Lecture 14 - Contents

An overview of the main sections in this lecture.

This outline is for guidance. Navigate the slides with the left/right arrow keys.

Lecture 14:

Continuous Learning: Evolving Medical AI

Ho-min Park

homin.park@ghent.ac.kr

powersimmani@gmail.com

Continual Learning Overview

🔄 Continual Learning

The ability to continuously learn new data and knowledge while maintaining previously learned content

🏥 Need for Medical AI

Medical knowledge and clinical protocols continuously change, and AI models must adapt accordingly

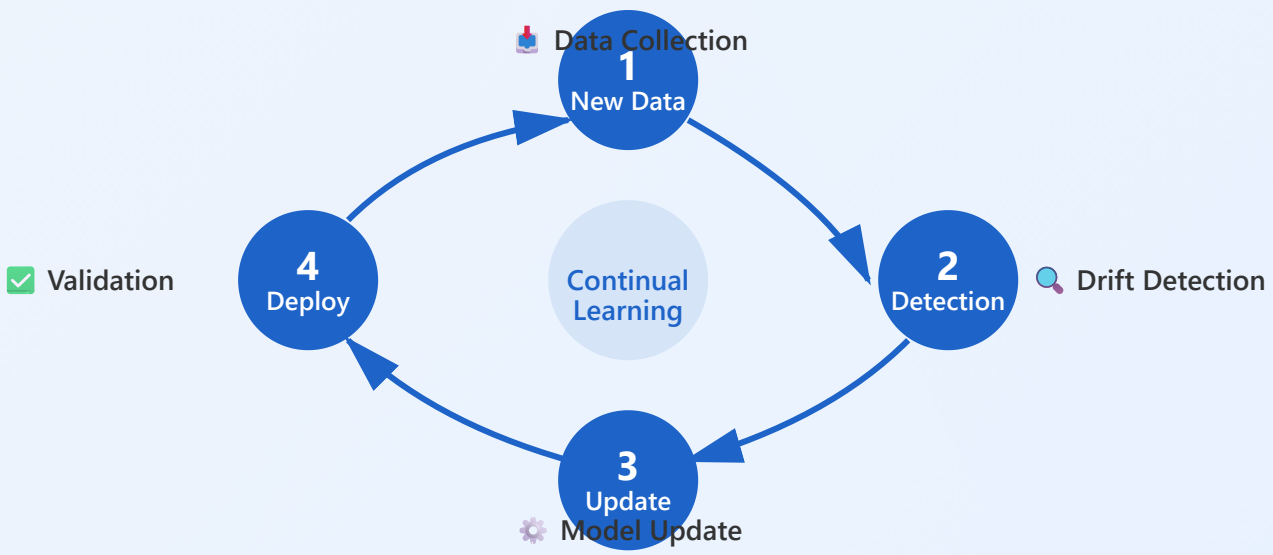
📊 Concept Drift

The phenomenon where data distribution and medical concepts change over time

⚡ Real-time Adaptation

Model update mechanism that quickly adapts to new diseases, treatments, and patient characteristics

🔄 Medical Knowledge Update Cycle



Knowledge Evolution Timeline

Initial Training	New Disease	Treatment Change	Continuous Improvement
Base Model Training	COVID-19, etc.	Protocol Update	Performance Optimization

Part 1:

Medical Knowledge Evolution

Medical Knowledge Updates (Medical Knowledge Updates)

📚 Need for Knowledge Updates

- New research findings published (hundreds of papers daily)
- Clinical guideline updates (1-2 times per year)
- New drug and treatment approvals
- Disease classification system changes (ICD codes, etc.)

🔄 Update Cycle

- Critical Updates: Immediate reflection required
- Major Guidelines: 6-12 month cycle
- Minor Revisions: Quarterly updates
- Routine Improvements: Continuous monitoring

⚠️ Impact of Changes

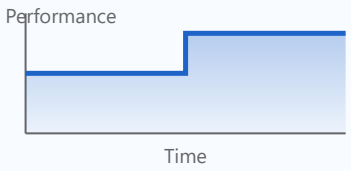
- Diagnostic criteria changes: Sensitivity/specificity adjustments
- Treatment protocol changes: Guideline updates
- Risk factor reassessment: Scoring system modifications

- Drug interactions: New contraindications

Concept Drift Detection (Concept Drift Detection)

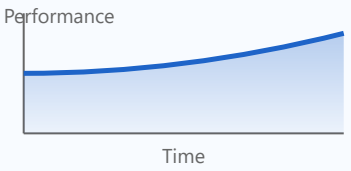
Drift Types

Sudden Drift



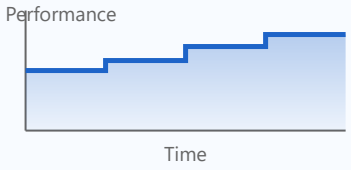
Abrupt change (e.g., Pandemic)

Gradual Drift



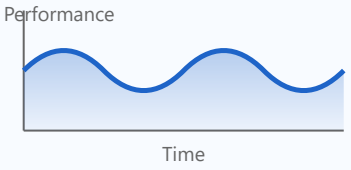
Progressive change (e.g., Population aging)

Incremental Drift



Stepwise change

Recurring Drift



Periodic pattern (e.g., Seasonal diseases)

Detection Methods

- Statistical Tests: KS test, Chi-square test
- Performance Monitoring: Accuracy, AUC tracking
- Distribution Comparison: KL divergence, Wasserstein distance

- Window-based Methods: Sliding window, ADWIN

Detection Algorithms

- DDM (Drift Detection Method)
- EDDM (Early Drift Detection Method)
- Page-Hinkley Test
- CUSUM (Cumulative Sum Control Chart)

Distribution Shift Monitoring (Distribution Shift Monitoring)

📈 Monitoring Metrics

- Input distribution: Patient demographics, test value distributions
- Output distribution: Prediction probability distribution, diagnosis frequency
- Feature importance: Feature importance changes
- Prediction confidence: Confidence score distribution

🔔 Alert Thresholds

- Yellow Alert: 10% performance degradation
- Orange Alert: 20% performance degradation or distribution shift
- Red Alert: 30%+ performance degradation, immediate action required
- Critical: Safety issue, model suspension

📊 Time Series Change Tracking

- Rolling Statistics: Moving average, standard deviation
- Trend Analysis: Trend analysis, seasonal decomposition
- Anomaly Detection: Anomaly detection

- Baseline Comparison: Comparison with baseline performance

Catastrophic Forgetting (Catastrophic Forgetting)

🧠 Causes of Forgetting

- Weight overwriting: Previous weights changed during new task learning
- Distribution imbalance: New data significantly different from previous data
- Learning rate setting: Rapid changes due to excessive learning rate
- Batch size: Unstable updates with small batches

📉 Performance Degradation Patterns

- Rapid drop in previous task accuracy (10-50%)
- Loss of specific class recognition ability
- Decreased ability to diagnose rare diseases
- Overall generalization performance degradation

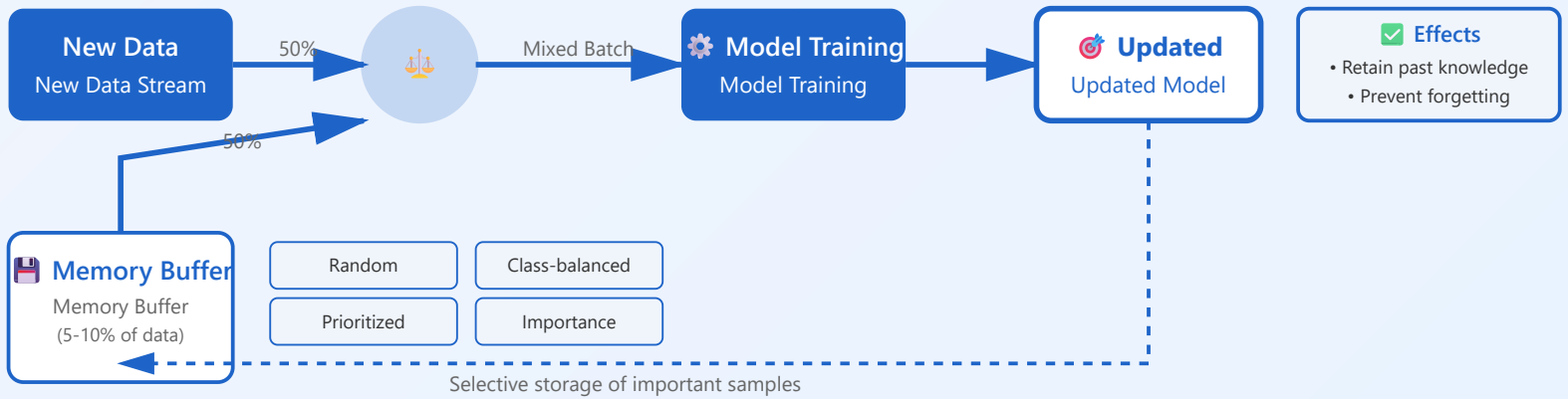
⚠️ Clinical Impact

- Increased risk of misdiagnosis of existing diseases
- Patient safety issues
- Decreased clinician trust

- Difficulty maintaining regulatory approval

Memory Replay Strategies (Memory Replay Strategies)

Memory Replay System Architecture



Replay Buffer

- Reservoir Sampling: Maintain samples with equal probability
- Class-balanced Buffer: Maintain class balance
- Importance-based Sampling: Prioritize important samples
- Buffer Size: 5-10% of total data

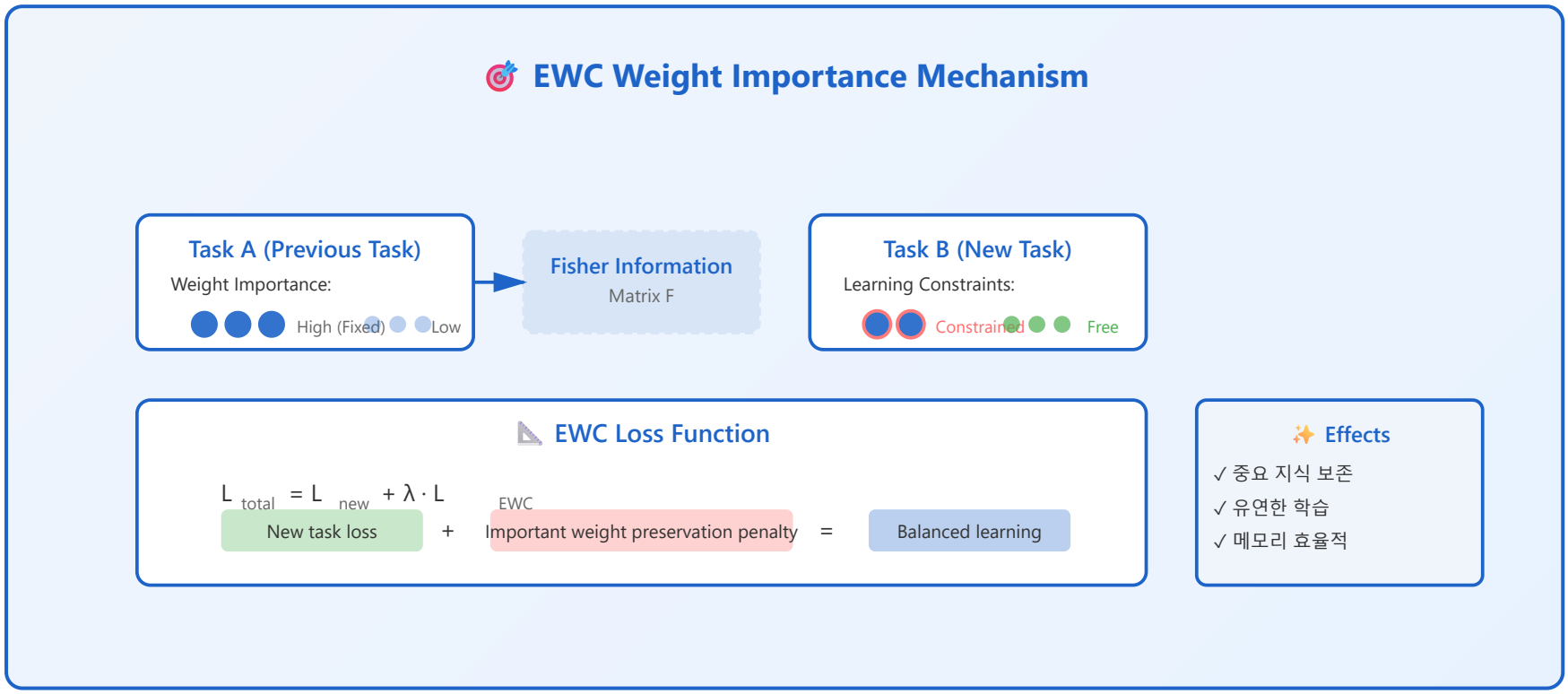
Experience Replay Methods

- Random Replay: Randomly replay past samples
- Prioritized Replay: Prioritize difficult samples
- Generative Replay: Generate past data with GAN
- Mixed Replay: Mix new data + stored samples

⚖️ Sampling Strategy

- Batch Composition: 50% new data + 50% replay
- Adaptive Sampling: Increase replay ratio on performance drop
- Stratified Sampling: Stratified balanced sampling
- Temporal Decay: Decrease weight of old samples

Elastic Weight Consolidation (EWC)



EWC Mechanism

- Fisher Information Matrix: Identify important weights
- Weight importance calculation: Identify parameters important to previous tasks
- Add constraints: Limit important weight changes
- Maintain flexibility: Freely update less important weights

Loss Function

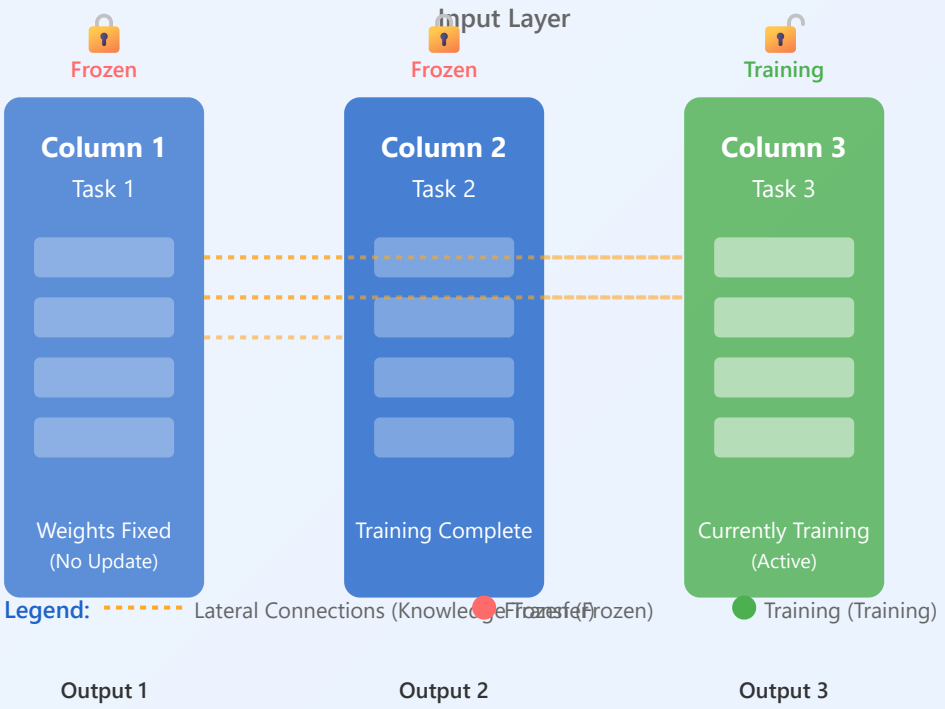
- $L_{total} = L_{new} + \lambda * L_{EWC}$
- $L_{EWC} = \Sigma F_i * (\theta_i - \theta^*_i)^2$
- F_i: Fisher information (weight importance)
- λ: Regularization strength (typically 1-1000)

Penalty Application

- High penalty for important weight changes
- Freely learn less important weights
- Cumulative penalty across multiple tasks
- Memory efficient: Store only Fisher matrix

Progressive Neural Networks (PNN)

PNN Network Architecture (Progressive Expansion)



PNN Architecture

- Column Architecture: Add new network column for each task
- Lateral Connections: Transfer features from previous columns

- Frozen Columns: Freeze previous task weights
- Adapter Layers: Transform features between tasks

➕ Expansion Process

- Step 1: Initial task learning (Column 1)
- Step 2: Add Column 2 for new task
- Step 3: Knowledge transfer via lateral connections
- Step 4: Continue expanding with additional columns as needed

⚖️ Pros and Cons

- Pros: Complete forgetting prevention, knowledge transfer possible
- Pros: Minimize inter-task interference
- Cons: Linear model size growth
- Cons: Increased memory and computational cost

Part 2:

Model Update Strategies

Incremental Learning (Incremental Learning)

🔄 Incremental Learning Concept

- Progressively integrate new data
- Update model without retraining on all data
- Memory-efficient learning approach
- Support both online/offline learning

📊 Data Stream Processing

- Mini-batch Updates: Continuous learning with small batches
- Sliding Window: Weight on recent data
- Class Incremental: Learn new classes
- Task Incremental: Add new tasks

⚡ Incremental Process

- Step 1: Collect new data batch
- Step 2: Pre-validate with current model
- Step 3: Progressive parameter update

- Step 4: Performance monitoring and validation

Online Learning Systems (Online Learning Systems)

⚡ Real-time Adaptation

- Update model immediately upon data arrival
- Utilize Stochastic Gradient Descent (SGD)
- Low Latency
- Streaming data processing

🔧 Online Pipeline

- Data Ingestion: Real-time data collection
- Feature Extraction: 즉시 특성 추출
- Model Update: Progressive weight update
- Deployment: Hot-swapping model deployment

📈 Update Strategy

- Per-sample Update: Update per sample
- Mini-batch Update: Small batch update
- Adaptive Learning Rate: Dynamic learning rate adjustment

- Forgetting Factor: Decrease weight of old data

Batch Update Strategies (Batch Update Strategies)

🗓️ Update Cycle

- Daily Updates: Update every midnight
- Weekly Updates: Weekend or weekly updates
- Monthly Updates: Major monthly updates
- Triggered Updates: Update on threshold trigger

📦 Batch Size Determination

- Small Batch (100-1000): Fast adaptation, unstable
- Medium Batch (1K-10K): Balanced choice
- Large Batch (10K+): Stable, slow adaptation
- Adaptive Batch: Adjust based on data volume

⚖️ Performance Impact

- Update frequency vs stability tradeoff
- Computational cost: Proportional to batch size
- Minimize downtime: Blue-Green deployment

- Rollback plan: Maintain previous version

Version Control for Models (Version Control for Models)

🔢 Version System

- Semantic Versioning: MAJOR.MINOR.PATCH
- MAJOR: Incompatible changes (architecture changes)
- MINOR: Feature addition, backward compatible
- PATCH: Bug fixes, performance improvements

📊 Tracking Information

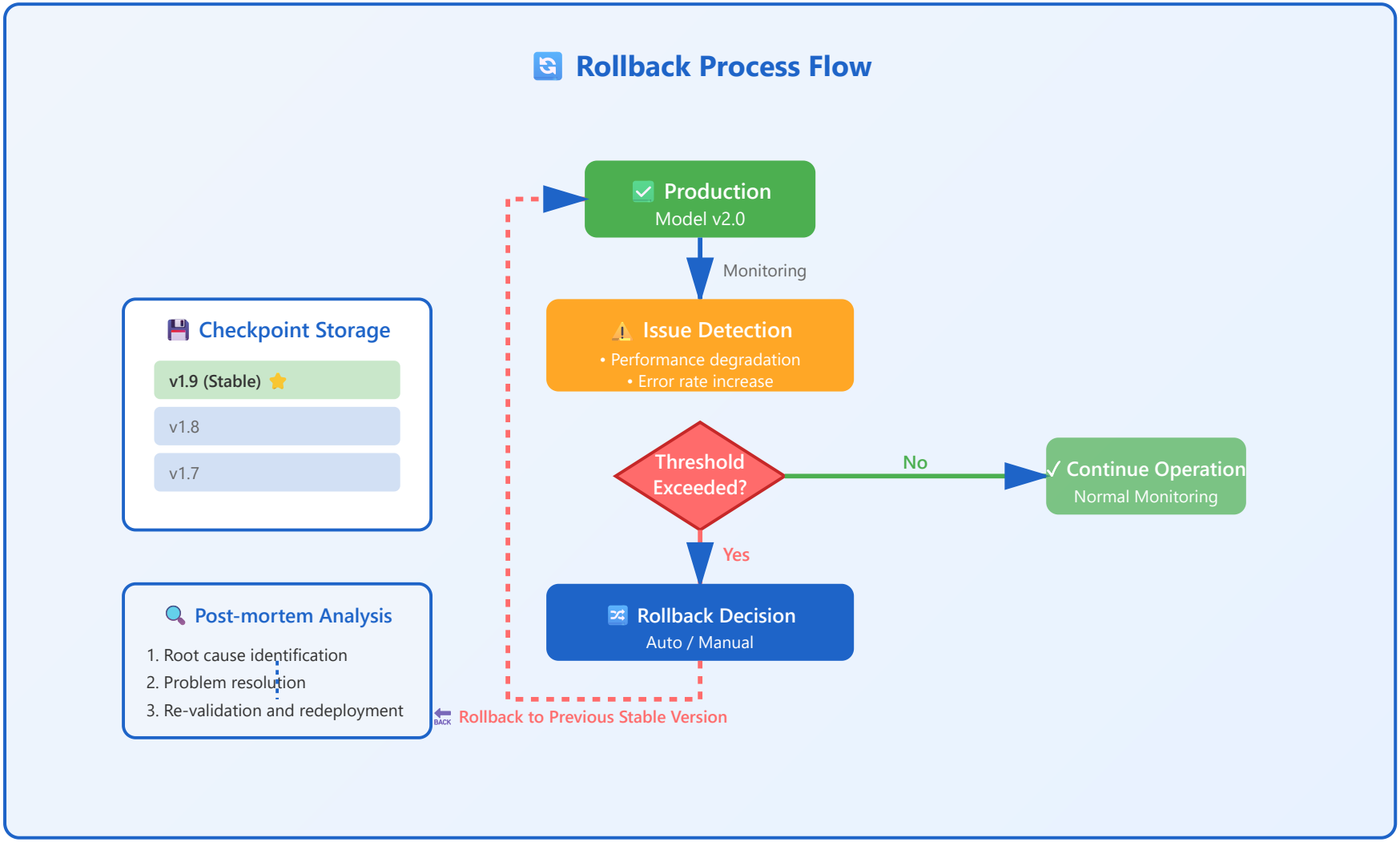
- Model weights and architecture
- Training dataset version and size
- Hyperparameter settings
- Performance metrics (accuracy, AUC, etc.)

🔧 MLOps Tools

- DVC (Data Version Control): Data + model version control
- MLflow: Experiment tracking, model registry
- Weights & Biases: Experiment visualization, collaboration

- Git + LFS: Large file version control

Rollback Mechanisms (Rollback Mechanisms)



Rollback Triggers

- Performance below threshold (e.g., accuracy < 90%)

- Error rate spike (error rate > 5%)
- User feedback spike (complaints spike)
- Safety issue discovered

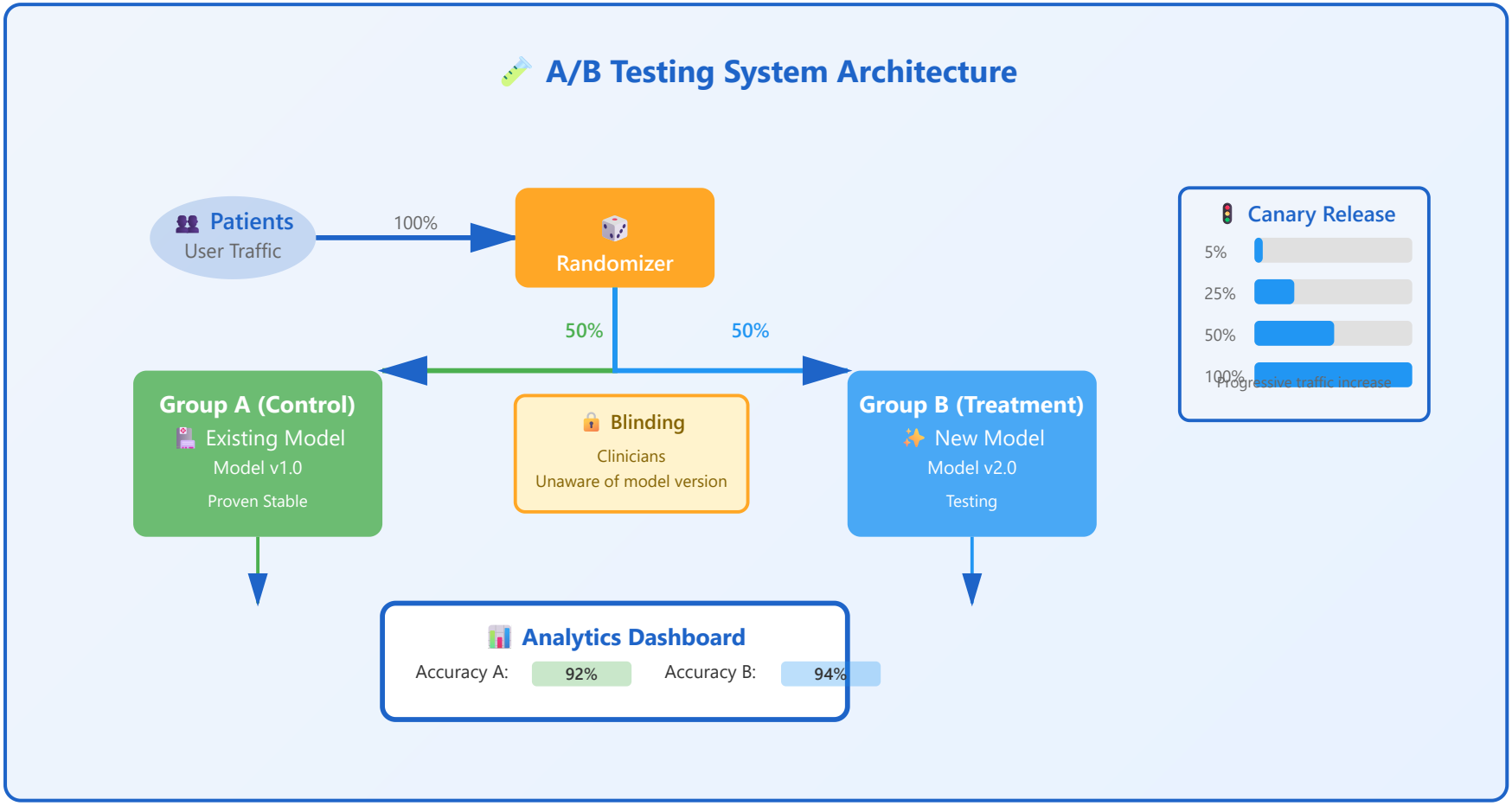
Rollback Process

- Step 1: Issue detection and alert
- Step 2: Automatic or manual rollback decision
- Step 3: Switch to previous stable version
- Step 4: Root Cause Analysis

Checkpoint Strategy

- Production Checkpoints: Deployed stable version
- Validation Checkpoints: Validated version
- Backup Frequency: Every major update
- Retention Policy: Keep at least 3 previous versions

A/B Testing in Production (A/B Testing in Production)



Test Design

- Control Group (A): Existing model (50% traffic)
- Treatment Group (B): New model (50% traffic)
- Randomization: Random assignment per patient

- Blinding: Clinicians unaware of model version

📊 Evaluation Metrics

- Primary Metrics: Diagnostic accuracy, AUC
- Secondary Metrics: Inference time, user satisfaction
- Safety Metrics: False positive/negative rates
- Business Metrics: Cost efficiency

🚦 Traffic Split

- Canary Release: 5% → 25% → 50% → 100%
- Gradual Rollout: Gradually increase traffic
- Geographic Split: Regional split testing
- Risk-based Split: Start with low-risk patients

Part 3:

Clinical System Integration

Regulatory Approval Updates (Regulatory Approval Updates)

📋 Regulatory Requirements

- FDA 510(k) or De Novo clearance
- EU MDR (Medical Device Regulation) compliance
- Clinical validation data submission
- Change Control Protocol

🔄 Model Update Approval

- Minor Updates: Simplified procedure (30 days)
- Major Updates: Full reapproval required (6-12 months)
- Predetermined Change Control Plan
- Real-time monitoring and reporting

📄 Documentation Requirements

- Detailed change log
- Performance validation report
- Risk Assessment

- Maintain Audit Trail

Clinical Validation Requirements (Clinical Validation Requirements)

🔬 Validation Protocol

- Independent test dataset (minimum 100-500 cases)
- Multi-center Validation
- Expert evaluation (Gold Standard)
- Prospective vs Retrospective study

📊 Re-validation Criteria

- On model architecture change
- Training data distribution change > 10%
- Addition of new disease or subtype
- Performance variation > 5%

⚖️ Performance Metrics

- Sensitivity ≥ 90%
- Specificity ≥ 85%

- AUC ≥ 0.90
- Subgroup performance evaluation (age, gender, race)

Performance Monitoring Dashboards (Performance Monitoring Dashboards)

📊 Real-time KPI Tracking

- Model accuracy (real-time, daily, weekly)
- Inference Latency
- Throughput: predictions/sec
- Error rate and exception frequency

🎯 Dashboard Components

- Real-time metric graphs (Line charts)
- Alert status display
- Heatmap: Performance vs time vs data characteristics
- Comparison chart: Current vs previous version

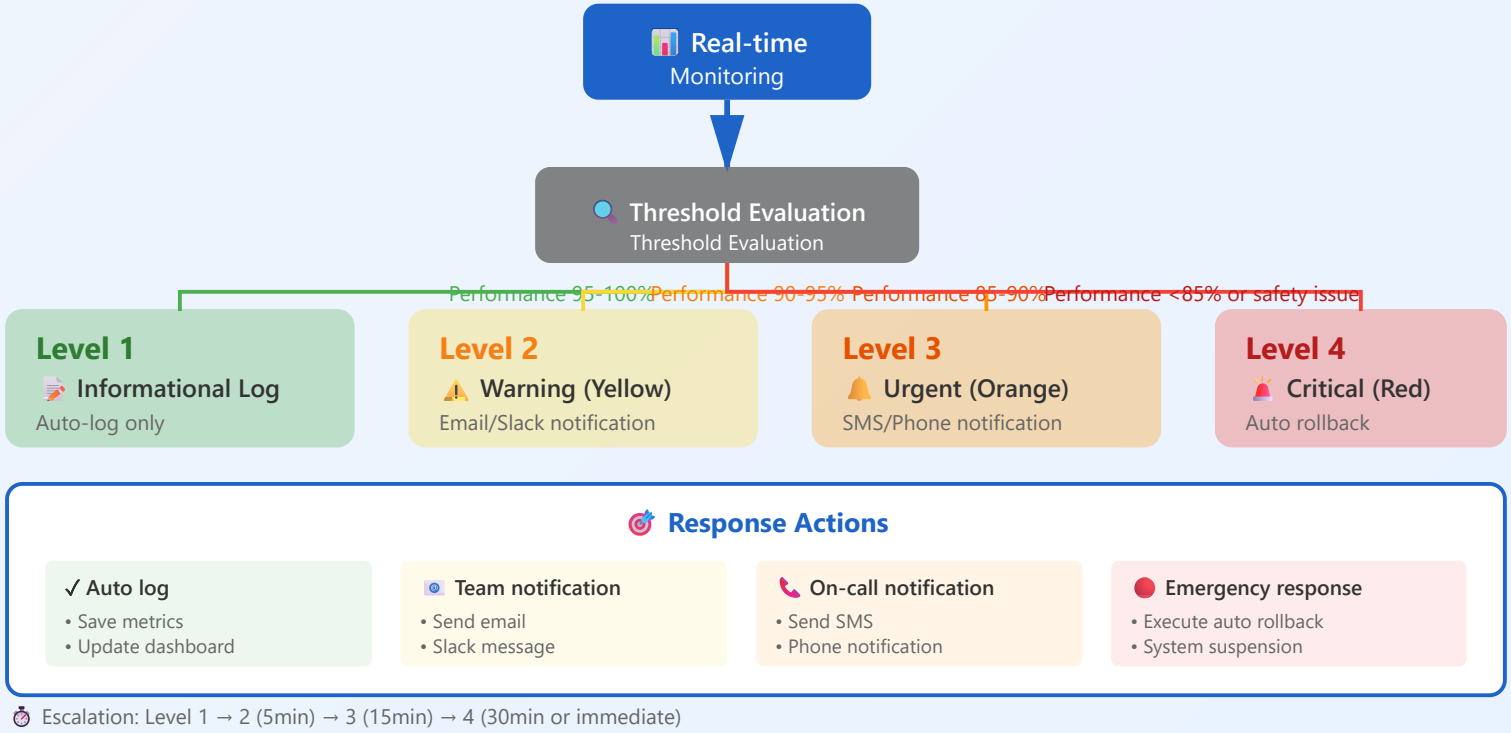
🔧 Dashboard Tools

- Grafana: Time-series data visualization
- Kibana: Log analysis and search

- Prometheus: Metric collection and alerting
- Custom Dashboards: Streamlit, Plotly Dash

Alert Systems (Alert Systems)

Multi-tier Alert System Architecture



Alert Types

- Performance Alerts: Accuracy drop, latency increase
- Data Drift Alerts: Input distribution change detection

- System Alerts: Server down, memory shortage
- Safety Alerts: Dangerous prediction patterns

⚠️ Threshold Settings

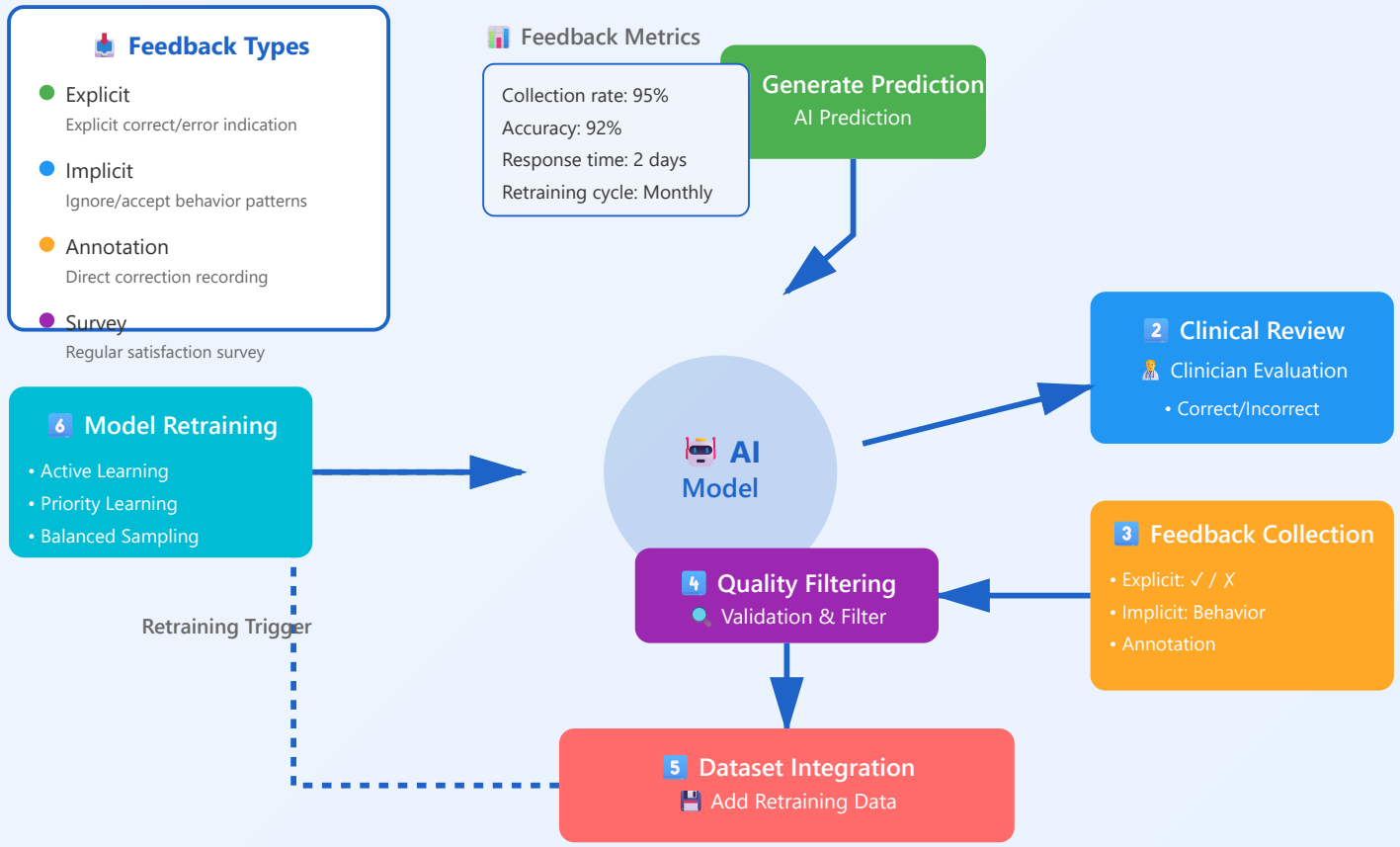
- Warning (Yellow): Accuracy < 95%
- Critical (Orange): Accuracy < 90%
- Emergency (Red): Accuracy < 85% or safety issue
- Adaptive Thresholds: Statistical anomaly detection

📢 Alert Flow

- Level 1: Auto log recording
- Level 2: Email/Slack notification
- Level 3: SMS/Phone notification (On-call)
- Level 4: Auto rollback and escalation

Feedback Integration (Feedback Integration)

Complete Feedback Loop System



User Feedback Collection

- Explicit Feedback: Explicit clinician evaluation (correct/incorrect)
- Implicit Feedback: Prediction ignore/accept behavior
- Annotations: Record corrections
- Satisfaction Survey: Regular user surveys

🔄 Feedback Loop

- Step 1: Real-time feedback collection
- Step 2: Feedback validation and filtering
- Step 3: Integrate into retraining dataset
- Step 4: Model retraining and validation

📊 Processing Pipeline

- Quality Filtering: Select reliable feedback only
- Aggregation: Aggregate multiple clinician opinions
- Prioritization: Prioritize important errors
- Active Learning: Prioritize uncertain cases for retraining

Quality Assurance (Quality Assurance)

✅ QA Process

- Unit Testing: Individual module testing
- Integration Testing: Full pipeline testing
- Regression Testing: Verify existing functionality
- Stress Testing: Load and stress testing

📋 Validation Checklist

- ✓ Model performance criteria met (accuracy, AUC)
- ✓ Inference time < 100ms (real-time system)
- ✓ Subgroup performance balance (no bias)
- ✓ Security and privacy compliance

🔍 Test Types

- Shadow Mode Testing: Parallel execution before deployment
- Adversarial Testing: Attack scenario testing
- Edge Case Testing: Extreme case evaluation

- User Acceptance Testing (UAT): Clinical user evaluation

Change Management (Change Management)

📋 Change Management Process

- Change Request: Write change request
- Impact Analysis: Impact analysis and risk assessment
- Approval: Multi-tier approval system
- Implementation: Phased implementation

🔄 Change Workflow

- Step 1: Change proposal and justification
- Step 2: Technical review and testing
- Step 3: Clinical review and approval
- Step 4: Deployment and monitoring

👥 Approval Chain

- Level 1: ML engineer review
- Level 2: Medical expert approval
- Level 3: Regulatory officer confirmation

- Level 4: Final authority approval

Documentation Standards (Documentation Standards)

📄 Required Documents

- Model Card: Model characteristics, performance, limitations
- Data Card: Training data source, characteristics, bias
- Technical Documentation: Architecture, hyperparameters
- Clinical Documentation: Usage guidelines, contraindications

🔍 Audit Trail

- All model versions and change history
- Training data lineage (Data Lineage)
- User access logs
- Prediction results and feedback records

📊 Document Templates

- Version Release Notes: Change summary
- Validation Reports: Detailed validation results
- Risk Assessment: Risk analysis and mitigation plan

- User Manual: User guide

Case Study: Pandemic Adaptation (Pandemic Adaptation)

🦠 COVID-19 Model Updates

- Initial: Based on pneumonia diagnosis model
- March 2020: Additional learning of COVID-19 features
- Mid-2020: Response to variant viruses
- 2021: Integration of vaccination status

📊 Adaptation Timeline

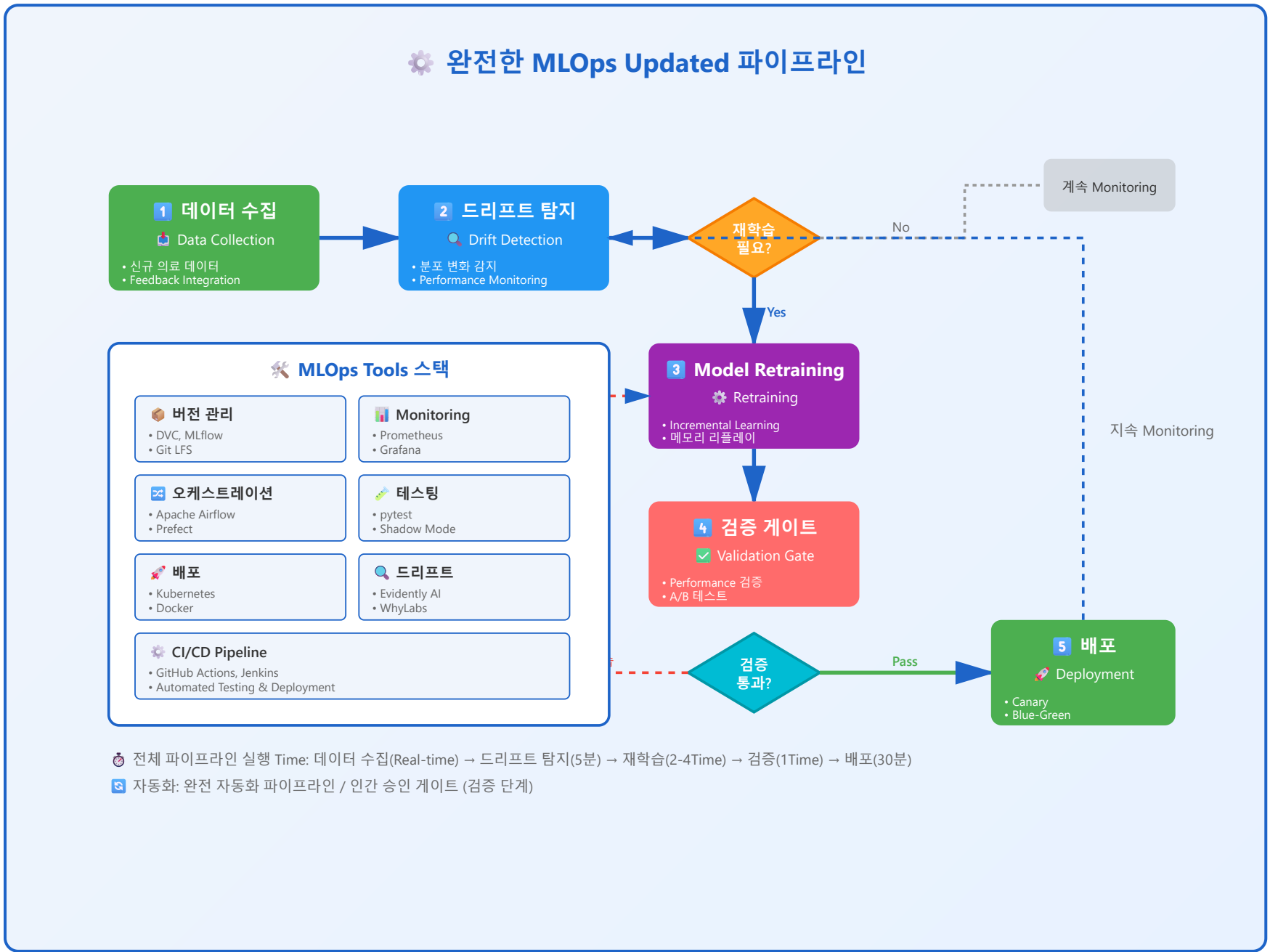
- Week 1-2: Initial data collection (China, Italy)
- Week 3-4: Emergency model update and validation
- Month 2-3: Global deployment and regional adaptation
- Year 1+: Continuous improvement and variant response

📈 Performance Changes

- Initial model: COVID-19 detection AUC 0.75
- After update: AUC 0.92 (17% improvement)
- False negative rate: 25% → 8% reduction

- Processing time: Maintained (< 50ms)

실습: Updated 파이프라인 구현



🔧 파이프라인 구성 요소

- Data Collection: 새로운 데이터 자동 수집
- Drift Detection: Monitoring 스크립트
- Retraining Trigger: 자동 재학습 조건
- Validation Gate: Performance 검증 체크포인트

💻 MLOps 통합

- CI/CD: GitHub Actions, Jenkins
- Model Registry: MLflow Model Registry
- Monitoring: Prometheus + Grafana
- Deployment: Kubernetes, Docker

📝 코드 예제

- Python: PyTorch/TensorFlow 재학습 스크립트
- Monitoring: Evidently AI, WhyLabs
- Orchestration: Apache Airflow, Prefect
- Testing: pytest, unittest 자동화

Best Practices and Lessons Learned

✅ Recommendations (Do's)

- ✓ Automate continuous monitoring
- ✓ Build multi-tier validation process
- ✓ Progressive deployment (Canary, Blue-Green)
- ✓ Prepare complete rollback mechanism

❌ Things to Avoid (Don'ts)

- ✗ Auto-deploy without validation
- ✗ Depend on single metric only
- ✗ Neglect documentation
- ✗ Ignore feedback loop

💡 Key Lessons

- Safety first: Patient safety > Performance improvement
- Transparency: Disclose all changes transparently
- Collaboration: Close clinician-engineer cooperation

- Adaptability: Respond to rapidly changing environment

Thank You!

Continuous Learning and Model Updates

📌 Key Summary

- Medical AI must adapt to continuously evolving knowledge
- Concept drift detection and response is key
- Safe and compliant update strategy is necessary
- Automated monitoring and validation system is important

🔮 Future Outlook

- Real-time adaptive AI systems
- Continuous improvement through federated learning
- Development of AI regulatory frameworks
- Self-evolving medical AI

📚 References

- Paper: "Continual Learning in Medical Imaging"

- Tools: MLflow, DVC, Evidently AI
- Standards: FDA's Predetermined Change Control Plan
- Community: MLOps Community, Healthcare AI Forums

Ho-min Park | homin.park@ghent.ac.kr