Lecture 14 - Contents

An overview of the main sections in this lecture.

Part 1
Medical Knowledge Evolution

Part 2
Model Update Strategies

Part 3
Clinical System Integration

Hands-on
Update Pipeline

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 (Continual Learning)

새로운 데이터와 지식을 지속적으로 학습하면서 이전 학습 내용을 유지하는 능력

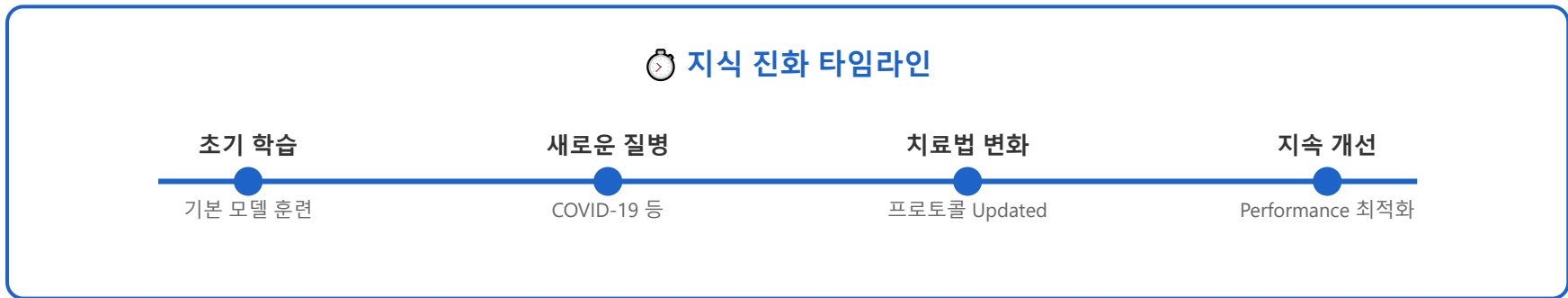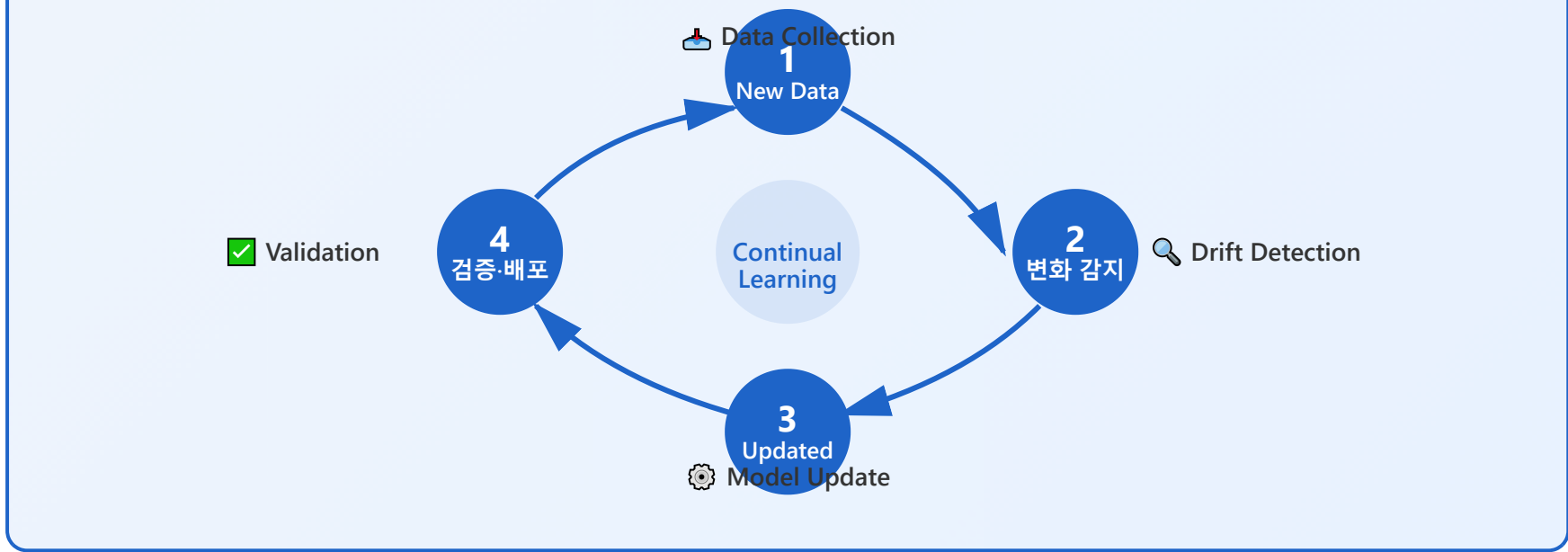## 🏥 Medical AI의 필요성

의학 지식과 진료 프로토콜은 지속적으로 변화하며, AI 모델도 이에 적응해야 함

📊 개념 드리프트

데이터 분포와 의학적 개념이 Time에 따라 변화하는 현상

⚡ Real-time Adaptation

새로운 질병, 치료법, Patients 특성에 빠르게 적응하는 모델 Updated 메커니즘

🔄 의료 지식 Updated 사이클

Data Collection

1
New Data

Continual Learning

4
검증·배포

☑ Validation

2
변화 감지

🔍 Drift Detection

3
Updated

⚙ Model Update

⏱ **지식 진화 타임라인**

초기 학습	새로운 질병	치료법 변화	지속 개선
기본 모델 훈련	COVID-19 등	프로토콜 Updated	Performance 최적화

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
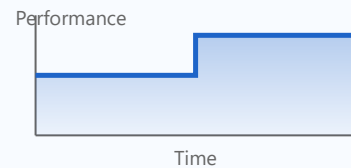
⚠️ Impact of Changes

- Diagnostic criteria changes: Sensitivity/specificity adjustments

- Treatment protocol changes: Guideline updates

- Risk factor reassessment: Scoring system modifications

- Drug interactions: New contraindications
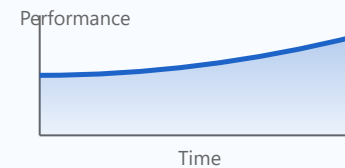
Concept Drift Detection (Concept Drift Detection)

📊 Drift Types

⚡ Sudden Drift

Performance

Time

Abrupt change (e.g., Pandemic)

📈 Gradual Drift

Performance

Time

Progressive change (e.g., Population aging)

📉 Incremental Drift

Performance

Time
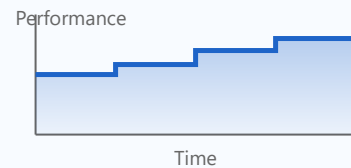
Stepwise change

🔄 Recurring Drift

Performance

Time
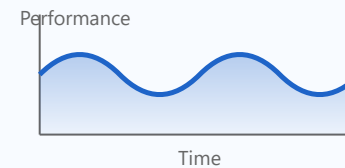
Periodic pattern (e.g., Seasonal diseases)

🔍 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

New Data
New Data Stream

→ 50% → ⚖️ → Mixed Batch → **⚙️ Model Training** (Model Training) → **🎯 Updated** (Updated Model)

50%

✅ Effects
• Retain past knowledge
• Prevent forgetting

💾 Memory Buffer
Memory Buffer
(5-10% of data)

| Random | Class-balanced |
| Prioritized | Importance |

Selective storage of important samples
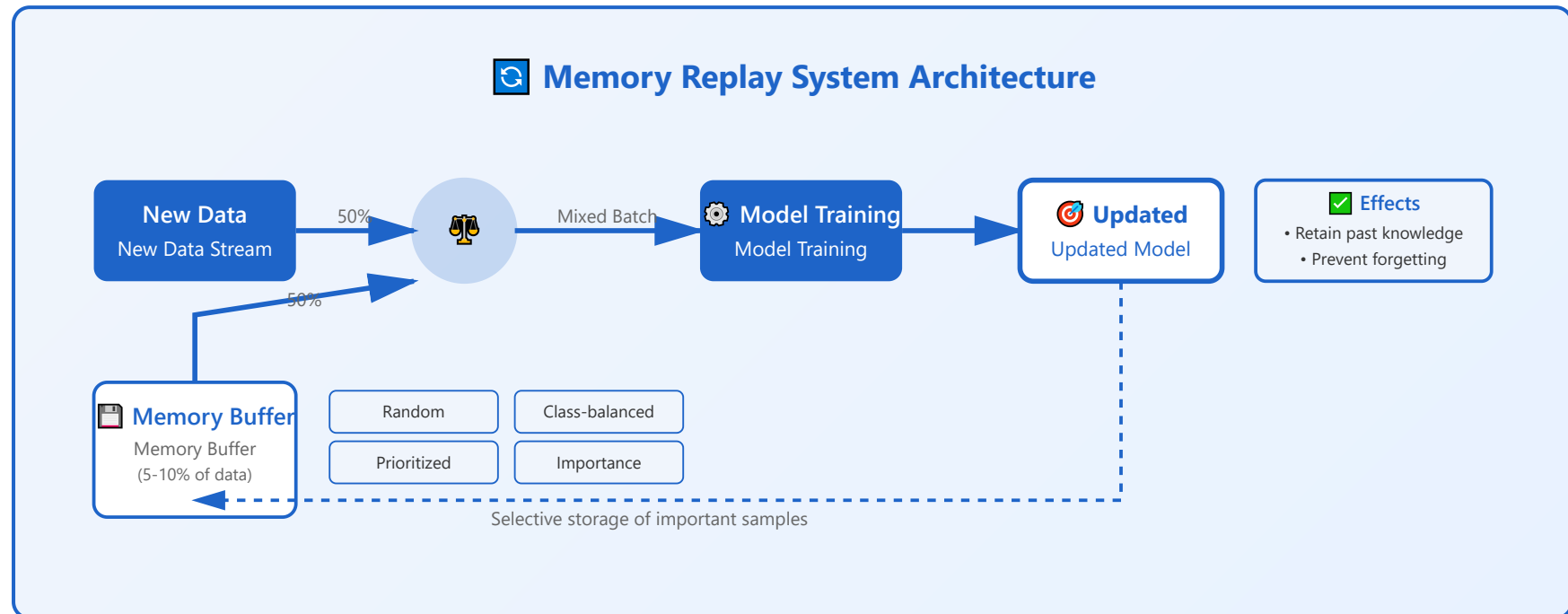
💾 Replay Buffer

- 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 Weight Importance Mechanism

Task A (Previous Task)

Weight Importance:

⬤⬤⬤ High (Fixed) ⬤⬤ Low

→ **Fisher Information**
Matrix F

Task B (New Task)

Learning Constraints:

⬤⬤ Constrained ⬤⬤ Free

⚠️ EWC Loss Function

$$L_{total} = L_{new} + \lambda \cdot L_{EWC}$$

New task loss + Important weight preservation penalty = Balanced learning

✨ Effects

✓ 중요 지식 보존
✓ 유연한 학습
✓ 메모리 효율적
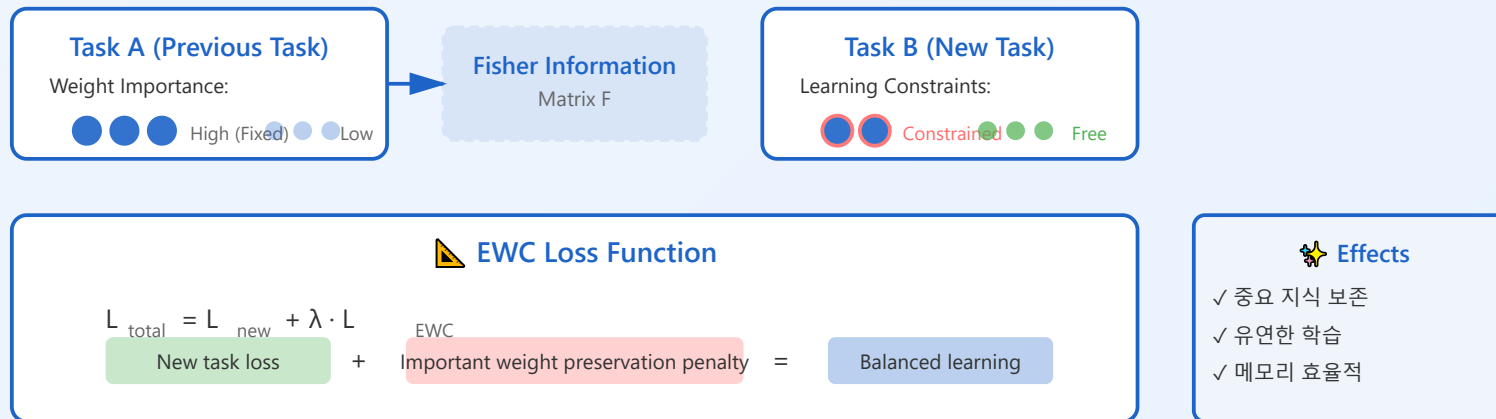
🎯 EWC Mechanism

- 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)

Input Layer

🔒
Frozen

🔒
Frozen

🔒
Training

Column 1
Task 1

Column 2
Task 2

Column 3
Task 3

Weights Fixed
(No Update)

Training Complete

Currently Training
(Active)

Legend: ········ Lateral Connections (Knowledge Transfer) ● Frozen (Frozen) ● Training (Training)

Output 1   Output 2   Output 3
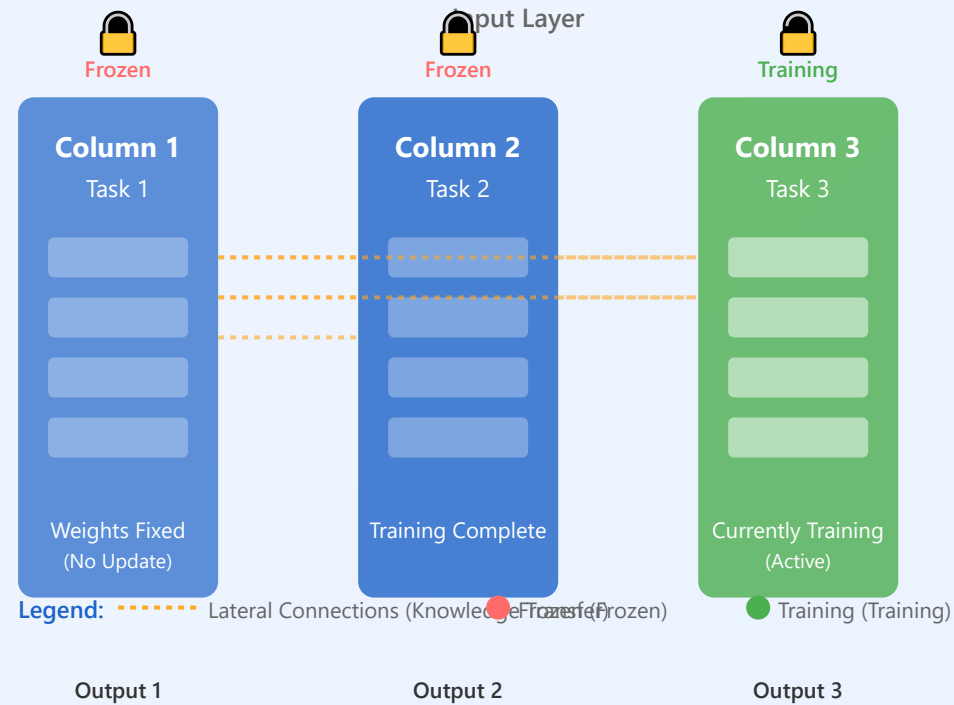
📃 PNN Architecture

- 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 Process Flow

💾 Checkpoint Storage

v1.9 (Stable) ⭐

v1.8

v1.7

🔍 Post-mortem Analysis

1. Root cause identification
2. Problem resolution
3. Re-validation and redeployment

☑ **Production**
Model v2.0

↓ Monitoring

⚠️ **Issue Detection**
• Performance degradation
• Error rate increase

Threshold Exceeded? — No → ✓ **Continue Operation** Normal Monitoring

Yes ↓

🔀 **Rollback Decision**
Auto / Manual

⬅ **Rollback to Previous Stable Version**

🔔 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
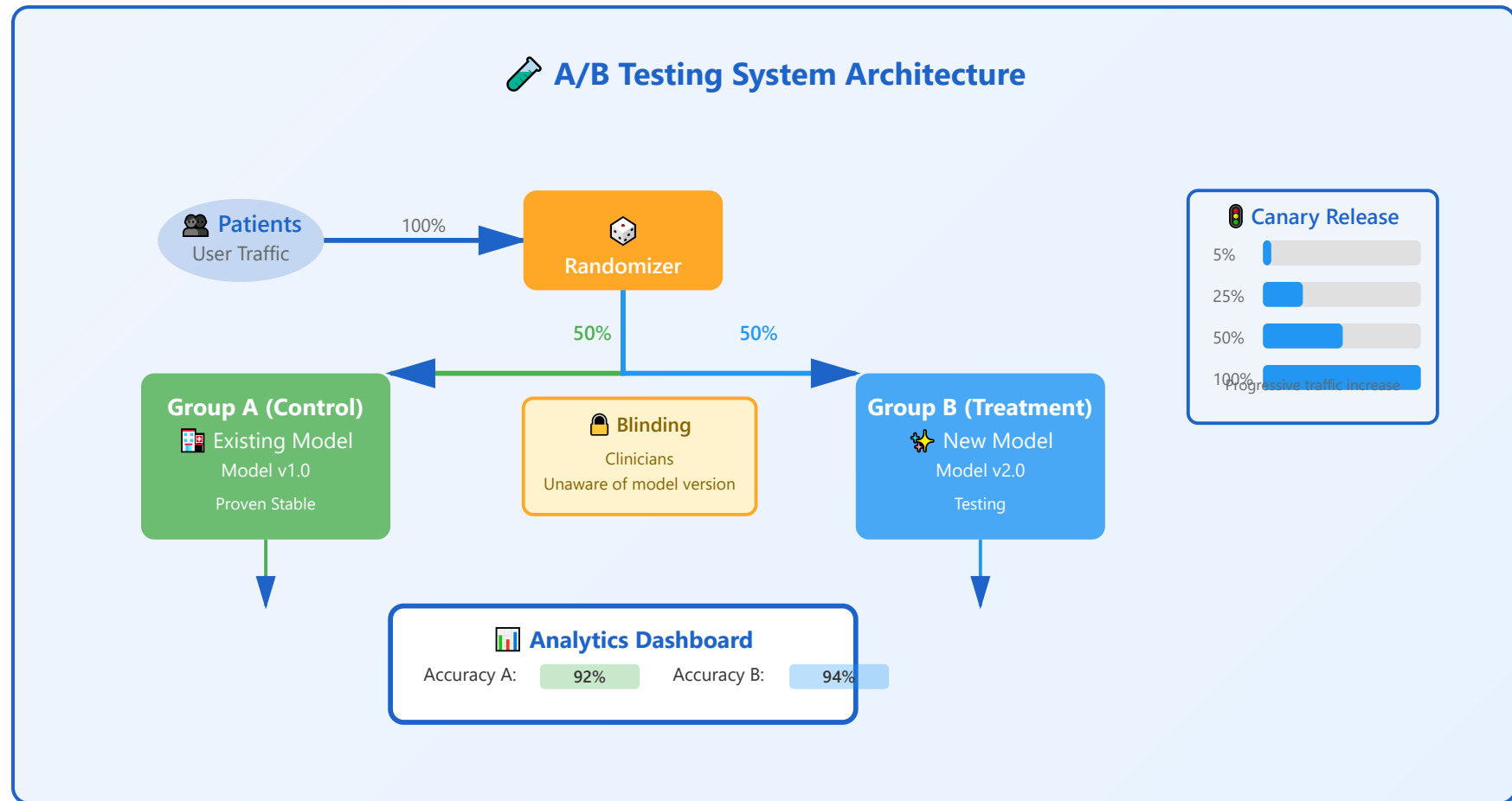
- Production Checkpoints: Deployed stable version

- 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)

🧪 A/B Testing System Architecture

👥 **Patients**
User Traffic

100% →

🎲 **Randomizer**

50%   50%

Group A (Control)
🏥 Existing Model
Model v1.0
Proven Stable

🔒 **Blinding**
Clinicians
Unaware of model version

Group B (Treatment)
✨ New Model
Model v2.0
Testing

🚦 **Canary Release**
- 5%
- 25%
- 50%
- 100%

Progressive traffic increase

📊 **Analytics Dashboard**
Accuracy A: 92%   Accuracy B: 94%

🧪 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

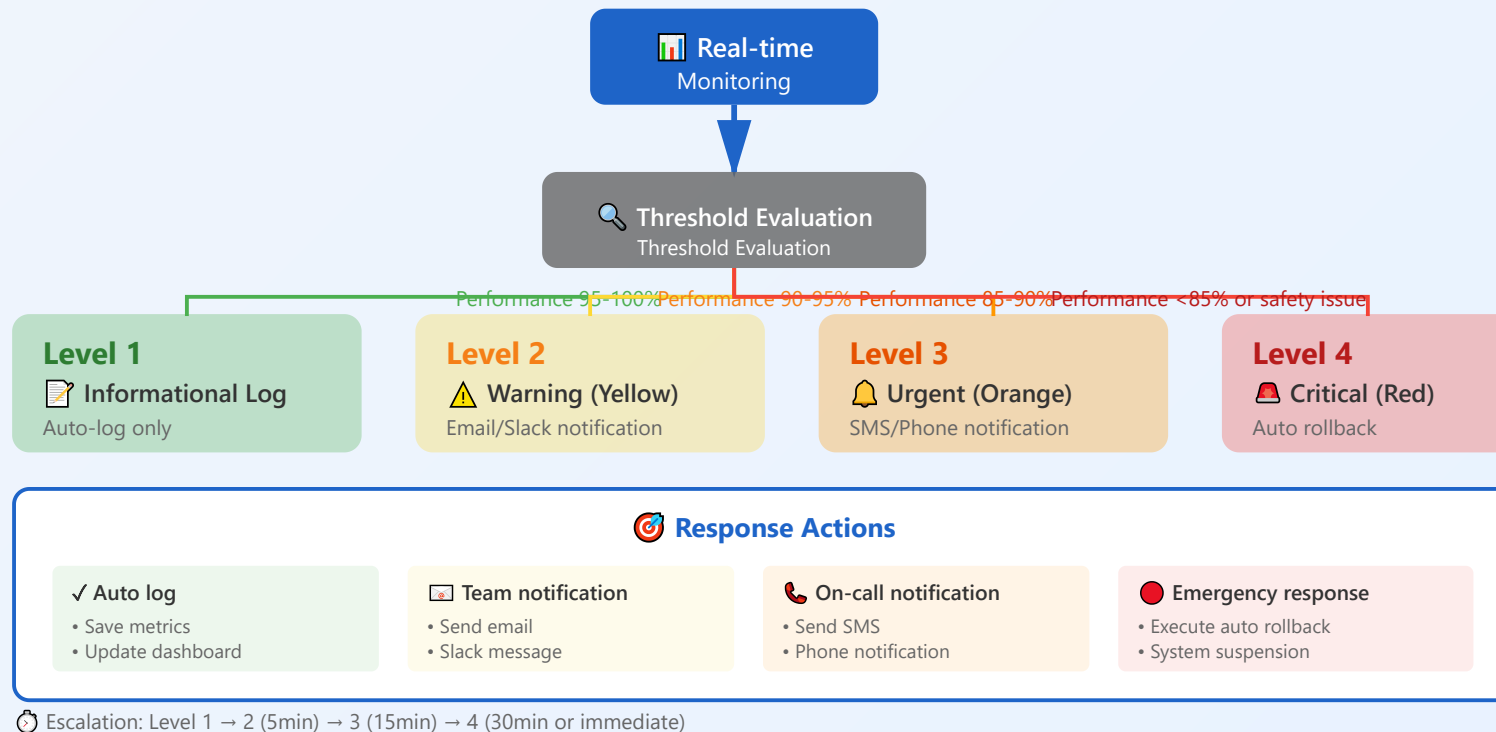- Comparison chart: Current vs previous version

🔧 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

📊 **Real-time**
Monitoring

🔍 **Threshold Evaluation**
Threshold Evaluation

Performance 95-100% Performance 90-95% Performance 85-90% Performance <85% or safety issue

Level 1
📝 Informational Log
Auto-log only

Level 2
⚠️ Warning (Yellow)
Email/Slack notification

Level 3
🔔 Urgent (Orange)
SMS/Phone notification

Level 4
🚨 Critical (Red)
Auto rollback

🎯 Response Actions

✓ **Auto log**
• Save metrics
• Update dashboard

📧 **Team notification**
• Send email
• Slack message

📞 **On-call notification**
• Send SMS
• Phone notification

🔴 **Emergency response**
• Execute auto rollback
• System suspension

⏱ Escalation: Level 1 → 2 (5min) → 3 (15min) → 4 (30min or immediate)

🚨 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
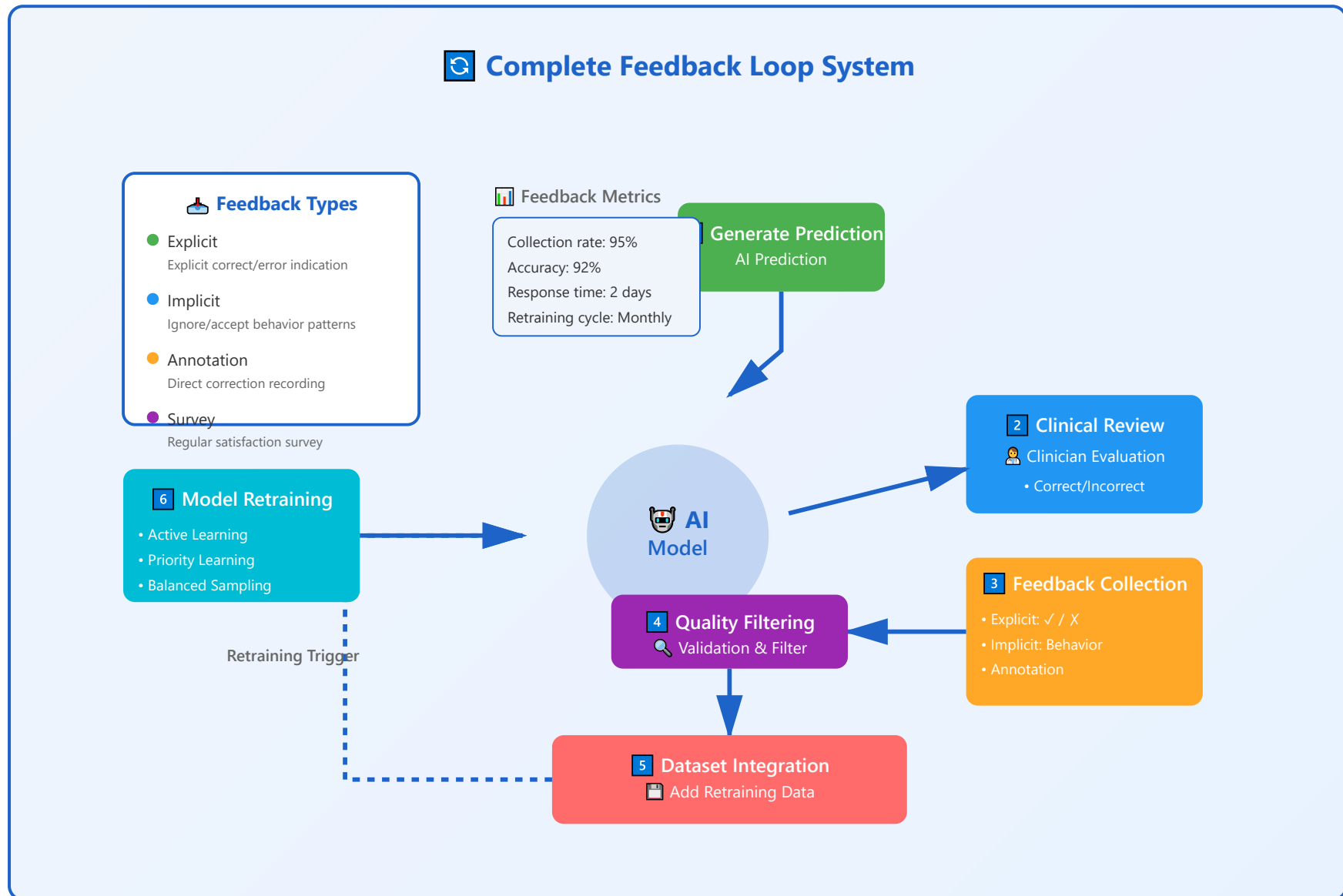
⚠️ Threshold Settings

- 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

🎛️ Feedback Types
- 🟢 **Explicit**
  Explicit correct/error indication
- 🔵 **Implicit**
  Ignore/accept behavior patterns
- 🟠 **Annotation**
  Direct correction recording
- 🟣 **Survey**
  Regular satisfaction survey

📊 Feedback Metrics

Collection rate: 95%
Accuracy: 92%
Response time: 2 days
Retraining cycle: Monthly

Generate Prediction
AI Prediction

🤖 **AI**
Model

2 Clinical Review
👨‍⚕️ Clinician Evaluation
  • Correct/Incorrect

3 Feedback Collection
  • Explicit: ✓ / X
  • Implicit: Behavior
  • Annotation

6 Model Retraining
  • Active Learning
  • Priority Learning
  • Balanced Sampling

4 Quality Filtering
🔍 Validation & Filter

5 Dataset Integration
💾 Add Retraining Data

Retraining Trigger

👨‍⚕️ 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
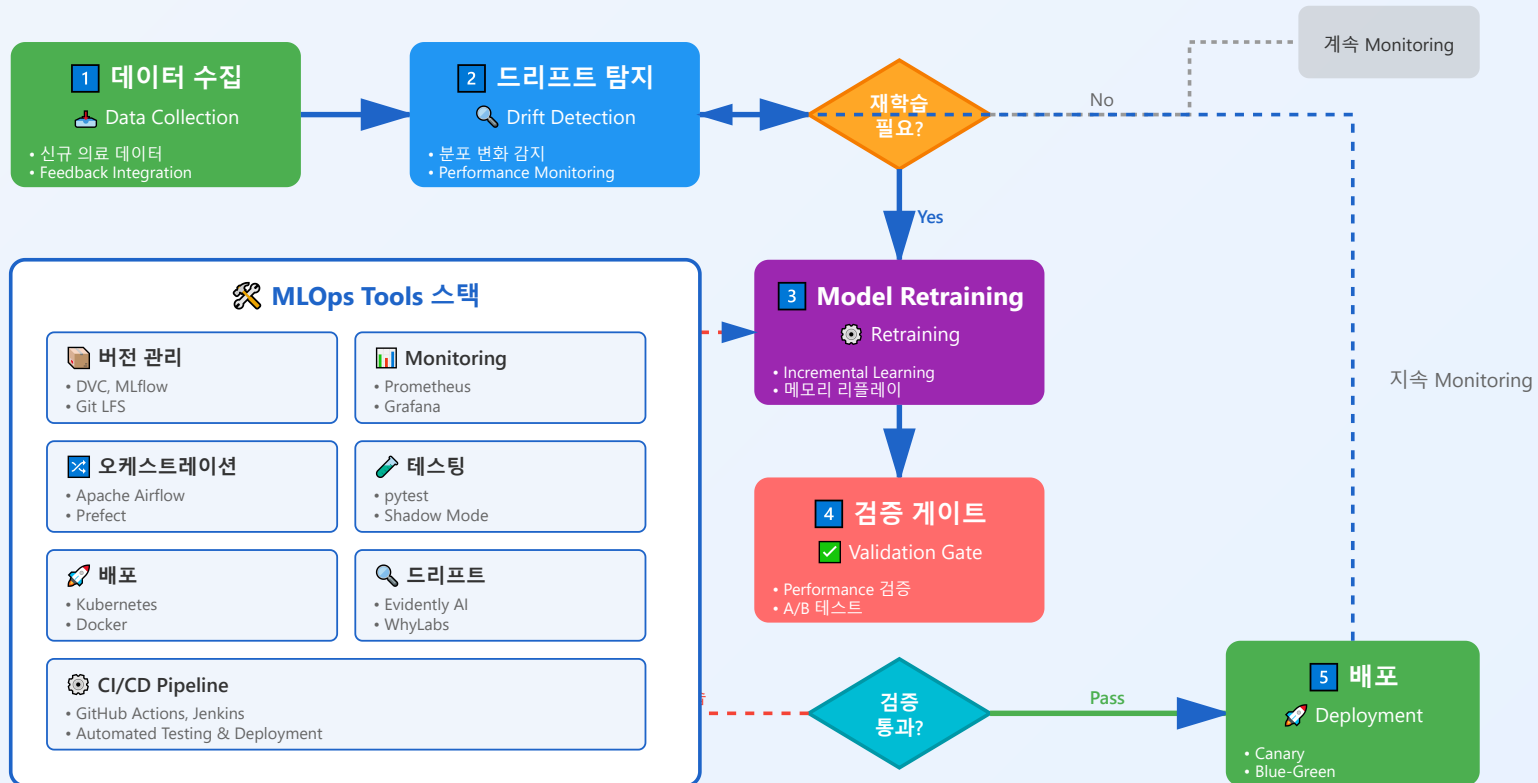- Year 1+: Continuous improvement and variant response

📈 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

⚙️ 완전한 MLOps Updated 파이프라인

1 데이터 수집
🔺 Data Collection
- 신규 의료 데이터
- Feedback Integration

2 드리프트 탐지
🔍 Drift Detection
- 분포 변화 감지
- Performance Monitoring

재학습 필요?
→ No → 계속 Monitoring
→ Yes

🛠️ MLOps Tools 스택

📦 버전 관리
- DVC, MLflow
- Git LFS

📊 Monitoring
- Prometheus
- Grafana

🔀 오케스트레이션
- Apache Airflow
- Prefect

🧪 테스팅
- pytest
- Shadow Mode

🚀 배포
- Kubernetes
- Docker

🔍 드리프트
- Evidently AI
- WhyLabs

⚙️ CI/CD Pipeline
- GitHub Actions, Jenkins
- Automated Testing & Deployment

3 Model Retraining
⚙️ Retraining
- Incremental Learning
- 메모리 리플레이

4 검증 게이트
✅ Validation Gate
- Performance 검증
- A/B 테스트

검증 통과?
→ Pass

5 배포
🚀 Deployment
- Canary
- Blue-Green

지속 Monitoring

⏱️ 전체 파이프라인 실행 Time: 데이터 수집(Real-time) → 드리프트 탐지(5분) → 재학습(2-4Time) → 검증(1Time) → 배포(30분)

🔄 자동화: 완전 자동화 파이프라인 / 인간 승인 게이트 (검증 단계)

🔧 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