removal
        - Independent Component Analysis (ICA)
    - Decomposition
        - Separate EEG signals into independent components.
    - Artifact Identification
        - Identify components representing artifacts like eye blinks or muscle activity.
            - Use algorithms like FastICA or Infomax ICA.
    - Component Exclusion
        - Remove identified artifact components and reconstruct clean EEG signals.
    - Regression Techniques
        - EMG Regression
            - electromyogram (EMG) recordings to model and subtract muscle artifacts.
    - Adaptive Filtering
        - Implement adaptive filters that adjust in real-time to changing artifact patterns.
    - Wavelet Decomposition
        - Multi-Resolution Analysis
            - Decompose signals into wavelets to isolate and remove artifacts at specific scales.
        - Automated Artifact Detection
    - Algorithm Development
        - Create algorithms to automatically detect artifacts based on amplitude thresholds, frequency content, or spatial patterns.
        - Machine Learning Approaches
            - Train models to recognize and remove artifacts from EEG data.
    - Segmentation:
        - Data Segmentation:
            - Divide continuous EEG data into epochs corresponding to each imagined movement prompted by the GUI.
    - Feature Extraction
        - Feature Identification
                - Extract features like mean amplitude, peak latency, zero-crossing rate, and variance.
                - Compute power spectral density (PSD) using Fourier Transform.
                - Analyze band powers (delta, theta, alpha, beta, gamma bands).
            - Time-Frequency Analysis

            - Use Short-Time Fourier Transform (STFT) or wavelet transforms to capture temporal changes in frequency content.
        - Dimensionality Reduction
            - Principal Component Analysis (PCA)
            - Linear Discriminant Analysis (LDA)
            - t-Distributed Stochastic Neighbor Embedding (t-SNE)
    - Feature Selection
        - Use t-tests, ANOVA, or mutual information to select features that significantly differentiate between classes.
        - Apply Lasso or Ridge regression to select features and prevent overfitting.
        - Normalization and Scaling
            - Z-Score Normalization
                - Standardize features to have zero mean and unit variance.
- Data Preparation for ML Models
    - Data Splitting
        - Training, Validation, Test Sets
            - Split data into separate sets to train models and evaluate performance objectively.
            - Ensure no data leakage between sets.
            - Add controlled noise to existing samples to improve model robustness
        - Handling Class Imbalance
            - Use oversampling (e.g., SMOTE) or undersampling to balance classes.
            - Adjust algorithm parameters to account for imbalanced datasets.

Algorithm Exploration

Teams involved: ML Team 1, ML Team 2

- Literature Review and Research
    - State-of-the-Art Analysis
        - Conduct a comprehensive review of current EEG signal classification techniques.
        - Identify algorithms commonly used for EEG data, such as Support Vector Machines (SVM), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory networks (LSTM).
        - Evaluate the applicability of emerging models like Graph Neural Networks (GNN) for spatial EEG data representation.
    - Benchmark Studies
        - Analyze benchmark datasets and competitions (e.g., BCI Competition datasets) to understand successful approaches.

- Algorithm Shortlisting
    - Criteria Definition
        - Define criteria for model selection, including accuracy, computational efficiency, robustness to noise, and real-time processing capability.
    - Preliminary Testing
        - Implement prototype versions of shortlisted algorithms.
        - Perform initial tests on a subset of the data to assess feasibility.

Dataset Preparation for Model Evaluation

- Data Augmentation
    - Techniques Implementation
        - Apply methods like signal scaling, time shifting, noise injection, and spectral manipulation to augment training data.
    - Class Balance Maintenance
        - Ensure that augmented data maintains class balance to prevent bias.
- Cross-Validation Strategy
    - Method Selection
        - Choose appropriate cross-validation techniques (e.g., k-fold, leave-one-subject-out) to evaluate model generalization.
    - Data Segregation
        - Ensure that data splits prevent leakage, especially in subject-dependent vs. subject-independent scenarios.

Performance Evaluation

- Metric Computation
    - Quantitative Metrics
        - Calculate accuracy, precision, recall, F1-score, confusion matrices, and area under the ROC curve (AUC-ROC).
    - Computational Metrics
        - Measure training time, inference time, and memory usage.
    - Statistical Analysis
        - Use statistical significance testing (e.g., t-tests) to compare model performances.

Model Benchmarking

- Comparative Analysis
    - Performance Comparison
        - Rank models based on performance metrics and computational efficiency.
    - Resource Utilization
        - Assess each model's compatibility with the hardware constraints of the deployment environment (e.g., processing power, memory).

Documentation

- Selection Rationale
    - Decision Logs
        - Document the reasoning behind selecting or rejecting each model.
    - Experiment Records
        - Maintain detailed records of experiments, including parameter settings and results.

Model Design and Training

Teams Involved: ML 1 Team, ML 2 Team

Model Architecture Design

- Custom Architecture Development
    - Model Customization
        - Design neural network architectures tailored to EEG data characteristics.
        - Incorporate spatial and temporal features of EEG signals.
    - Layer Configuration
        - Determine the number of layers, neurons, activation functions (e.g., ReLU, sigmoid), and dropout rates.
- Pre-trained Models Utilization
    - Transfer Learning
        - Explore the use of pre-trained models where applicable.
        - Fine-tune models on the specific EEG dataset to improve performance.

Model Training

- Training Protocols
    - Hyperparameter Initialization
        - Set initial values for learning rate, batch size, and optimization algorithms (e.g., Adam, SGD).
    - Data Pipeline
        - Establish efficient data loading and preprocessing pipelines.
        - Implement real-time data augmentation during training.
- Regularization Techniques
    - Overfitting Prevention
        - Apply techniques like L1/L2 regularization, early stopping, and batch normalization.
    - Validation Checks
        - Monitor training and validation losses to detect overfitting.

Hyperparameter Tuning

- Optimization Strategies
    - Search Methods
        - Use grid search, random search, or Bayesian optimization for hyperparameter tuning.
    - Parameter Ranges
        - Define ranges for key hyperparameters like learning rate, number of layers, and dropout probability.

Validation

- Model Evaluation
    - Cross-Validation
        - Perform k-fold cross-validation to assess model stability.
    - Generalization Testing
        - Test models on unseen data to evaluate generalizability.
- Error Analysis
    - Misclassification Investigation
        - Analyze incorrectly classified samples to identify patterns or data issues.
    - Model Refinement
        - Adjust model architecture or training data based on findings from error analysis.

Model Optimization and Interpretation

Teams Involved: ML 3 Team

Model Optimization

- Performance Enhancement
    - Quantization
        - Reduce model size and increase inference speed by quantizing weights (e.g., 16-bit or 8-bit precision).
    - Pruning
        - Remove redundant neurons or layers that do not significantly impact performance.
    - Hardware Acceleration
        - Leverage GPU acceleration or specialized hardware (e.g., FPGA, TPU) for faster computation.
- Resource Management
    - Memory Optimization
        - Optimize memory usage to ensure the model can run on embedded systems with limited resources.
    - Efficiency Trade-offs
        - Balance the trade-off between model complexity and computational efficiency.

Model Interpretation

- Explainability Techniques
    - Feature Importance
        - Use methods like permutation importance or saliency maps to determine which features contribute most to model decisions.
    - Class Activation Mapping
        - Implement Grad-CAM or similar techniques to visualize regions of input that influence the model's predictions.
- Reliability Assessment
    - Consistency Checks
        - Test the model's output consistency across multiple runs with the same input.
    - Robustness Testing
        - Evaluate the model's resilience to noise, artifacts, and other real-world signal variations.

Optimization Documentation

- Process Documentation
    - Optimization Steps
        - Record all optimization techniques applied and their effects on model performance.
    - Interpretation Insights
        - Document findings from model interpretation to guide future development.

Model Deployment

Teams Involved: Signals and ML Integration Team, Interface Design Team, ML 3 Team

Integration

- System Integration
    - API Development
        - Create RESTful APIs or use sockets for real-time communication between the EEG processing unit and the RC car control system.
    - Middleware Implementation
        - Develop middleware to handle data preprocessing, model inference, and command transmission.
- User Feedback Mechanisms
    - GUI Integration
        - Update the GUI to display real-time predictions and system status.
        - Provide visual cues (e.g., color changes, icons) to indicate the confidence level of predictions.
    - Alerts and Notifications

        - Implement alerts for signal loss, artifacts, or when the user's focus decreases.

Communication Protocols

- Data Transfer
    - Protocol Selection
        - Choose appropriate communication protocols (e.g., TCP/IP, UDP) based on latency and reliability requirements.
    - Data Encoding
        - Standardize data formats (e.g., JSON, Protocol Buffers) for efficient serialization and deserialization.

Testing & Validation

- System Testing
    - Integration Testing
        - Test the complete system to ensure all components work seamlessly together.
    - Latency Measurement
        - Measure and optimize end-to-end latency from EEG signal acquisition to RC car response.
- User Testing
    - Pilot Studies
        - Conduct sessions with users to test system usability and gather feedback.
    - Performance Metrics
        - Record metrics like response accuracy, user satisfaction, and ease of use.

Deployment

- Environment Setup
    - Hardware Configuration
        - Ensure the deployment environment meets all hardware requirements.
    - Software Installation
        - Install necessary software dependencies, drivers, and libraries.
- Scalability Planning
    - Future Expansion
        - Design the system to accommodate additional features or increased user load in the future.

Monitoring

- Performance Monitoring
    - Logging Systems
        - Implement logging for system events, errors, and user interactions.

        - Real-Time Monitoring
            - Use dashboards to monitor system health, resource usage, and performance metrics.
- Issue Resolution
    - Alert Systems
        - Set up alerts for critical system failures or performance drops.
    - Maintenance Protocols
        - Establish procedures for regular system checks and updates.

Documentation

- Deployment Guides
    - Installation Manuals
        - Provide step-by-step instructions for setting up and deploying the system.
    - Troubleshooting
        - Include a comprehensive troubleshooting section for common issues.
- System Architecture
    - Diagrams and Flowcharts
        - Create visual representations of the system architecture, data flow, and component interactions.
- User Manuals
    - Operator Instructions
        - Develop manuals for system operators detailing how to run and maintain the system.
    - End-User Guides
        - Provide guides for users interacting with the RC car, including best practices for optimal performance.

Model optimization and interpretation

- Teams Involved: ML 3 Team
- Model Optimization:
    - Performance Enhancement:
        - Fine-tune models to improve inference speed and reduce computational load, ensuring real-time applicability.
    - Resource Management:
        - Optimize models for deployment on embedded systems or edge devices controlling the RC car.
- Model Interpretation:
    - Explainability:
        - Utilize techniques like SHAP (SHapley Additive exPlanations) or LIME (Local Interpretable Model-agnostic Explanations) to interpret model decisions.
    - Reliability Assessment:

        - Analyze model behavior to ensure consistent and reliable performance across different users and conditions.
- Optimization Documentation:
    - Reporting:
        - Document the optimization processes and interpretability findings to provide insights for future improvements and maintenance.

Model deployment

- Teams Involved: Signals and ML Integration Team, Interface Design Team, ML 3 Team
- Integration:
    - System Integration:
        - Integrate the optimized model into the real-time system that processes EEG data and controls the RC car.
    - User Feedback:
        - Implement real-time feedback mechanisms within the GUI to inform users about the RC car's status and actions based on their EEG inputs.
- Communication Protocols:
    - Ensure seamless communication between EEG data acquisition, model inference, and the RC car's control system.
- Testing & Validation:
    - System Testing:
        - Conduct comprehensive tests to evaluate the system's latency, accuracy, and responsiveness.
        - Perform user testing to gather feedback and make necessary adjustments.
- Deployment:
    - Launch Preparation:
        - Prepare deployment environments, ensuring all components are compatible and functional.
    - Monitoring:
        - Implement monitoring tools to track system performance post-deployment and identify any issues promptly.
- Documentation:
    - Deployment Guides:
        - Create detailed documentation for deployment procedures, system architecture, and troubleshooting guidelines.

Collaboration and Communication Protocols

Inter-Team Coordination

- Regular Meetings and Updates
    - Schedule weekly meetings between teams to discuss progress, challenges, and upcoming tasks.
    - Share meeting minutes and action items with all team members.

- Shared Documentation Platforms
    - Use collaborative tools (e.g., Confluence, Google Docs) for documentation and knowledge sharing.
    - Maintain version-controlled repositories for code and documentation.
- Issue Tracking Systems
    - Implement issue tracking software (e.g., Jira) to manage tasks and bugs.
    - Assign responsibilities and set deadlines for issue resolution.

ML 1
ML 2
ML 3
interface design (including webdev)
VR design
signals and ML integration
Real time signal acquisition