

## Lecture 12 - Contents

An overview of the main sections in this lecture.

### Part 1

Model Compression

### Part 2

Distillation, Quantization, and Pruning

### Part 3

Edge and Mobile Deployment

### Hands-on

Compression Hands-on

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



Lecture 12:

# **Knowledge Distillation and Model Compression**

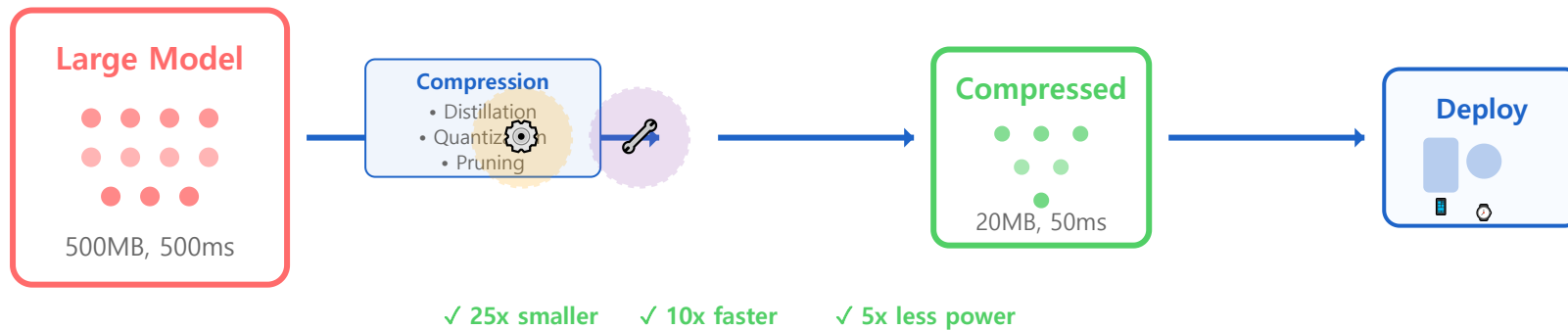
Efficient Medical AI: Compression for Clinical Deployment

# Model Compression Overview

## 왜 압축이 필요한가?

- 의료 현장의 제한된 컴퓨팅 리소스 (모바일, 웨어러블, POC 장비)
- 실시간 진단 요구사항 (저지연 추론)
- 배터리 효율성 및 전력 소비 최소화
- 개인정보 보호를 위한 온-디바이스 처리

## Compression Pipeline



## 주요 압축 방법론



## Knowledge Distillation

Teacher 모델의 지식을 작은  
Student 모델로 전달



## Quantization

가중치 비트 수 감소  
(FP32→INT8)



## Pruning

불필요한 가중치 및 뉴런 제거

**핵심 트레이드오프:** 모델 크기/속도 향상 ↔ 정확도 유지

## Part 1/3:

# Knowledge Distillation for Medical Models

1. Teacher-Student Framework
2. Soft Target Training
3. Temperature Scaling
4. Feature Distillation
5. Attention Transfer
6. Progressive & Multi-Teacher Distillation

# Teacher-Student Framework



## Teacher Model

**크기:** 대형 (수백만~수십억 파라미터)

**성능:** 최고 정확도

**용도:** 지식 제공

**예시:** ResNet-152, BERT-Large



## Student Model

**크기:** 소형 (10~100배 작음)

**성능:** Teacher에 근접

**용도:** 실제 배포

**예시:** MobileNet, DistilBERT

Knowledge Transfer



## 전달되는 지식



**Soft Targets**  
(확률 분포)

**Feature Maps**  
(중간층 출력)

**Attention Maps**  
(주목 영역)



# Soft Target Training

## Hard Targets

[0, 0, 1, 0]

**One-hot 벡터**  
정답 클래스만 1

✗ 클래스 간 관계 정보 없음

## Soft Targets

[0.05, 0.15, 0.70, 0.10]

**확률 분포**  
모든 클래스에 확률

✓ 클래스 간 유사도 학습

## Soft Targets의 장점

- ✓ 더 풍부한 정보: Teacher가 학습한 클래스 간 관계 전달
- ✓ 일반화 성능 향상: 과적합 방지
- ✓ 의료 영상: 질병 간 유사성 정보 활용 (예: 양성/악성 경계 케이스)

# Temperature Scaling

Softmax with Temperature

$$p_i = \exp(z_i/T) / \sum \exp(z_j/T)$$

T = Temperature 파라미터



**T = 1**

일반 Softmax

[0.05, 0.10, 0.80, 0.05]

Sharp 분포



**T = 3~5**

증류에 최적

[0.15, 0.20, 0.45, 0.20]

Soft 분포



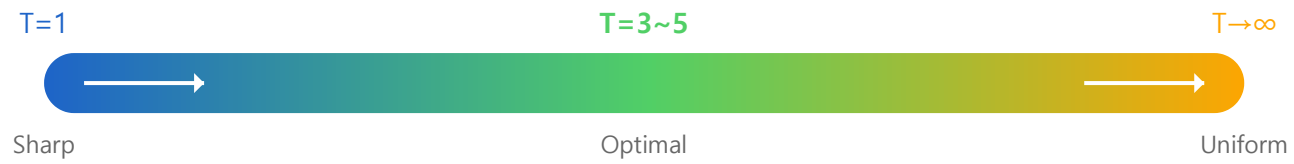
**T → ∞**

극한 케이스

[0.25, 0.25, 0.25, 0.25]

Uniform 분포





**실전 팁:** 의료 영상 분류에서  $T=3\sim 4$ 가 효과적  
높은  $T \rightarrow$  클래스 간 관계 정보 풍부  $\rightarrow$  **Student** 학습 향상

# Feature Distillation

## 특징 수준 증류 (Feature-Level Distillation)

Teacher와 Student의 중간층(intermediate layers) 출력을 매칭

### 레이어 매칭 전략

#### 1:1 Matching

Teacher Layer 12 → Student Layer 6

#### Multiple Matching

여러 레이어 동시 매칭

#### Adaptive Matching

학습 가능한 변환 사용

### Feature Distillation Loss

$$L_{\text{feature}} = ||f_T(x) - f_S(x)||^2$$

$f_T$ : Teacher features,  $f_S$ : Student features

# Attention Transfer

## Attention Transfer

Teacher 모델의 attention map을 Student에 전달하여 중요 영역 학습

## 의료 영상에서의 활용

### 병변 위치 학습

Teacher가 주목하는 병변 영역을  
Student도 학습

### X-ray/CT 분석

폐렴, 종양 등 이상 부위 attention 전  
달

### 해석 가능성 향상

임상의가 이해 가능한 판단 근거 제공

## Attention Transfer Loss

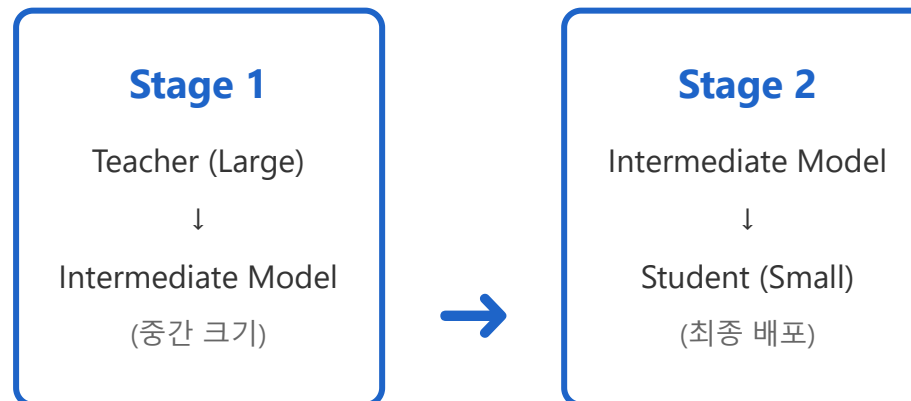
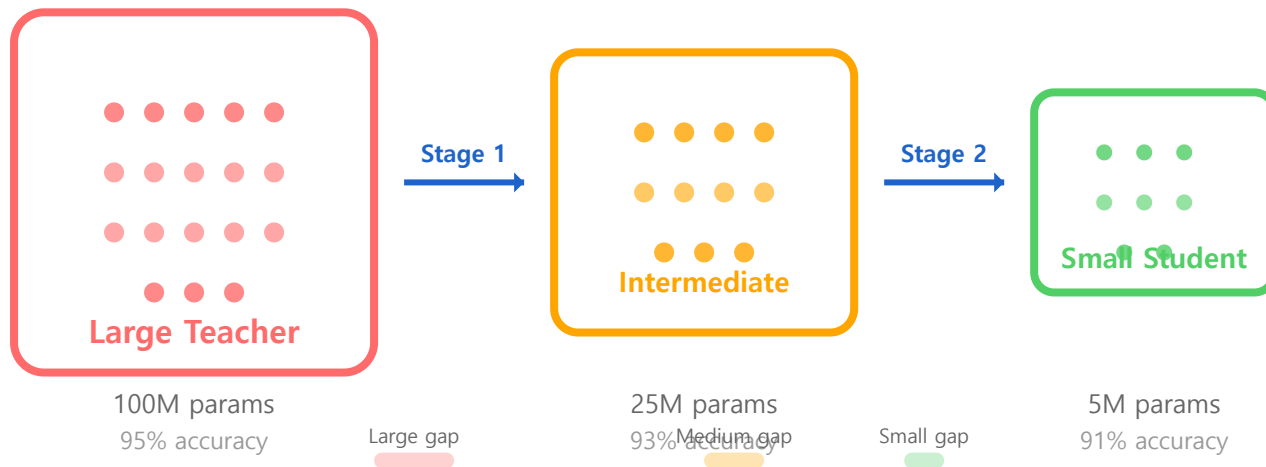
$$L_{AT} = ||A_T - A_S||_2$$

A: Attention maps (spatial activation)

# Progressive Distillation

## 점진적 증류 (Progressive Distillation)

큰 모델에서 작은 모델로 단계적으로 지식 전달



## 장점

✓ Knowledge gap 감소: 큰 차이를 단계별로 줄임

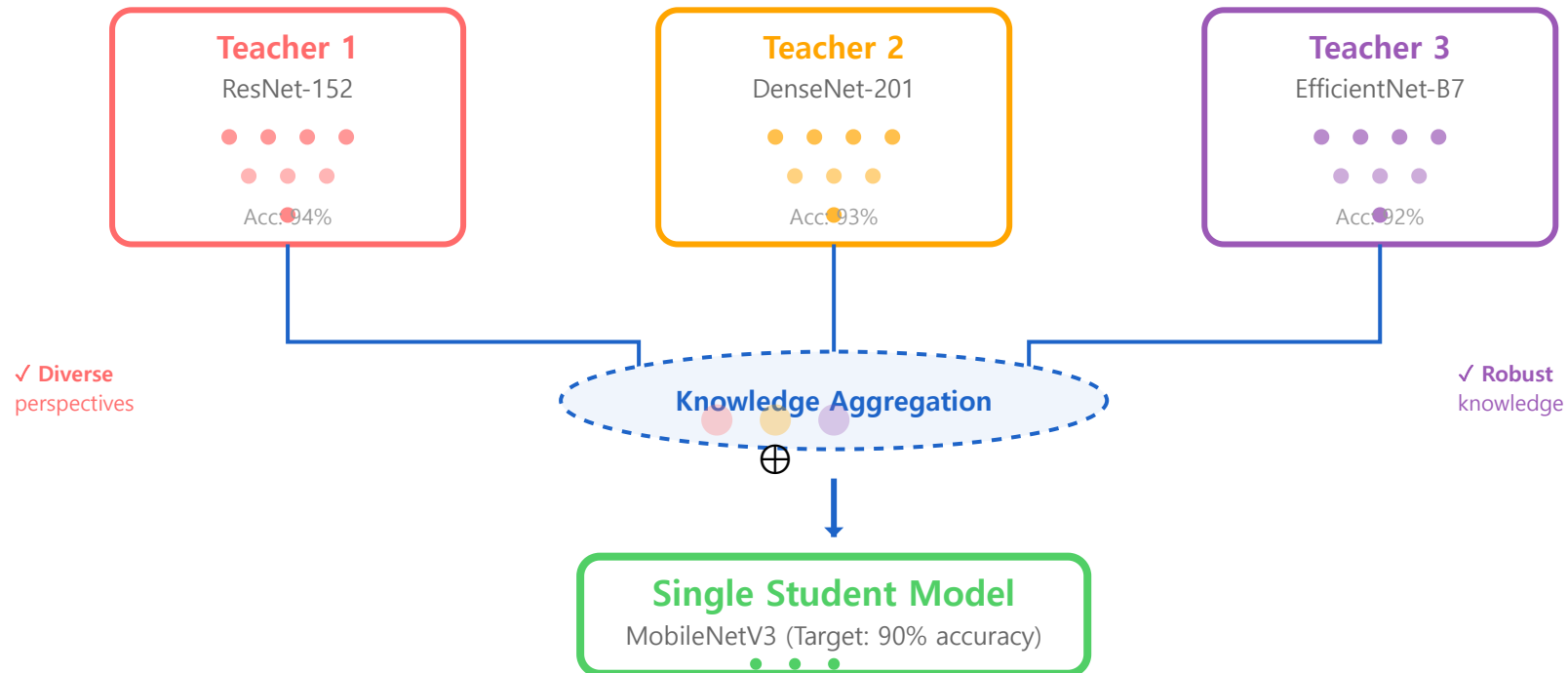
✓ 안정적 학습: 급격한 성능 저하 방지

✓ 다양한 크기 모델 확보: 배포 환경에 맞게 선택

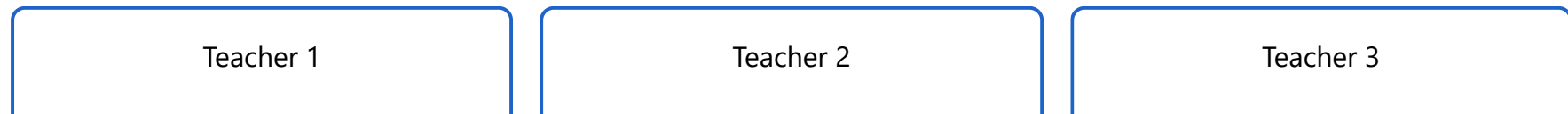
# Multi-Teacher Distillation

## Multi-Teacher Distillation

여러 Teacher 모델의 앙상블 지식을 Student에 전달



## Multiple Teachers





ResNet-152

DenseNet-201

EfficientNet-B7



Knowledge Aggregation



Single Student Model

MobileNetV3

## 지식 통합 방법



균등 가중

평균 (Average)



차등 가중

가중 평균 (Weighted)



동적 선택

Attention 기반

## Part 2/3:

# Quantization and Pruning

1. INT8/INT4 Quantization
2. Mixed Precision Strategies
3. Structured Pruning
4. Magnitude Pruning
5. Lottery Ticket Hypothesis
6. Dynamic Sparsity

# INT8/INT4 Quantization

## 정수 양자화 (Integer Quantization)

부동소수점(FP32/FP16)을 정수(INT8/INT4)로 변환하여 모델 크기 감소

### 비트 수 비교

#### FP32

32 bits  
4 bytes  
기본 학습 정밀도

#### INT8

8 bits  
1 byte  
75% 메모리 절감

#### INT4

4 bits  
0.5 bytes  
87.5% 메모리 절감

### 양자화의 효과



메모리 사용량 감소



추론 속도 향상



전력 소비 감소

**정확도 트레이드오프:** INT8은 일반적으로 1% 미만 정확도 감소  
INT4는 Quantization-Aware Training 필요

# Mixed Precision Strategies

## 혼합 정밀도 전략

레이어별로 다른 정밀도를 사용하여 성능과 효율성의 균형 달성

## 전략 예시

### 입력층

INT8/FP16  
빠른 처리 가능

### 중간층

INT8  
대부분의 연산

### 출력층

FP16/FP32  
정확한 확률 계산

레이어 민감도 분석

민감한 레이어 → FP16/FP32 유지

덜 민감한 레이어 → INT8/INT4 적용

도구

PyTorch Quantization, TensorRT, ONNX Runtime

# Structured Pruning

## 구조적 가지치기 (Structured Pruning)

전체 채널, 필터, 또는 레이어를 제거하여 구조적으로 압축

### 구조적 가지치기 유형

#### Channel Pruning

[■][■][□][■]  
채널 단위 제거

#### Filter Pruning

Filter 1: ■■■■  
Filter 2: □□□□  
Filter 3: ■■■■

#### Layer Pruning

Layer 1 → ■  
Layer 2 → □  
Layer 3 → ■

### 구조적 가지치기의 장점

✓ 하드웨어 친화적: 특수 하드웨어 불필요

✓ 실제 속도 향상: dense matrix 연산 유지

✓ 메모리 감소: 실제 파라미터 수 감소

**의료 응용:** X-ray 분류 모델에서 40% 채널 제거 → 2배 빠른 추론, 정확도 98.5% → 97.8% (1% 미만 감소)



# Magnitude Pruning

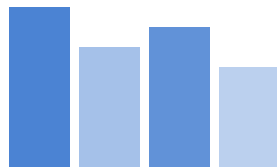
## 크기 기반 가지치기 (Magnitude Pruning)

작은 가중치 값을 0으로 만들어 희소성(sparsity) 증가

### 가지치기 프로세스



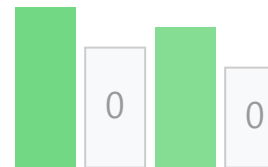
### Before Pruning



All weights

Prune →

### After Pruning



Sparse (50%)

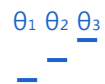
Threshold

### 임계값 설정 전략



### Global Threshold

전체 네트워크 단일 임계값



### Layer-wise Threshold

레이어별 다른 임계값



### Top-k Pruning

상위 k% 가중치만 유지

## 희소성 예시

Dense: [0.8, 0.3, -0.5, 0.1, -0.9]

↓ Threshold = 0.4

Sparse: [0.8, 0, -0.5, 0, -0.9]



40% 희소성 달성



# Lottery Ticket Hypothesis

## 복권 티켓 가설 (Lottery Ticket Hypothesis)

큰 네트워크에는 처음부터 학습해도 좋은 성능을 내는  
작은 서브네트워크("winning ticket")가 존재한다



## Winning Ticket 찾기



① 무작위 초기화 네트워크를 무작위로 초기화



② 학습 전체 네트워크를 학습



③ 가지치기 중요하지 않은 가중치 제거





#### ④ 재초기화

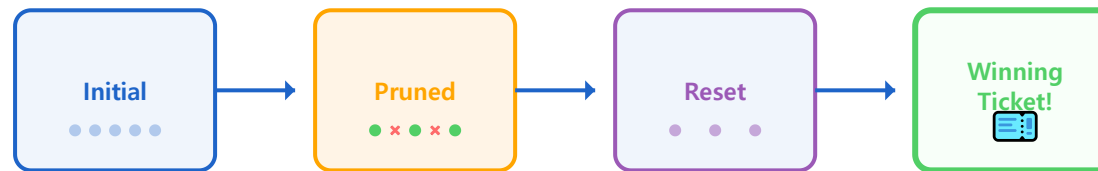
원래 초기값으로 되돌림



#### ⑤ 재학습

서브네트워크만 학습 (Winning Ticket!)

### Network Evolution



### 주요 발견



10-20%의 가중치만으로도 원래 성능 달성 가능



학습 속도도 빨라짐



초기화 방법이 매우 중요

# Dynamic Sparsity

## 동적 희소성 (Dynamic Sparsity)

학습 중 가지치기 패턴을 동적으로 조정하여 최적 희소 구조 탐색

### 정적 vs 동적 희소성

#### Static Sparsity

- 한 번 가지치기 → 고정
- 간단하지만 비최적
- 성능 제한적

#### Dynamic Sparsity

- 학습 중 지속적 재구성
- 복잡하지만 최적화
- 더 나은 성능

### 동적 가지치기 방법

#### RigL

Random Sparse Training  
주기적으로 가중치 grow/prune

#### SET

Sparse Evolutionary Training  
진화 알고리즘 기반

#### DST

Dynamic Sparse Training  
gradient 기반 재구성

**장점:** 학습 초기부터 희소성 유지 → 학습 비용 감소  
최종 희소 구조가 더 효율적

## Part 3/3:

# Edge Deployment in Healthcare

1. Mobile Health Apps & Wearable Devices
2. Point-of-Care Systems
3. Latency & Battery Optimization
4. Model Serving on Edge
5. Performance-Size Tradeoffs
6. Case Studies & Hands-on

# Mobile Health Apps

## 모바일 헬스 애플리케이션

스마트폰에서 실행되는 의료 AI 모델의 최적화 및 배포

### 모바일 제약 조건



#### 메모리

2-8 GB RAM  
모델 크기 제한



#### 연산

제한된 FLOPS  
CPU/NPU 활용



#### 배터리

전력 소비 최소화  
장시간 사용

### 최적화 전략

✓ 모델 크기 < 50MB (이상적으로 < 20MB)

✓ INT8 양자화 필수

✓ MobileNet, EfficientNet 아키텍처 활용

✓ TensorFlow Lite, Core ML 변환

### 의료 응용 사례

피부 병변 분류

당뇨병성 망막병증 스크리닝

심전도(ECG) 분석

# Wearable Device Models

## 웨어러블 디바이스 AI

스마트워치, 피트니스 트래커 등 초소형 디바이스에서의 실시간 AI 처리

## 웨어러블 디바이스 사양

### Apple Watch

RAM: 1GB

CPU: 저전력

배터리: 18시간

### Fitbit

RAM: < 512MB

매우 제한적

배터리: 5-7일

## 극한의 모델 요구사항

모델 크기:

< 5MB (이상적: < 1MB)

추론 시간:

< 100ms (실시간 처리)

정밀도:

INT8 또는 INT4

배치 크기:

1 (단일 샘플)

## 웨어러블 의료 응용



심박수 이상 탐지



수면 패턴 분석



낙상 감지



부정맥 모니터링



# Point-of-Care Systems

## POC (Point-of-Care) 진단 시스템

환자 곁에서 즉시 수행하는 진단을 위한 휴대용 AI 장비

### POC 시스템 예시

#### 휴대용 초음파

- Butterfly iQ, Lumify
- 실시간 영상 분석
- 병변 자동 탐지

#### 혈액 분석기

- 즉석 혈액 검사
- AI 기반 결과 해석
- 응급실, 외래 활용

#### 망막 카메라

- 당뇨망막병증 스크리닝
- 오프라인 동작
- 개발도상국 활용

### POC 시스템 요구사항



정확도 최우선: 의료 기기 수준 (95%+)



빠른 응답: 5-30초 이내 결과



오프라인 동작: 네트워크 불필요



개인정보 보호: 데이터 외부 전송 없음

**배포 고려사항:** FDA/CE 인증 필요, 임상 검증 필수,  
의료진 교육 및 사용자 인터페이스 중요

# Latency Optimization

## 지연시간 최적화 (Latency Optimization)

실시간 의료 응용을 위한 추론 속도 향상 기법

### Inference Latency Breakdown



### 지연시간 구성 요소

<div>데이터 전처리</div> <div>이미지 리사이징, 정규화</div>	5-20ms
<div>모델 추론</div> <div>가장 큰 비중</div>	50-500ms
<div>후처리</div> <div>결과 해석, 시각화</div>	5-10ms

### 최적화 기법



### 모델 압축

- Quantization
- Pruning
- Distillation

**2-10x 속도 향상**



### 추론 엔진

- TensorRT
- ONNX Runtime
- TFLite

**1.5-3x 속도 향상**

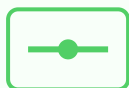


### 하드웨어 가속

- GPU
- NPU/TPU
- FPGA

**5-100x 속도 향상**

## 실시간 처리 기준



### Soft Real-time

**< 100ms**

일반 진단 보조



### Hard Real-time

**< 10ms**

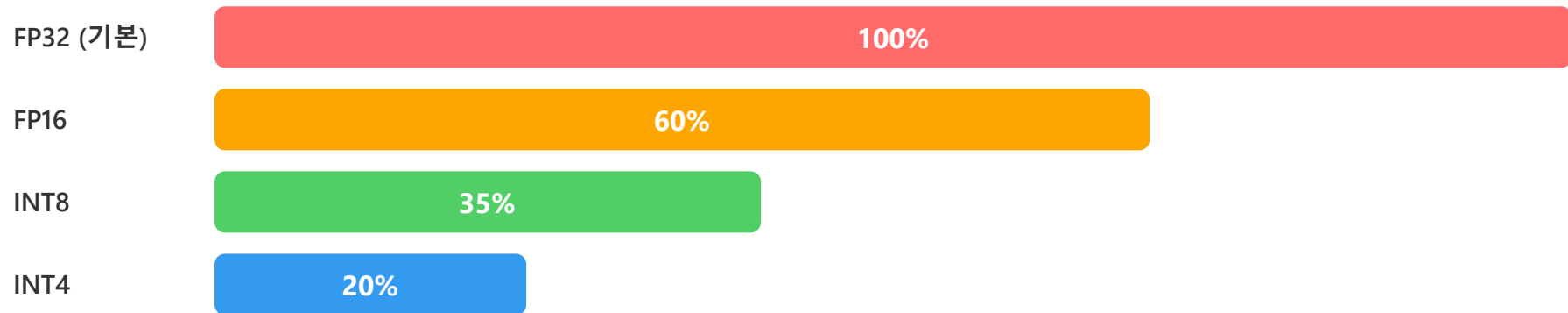
수술 로봇, 응급 시스템

# Battery Efficiency

## 배터리 효율성 최적화

모바일/웨어러블 디바이스의 장시간 사용을 위한 전력 소비 최소화

### 모델 실행 시 전력 소비



### 전력 절감 전략

- 1 적극적 양자화**  
INT8/INT4 사용으로 연산량 감소
- 2 온디맨드 추론**  
필요할 때만 모델 실행 (이벤트 기반)
- 3 배치 처리**  
여러 샘플 모아서 한 번에 처리

4

## 하드웨어 가속기 활용

NPU/DSP 사용 (CPU보다 효율적)

**트레이드오프:** 배터리 수명 vs 모델 성능/정확도  
의료 응용에서는 안전성을 해치지 않는 선에서 최적화 필수

# Model Serving on Edge

## 엣지 디바이스에서의 모델 서빙

압축된 모델을 효율적으로 배포하고 실행하기 위한 인프라

### 엣지 추론 프레임워크



#### TensorFlow Lite

- Google 제공
- Android/iOS 지원
- 광범위한 하드웨어
  - .tflite 포맷



#### Core ML

- Apple 전용
- iOS/macOS 최적화
- Neural Engine 활용
  - .mlmodel 포맷



#### ONNX Runtime

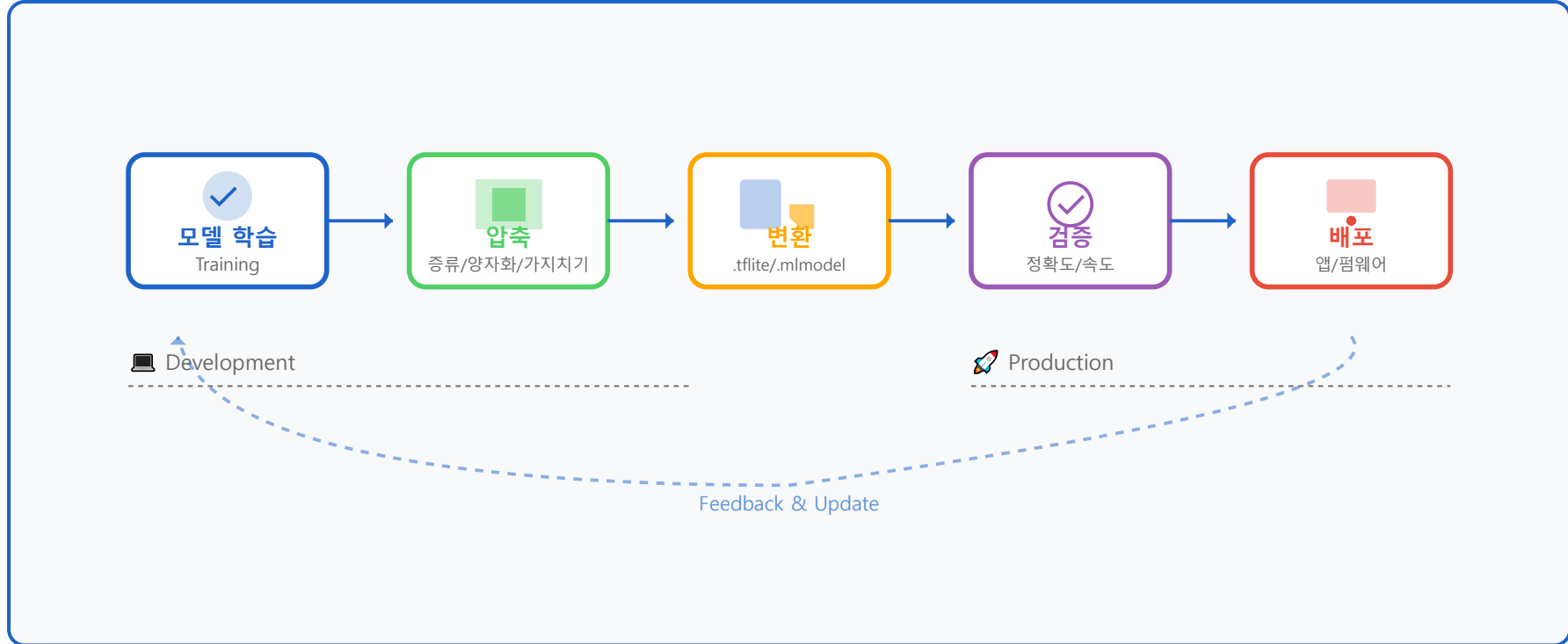
- 크로스 플랫폼
- 다양한 백엔드
- 모델 최적화 도구
  - .onnx 포맷



#### PyTorch Mobile

- PyTorch 기반
- Android/iOS
- 쉬운 변환
  - .ptl 포맷

### 배포 파이프라인



## 모범 사례

✓ 디바이스별 벤치마크 필수 (실제 하드웨어에서 테스트)

✓ OTA(Over-The-Air) 업데이트 지원

✓ Fallback 메커니즘 (클라우드 추론)

✓ 모니터링 및 로깅



# Performance vs Size Tradeoffs

## 성능-크기 트레이드오프

압축 수준에 따른 모델 크기, 정확도, 속도의 균형점 찾기

## 파레토 최적 곡선 (Pareto Frontier)

### Large Model

크기: 500MB  
정확도: 95%  
속도: 500ms

### Medium Model

크기: 100MB  
정확도: 93%  
속도: 100ms

### Small Model

크기: 20MB  
정확도: 90%  
속도: 20ms

### Tiny Model

크기: 5MB  
정확도: 85%  
속도: 5ms

## 모델 선택 기준

### 클라우드 배포

Large/Medium 모델

정확도 최우선

### 모바일 앱

Medium/Small 모델

균형잡힌 성능

### 웨어러블

Small/Tiny 모델

크기 최소화

#### 의사결정 가이드:

1. 최소 허용 정확도 결정
2. 목표 디바이스의 제약 조건 파악
3. 파레토 곡선 상에서 최적점 선택

# Accuracy Preservation

## 정확도 유지 (Accuracy Preservation)

압축 과정에서 모델 성능을 최대한 보존하는 방법

### 압축률에 따른 정확도 변화

압축 없음

95%

경량 압축

93%

중간 압축

90%

고강도 압축

85%

← 임계점 (Critical Threshold)

### 정확도 유지 전략

1

#### Quantization-Aware Training (QAT)

학습 중 양자화 시뮬레이션 → 정확도 손실 최소화

2

#### Fine-tuning after Compression

압축 후 재학습으로 성능 회복

3

### Sensitivity Analysis

레이어별 민감도 분석 후 선택적 압축

4

### Knowledge Distillation

Teacher 지식 활용으로 정확도 보존

#### 의료 AI 특수 고려사항:



정확도 하락이 환자 안전에 직결 → 엄격한 임계값 설정 필수  
FDA/CE 승인 기준 충족 여부 확인

# Case Study: Mobile Diagnostics

## 사례 연구: 모바일 피부암 진단 앱

딥러닝 모델 압축을 통한 실시간 모바일 진단 시스템 구현

### 문제 정의



목표: 피부 병변 사진으로 양성/악성 분류



플랫폼: iOS/Android 모바일 앱



요구사항: 3초 이내 결과, 오프라인 동작

### 압축 접근법

#### Step 1: Base Model

ResNet-50 (98MB, FP32)

정확도: 94.5%

추론: 2.5초



#### Step 2: Knowledge Distillation

MobileNetV3 Student

크기: 24MB

정확도: 93.2%



### Step 3: INT8 Quantization

크기: 6MB (75% 감소)

정확도: 92.8%

추론: 0.8초



### Final: TFLite Optimization

최종 크기: 5.2MB

정확도: 92.5%

추론: 0.6초

### 최종 결과

**94.7%**

크기 감소

**76%**

속도 향상

**-2.0%**

정확도 변화

# Hands-on: Model Compression

## 실습: 모델 압축 파이프라인

PyTorch를 이용한 Knowledge Distillation 및 Quantization 실습

### 1. Knowledge Distillation 코드

```
# Teacher-Student 증류 손실
def distillation_loss(student_logits, teacher_logits, labels, T=3):
    # Soft targets
    soft_loss = nn.KLDivLoss()(F.log_softmax(student_logits/T, dim=1),
                               F.softmax(teacher_logits/T, dim=1)) * T*T

    # Hard targets
    hard_loss = nn.CrossEntropyLoss()(student_logits, labels)

    return 0.7 * soft_loss + 0.3 * hard_loss
```

### 2. INT8 Quantization 코드

```
import torch.quantization

# 정적 양자화
model.qconfig = torch.quantization.get_default_qconfig('qnnpack')
model_prepared = torch.quantization.prepare(model)

# 캘리브레이션
model_prepared(calibration_data)

# 양자화 적용
model_quantized = torch.quantization.convert(model_prepared)
```

유용한 도구 및 라이브러리

### PyTorch

torch.quantization  
torch.nn.utils.prune

### TensorFlow

TF-MOT (Model Optimization)  
Quantization-Aware Training

### ONNX

onnxruntime  
Cross-framework

### Hugging Face

Optimum  
Transformer 최적화

### 실습 과제:

CIFAR-10 데이터셋으로 ResNet 모델 압축해보기  
(증류 + 양자화 파이프라인 구현)

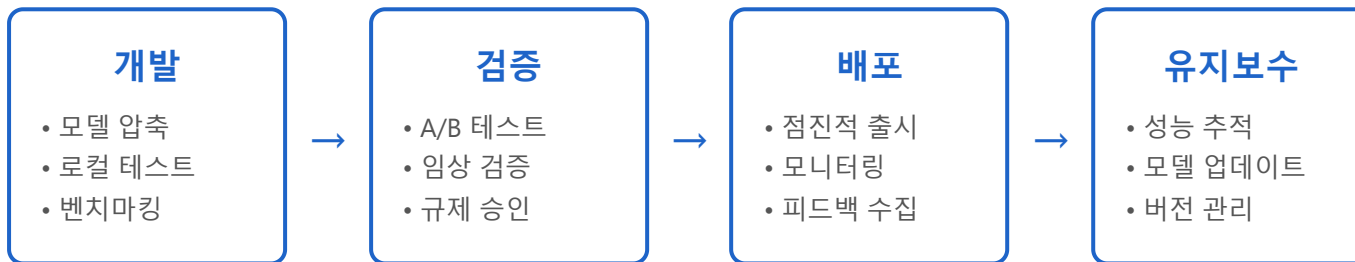


# Deployment Strategies

## 배포 전략 및 체크리스트

압축 모델의 성공적인 프로덕션 배포를 위한 전략

### 배포 워크플로우



### 배포 전 체크리스트

#### 성능 (Performance)

- ☐ 정확도 목표 달성 (Accuracy goal achieved)
- ☐ 지연시간 요구사항 충족 (Latency requirements met)
- ☐ 메모리 제약 내 (Within memory constraints)

#### 안정성 (Stability)

- ☐ Edge case 테스트 (Edge case testing)
- ☐ 에러 핸들링 (Error handling)
- ☐ Fallback 메커니즘 (Fallback mechanism)

#### 규제 (Regulatory)

- ☐ FDA/CE 인증 (필요시) (FDA/CE certification (if needed))
- ☐ 개인정보 보호 (Data privacy)
- ☐ 설명 가능성 (Explainability)

#### 운영 (Operations)

- ☐ 로깅/모니터링 (Logging/Monitoring)
- ☐ OTA 업데이트 (OTA updates)
- ☐ 사용자 교육 (User education)

## 플랫폼별 권장 전략

플랫폼	압축 방법	프레임워크
iOS	증류 + INT8	Core ML
Android	증류 + INT8	TF Lite
Web	경량 압축	TensorFlow.js
Embedded	INT4 + 가지치기	TF Lite Micro

# Thank You

## 강의 요약

### Part 1

#### Knowledge Distillation

- Teacher-Student 프레임워크
- Soft targets & Temperature
- Feature/Attention 전달

### Part 2

#### Quantization & Pruning

- INT8/INT4 양자화
- 구조적/비구조적 가지치기
- 동적 희소성

### Part 3

#### Edge Deployment

- 모바일/웨어러블 최적화
- POC 시스템
- 성능-크기 트레이드오프

## 핵심 베스트 프랙티스

✓ 압축 전 정확도 목표 및 제약사항 명확히 정의

✓ 여러 압축 기법 조합으로 시너지 효과

✓ 실제 디바이스에서 벤치마크 필수

✓ 의료 AI는 정확도 하락에 특히 민감 - 임상 검증 필수

✓ 배포 후 지속적 모니터링 및 업데이트

## 추천 도구 모음

**압축:** PyTorch Quantization, TensorFlow Model Optimization

**변환:** ONNX, TensorFlow Lite, Core ML Tools

**추론:** TensorRT, ONNX Runtime, TFLite

**벤치마킹:** MLPerf Mobile, AI Benchmark

# Thank You!

Lecture 12: Knowledge Distillation and Model Compression

Questions? Contact via course forum or office hours