10th Anniversary

GISC 2026

Global ICT Standards Conference 2026

피지컬 AI ICT 융합 표준프레임워크 공청회

로봇지능 분할실행 기술

윤주상 교수
동의대학교

ICT Standards and Intellectual Property:
Shaping the Physical AI

주최 과학기술정보통신부 지식재산처 주관 국립전파연구원 TTA IITP KEA kista ETRI

Index

01 연구 배경

02 피지컬 AI 분할 학습 및 실행 기술 소개

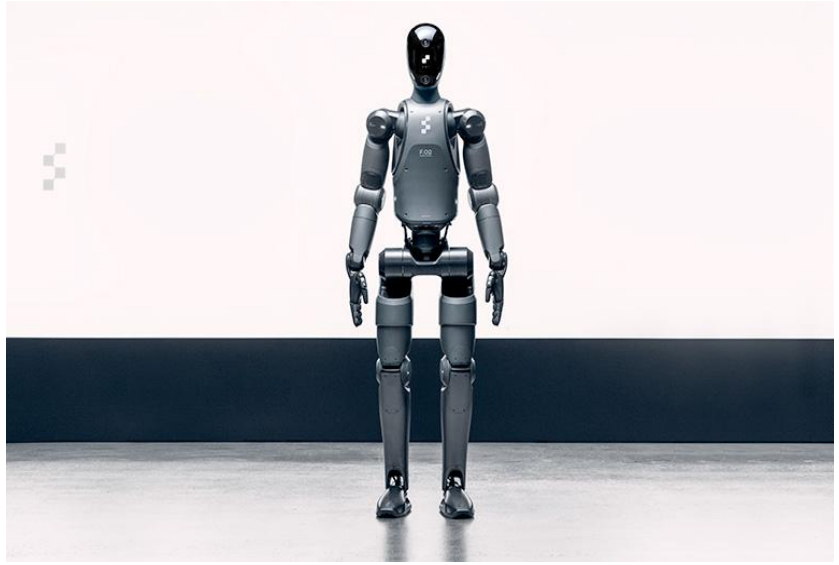
03 로봇 지능 분할 실행 기술 소개

연구 배경

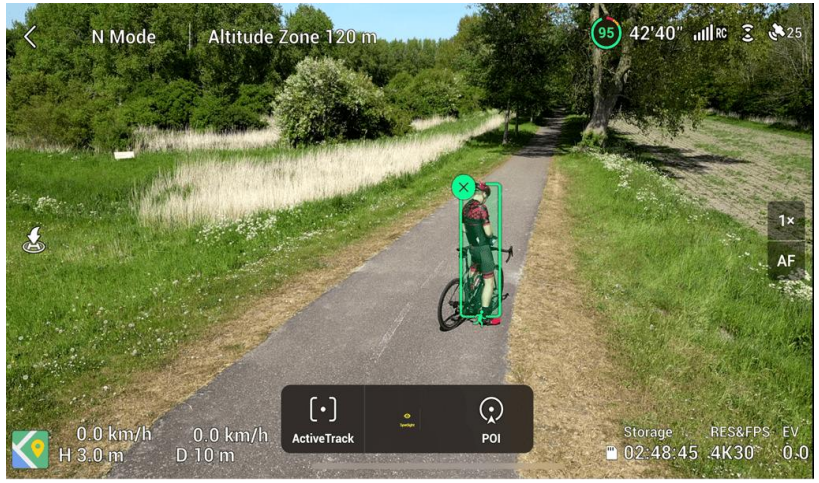


연구 배경

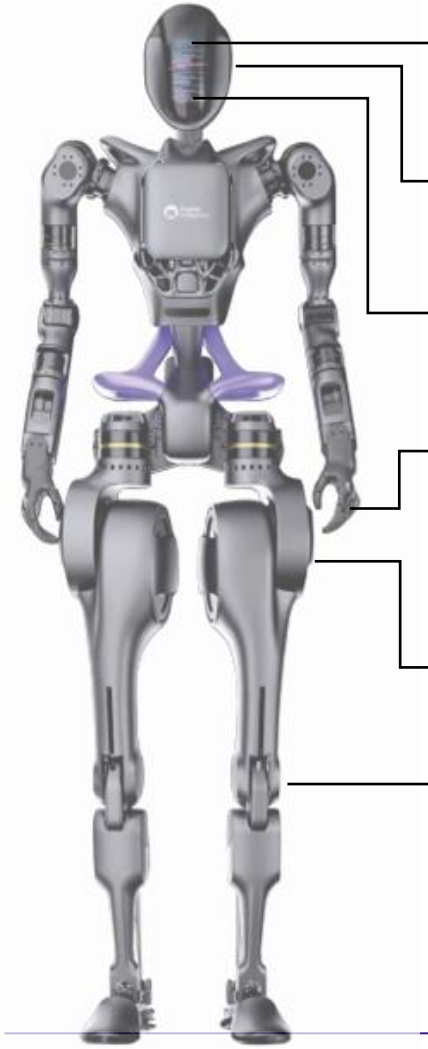

F.02
FIGURE 02
F.02
LAUNCH




MINI 3 PRO


N Mode
Altitude Zone 120 m
95
42'40"
RC
25
1×
AF
0.0 km/h
H 3.0 m
0.0 km/h
D 10 m
ActiveTrack
POI
Storage
02:48:45
RES&FPS
4K30
EV
0.0

연구 배경



카메라/라이다
마이크
후각센서
촉각센서

Sensor Data

Robot AI
Edge AI
Cloud AI

Robot AI
YOLO

Small LLM
Llama 8b
MISTRAL AI_ 7b

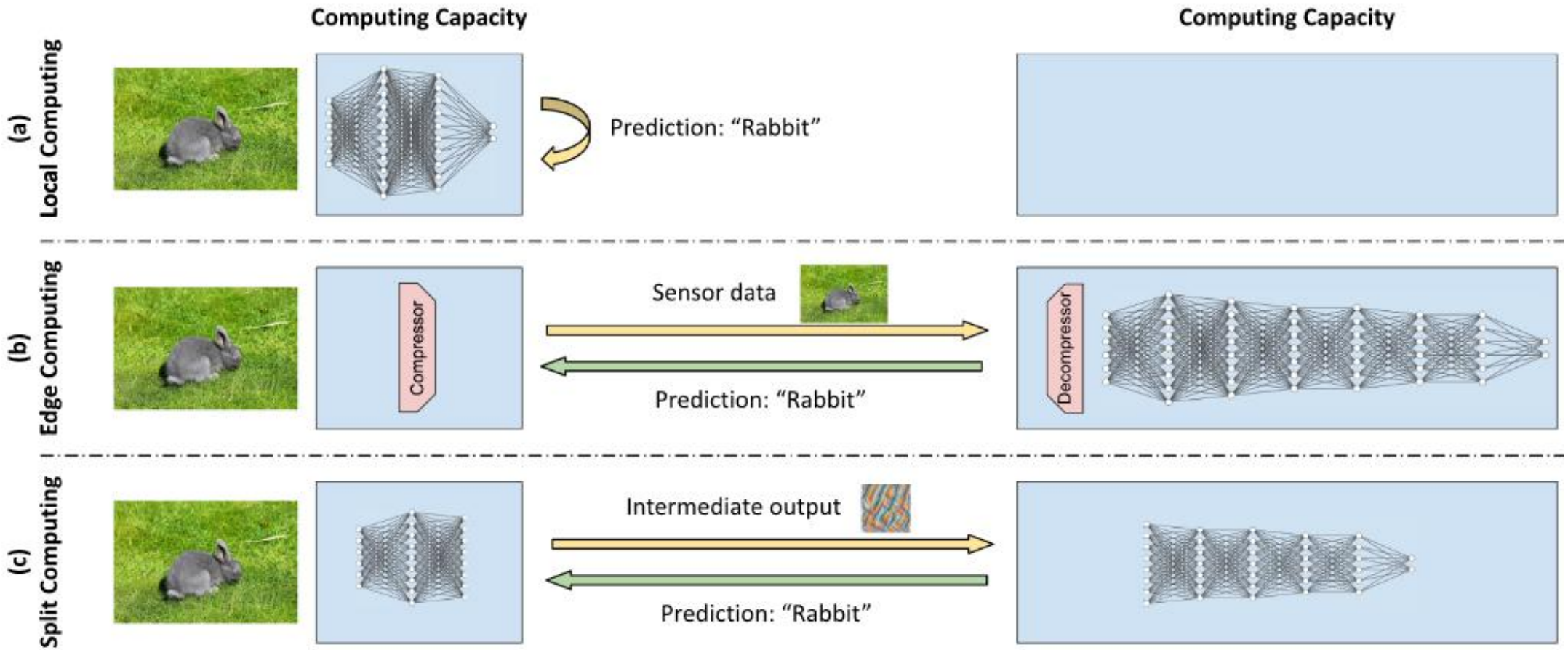
LLM
GPT
Claude
Gemini

Split Computing

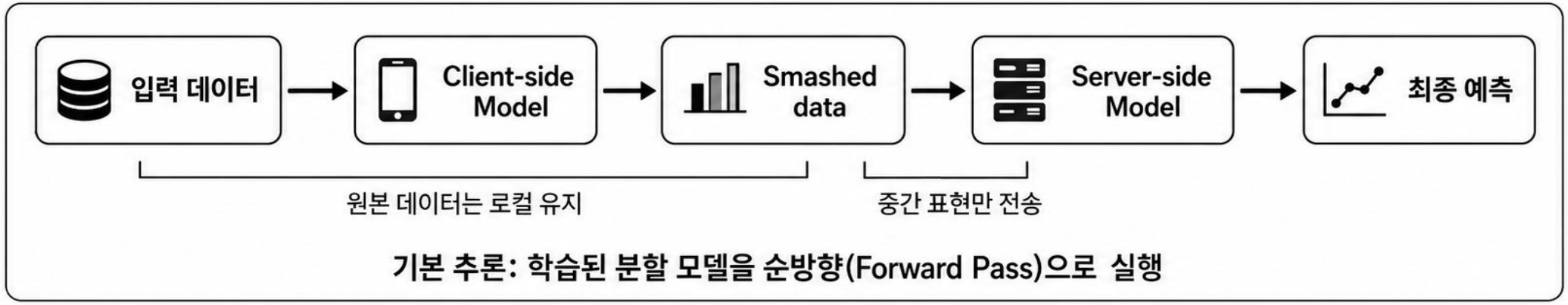
Controller
Robot Control Interface

Actuator 1
Actuator 2
⋮
Actuator n

Docker
Kubernetes
ROS


Computing Capacity
Computing Capacity
(a)
Local Computing
Prediction: “Rabbit”
(b)
Edge Computing
Compressor
Sensor data
Prediction: “Rabbit”
Decompressor
(c)
Split Computing
Intermediate output
Prediction: “Rabbit”

분할 모델 추론 워크플로우



지연(Latency)

- 통신 왕복 시간이 추론 속도에 영향
- 실시간 서비스에서는 응답 지연이 핵심 변수

프라이버시(Privacy)

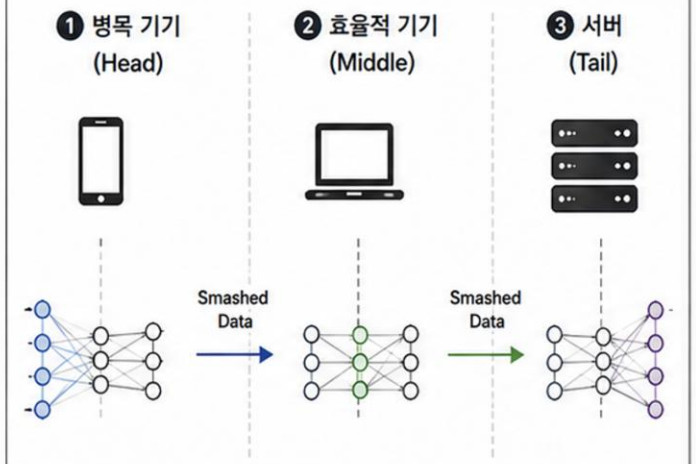
- Smashed data 노출 시 정보 유출 가능
- 추론 단계에서도 보안 방어 필요

협업 확장(Collaboration)

- 협업 추론·안전한 추론 연구 진행
- 멀티모달·LLM 기반 추론으로 확장

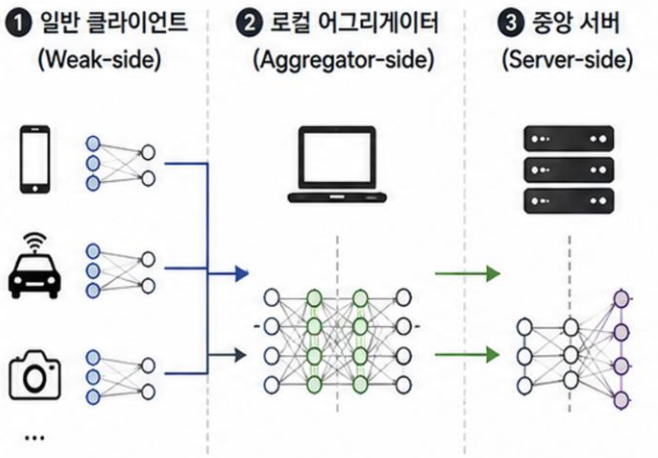
분할 실행 추론 실행 연구

1. 릴레이 협업 추론



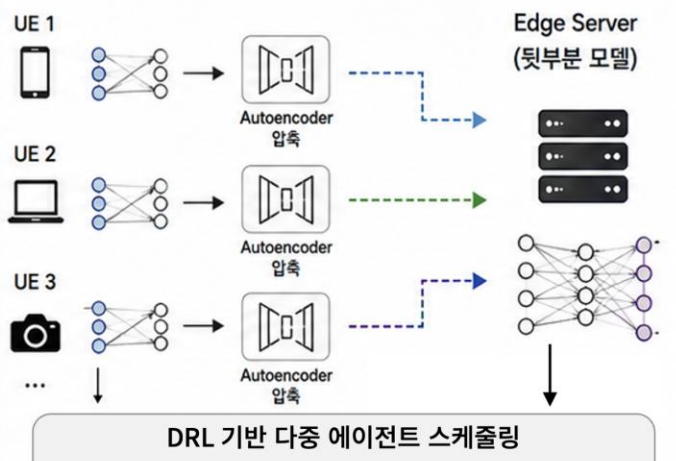
느린 기기를 동료가 도와 전체 추론 지연 최소화

2. 계층형 협업 추론

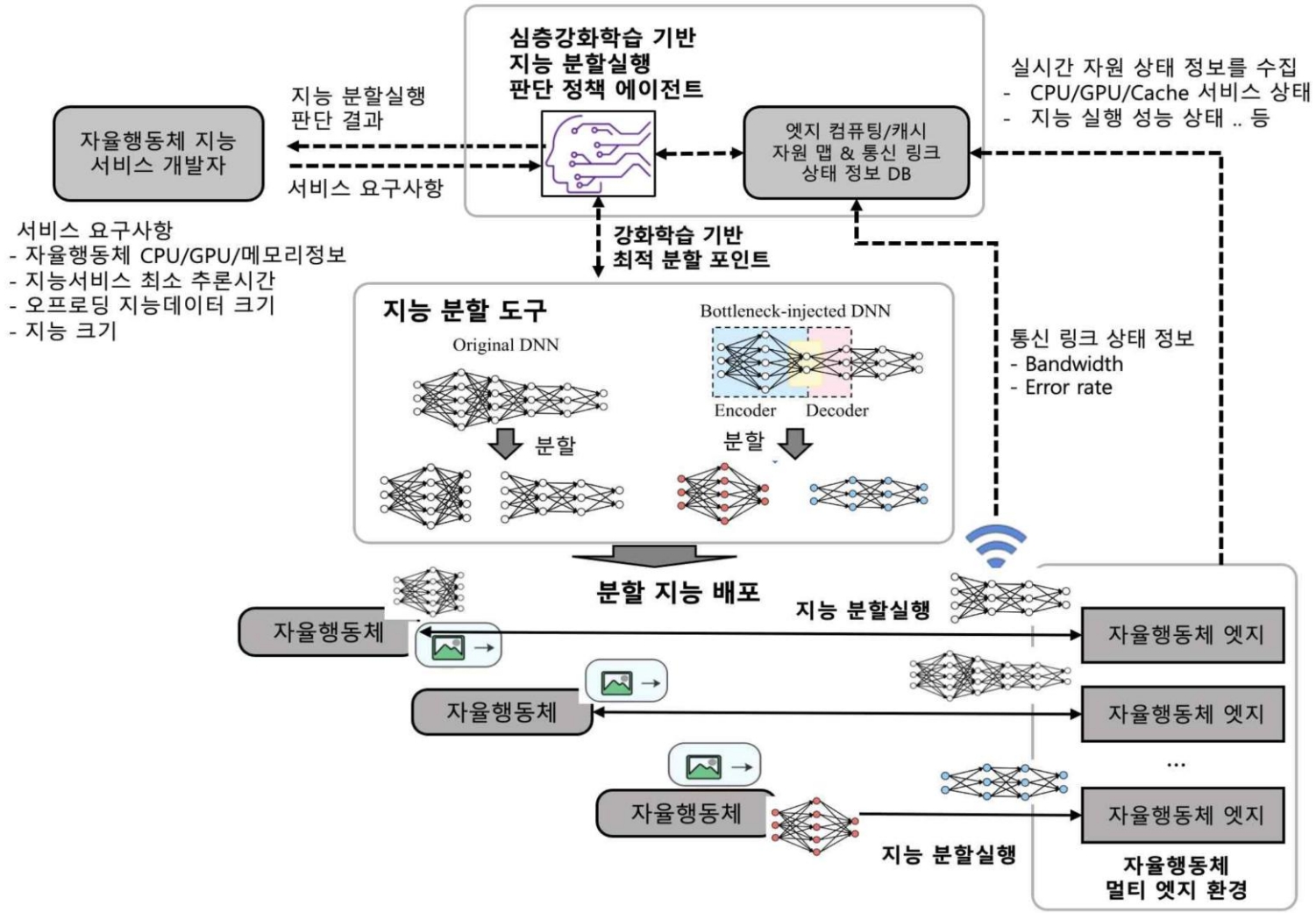


계층적 협업으로 서버 통신 부하 절감

3. 특징 압축 및 DRL 기반 스케줄링



지능 분할실행 판단 & 분할실행 프레임워크



로봇-엣지 간 로봇지능 분할 프레임워크 개발 절차



1. 모델 변환

Torch, TensorFlow, Keras 등으로 작성된 모델을 ONNX 포맷으로 변환



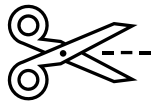
2. 모델 정보 추출

모델의 구조, 레이어별 출력, 텐서 크기 등 모델 분할 시 필요한 정보 추출



3. 분할 후보 지점 추출 및 분할 지점 선정

추출한 모델의 정보를 바탕으로, 분할 후보 지점 추출 및 분할 지점 선정



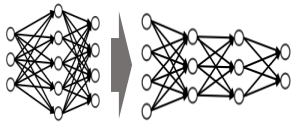
4. 모델 분할

지능 분할실행 판단 및 분할지점 선정 결과에 따라 지능서비스 모델 분할



5. 중간 표현 압축

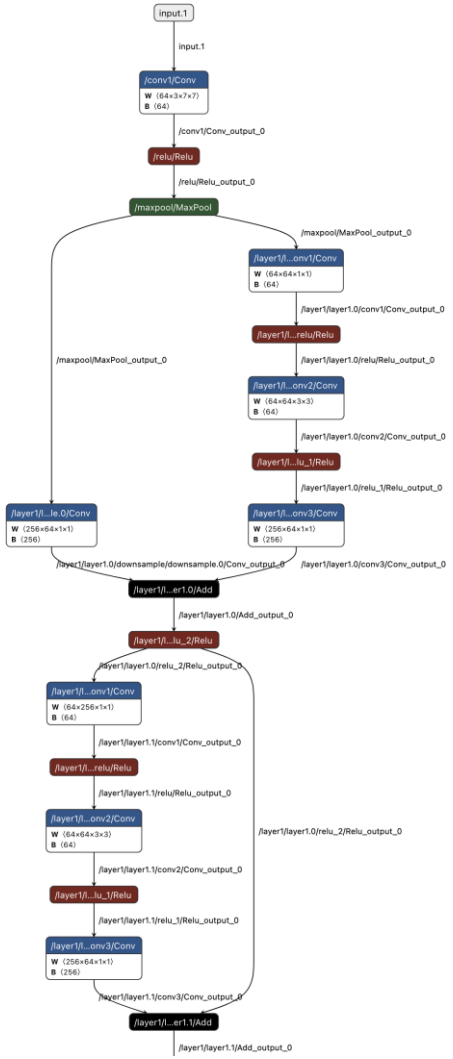
전송 트래픽 감소를 위해 분할된 모델의 중간 표현 데이터 압축



6. 분할 모델 실행

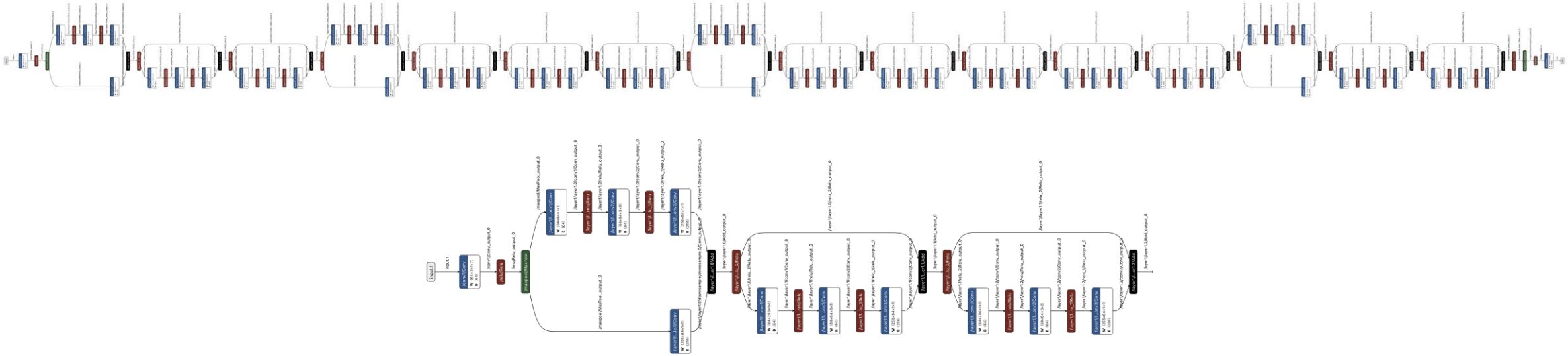
분할된 모델이 서로 다른 디바이스에서 통신하며 실행될 수 있도록 개발

ONNX 변환 및 모델 정보 추출



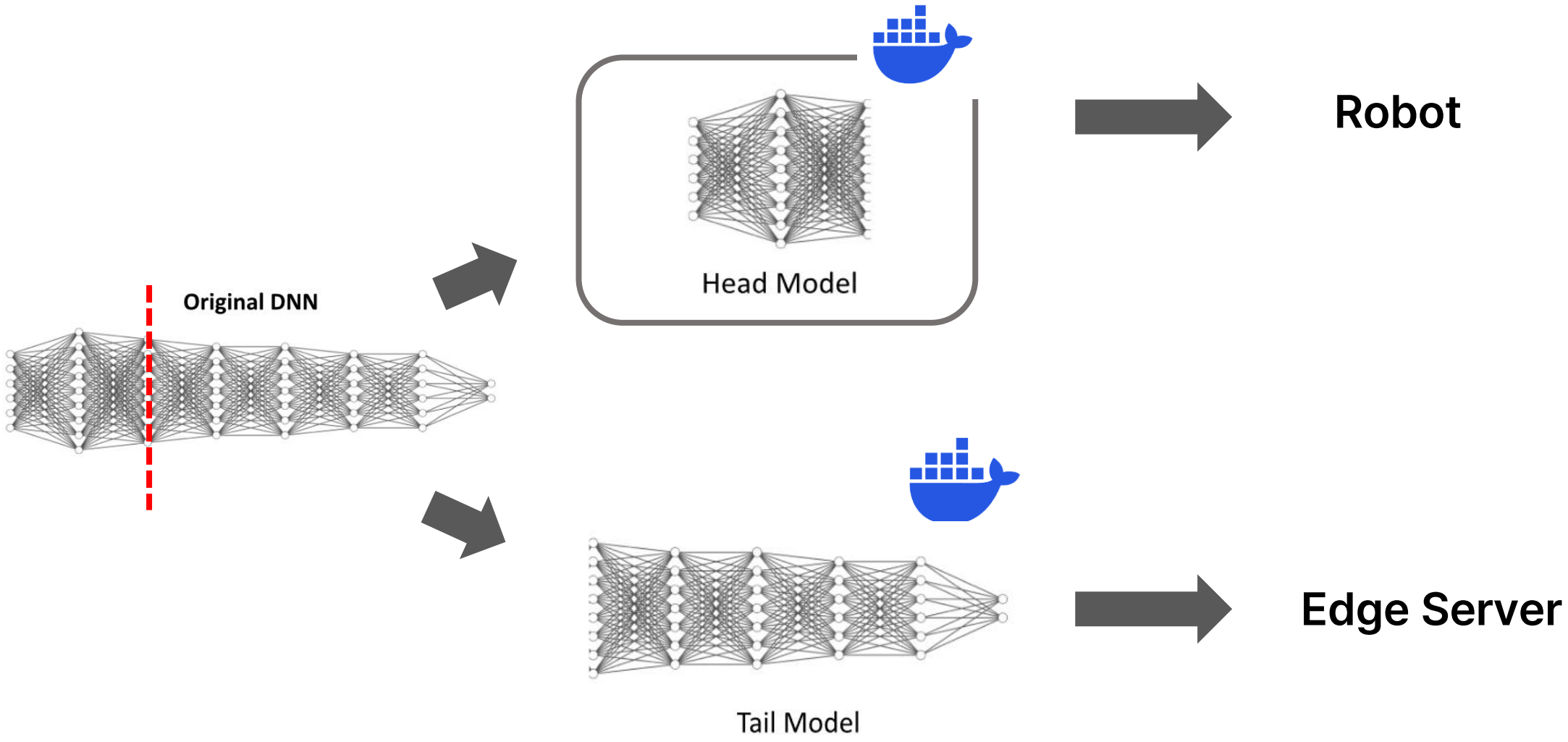

```
----------------------------------------------------------------
  Layer (type)               Output Shape         Param #
================================================================
      Conv2d-1         [-1, 64, 224, 224]           1,792
        ReLU-2         [-1, 64, 224, 224]               0
      Conv2d-3         [-1, 64, 224, 224]          36,928
        ReLU-4         [-1, 64, 224, 224]               0
   MaxPool2d-5         [-1, 64, 112, 112]               0
      Conv2d-6        [-1, 128, 112, 112]          73,856
        ReLU-7        [-1, 128, 112, 112]               0
      Conv2d-8        [-1, 128, 112, 112]         147,584
        ReLU-9        [-1, 128, 112, 112]               0
  MaxPool2d-10          [-1, 128, 56, 56]               0
     Conv2d-11          [-1, 256, 56, 56]         295,168
       ReLU-12          [-1, 256, 56, 56]               0
     Conv2d-13          [-1, 256, 56, 56]         590,080
       ReLU-14          [-1, 256, 56, 56]               0
     Conv2d-15          [-1, 256, 56, 56]         590,080
       ReLU-16          [-1, 256, 56, 56]               0
  MaxPool2d-17          [-1, 256, 28, 28]               0
     Conv2d-18          [-1, 512, 28, 28]       1,180,160
       ReLU-19          [-1, 512, 28, 28]               0
     Conv2d-20          [-1, 512, 28, 28]       2,359,808
       ReLU-21          [-1, 512, 28, 28]               0
     Conv2d-22          [-1, 512, 28, 28]       2,359,808
       ReLU-23          [-1, 512, 28, 28]               0
  MaxPool2d-24          [-1, 512, 14, 14]               0
     Conv2d-25          [-1, 512, 14, 14]       2,359,808
       ReLU-26          [-1, 512, 14, 14]               0
     Conv2d-27          [-1, 512, 14, 14]       2,359,808
       ReLU-28          [-1, 512, 14, 14]               0
     Conv2d-29          [-1, 512, 14, 14]       2,359,808
```

지능 분할 지점 선정

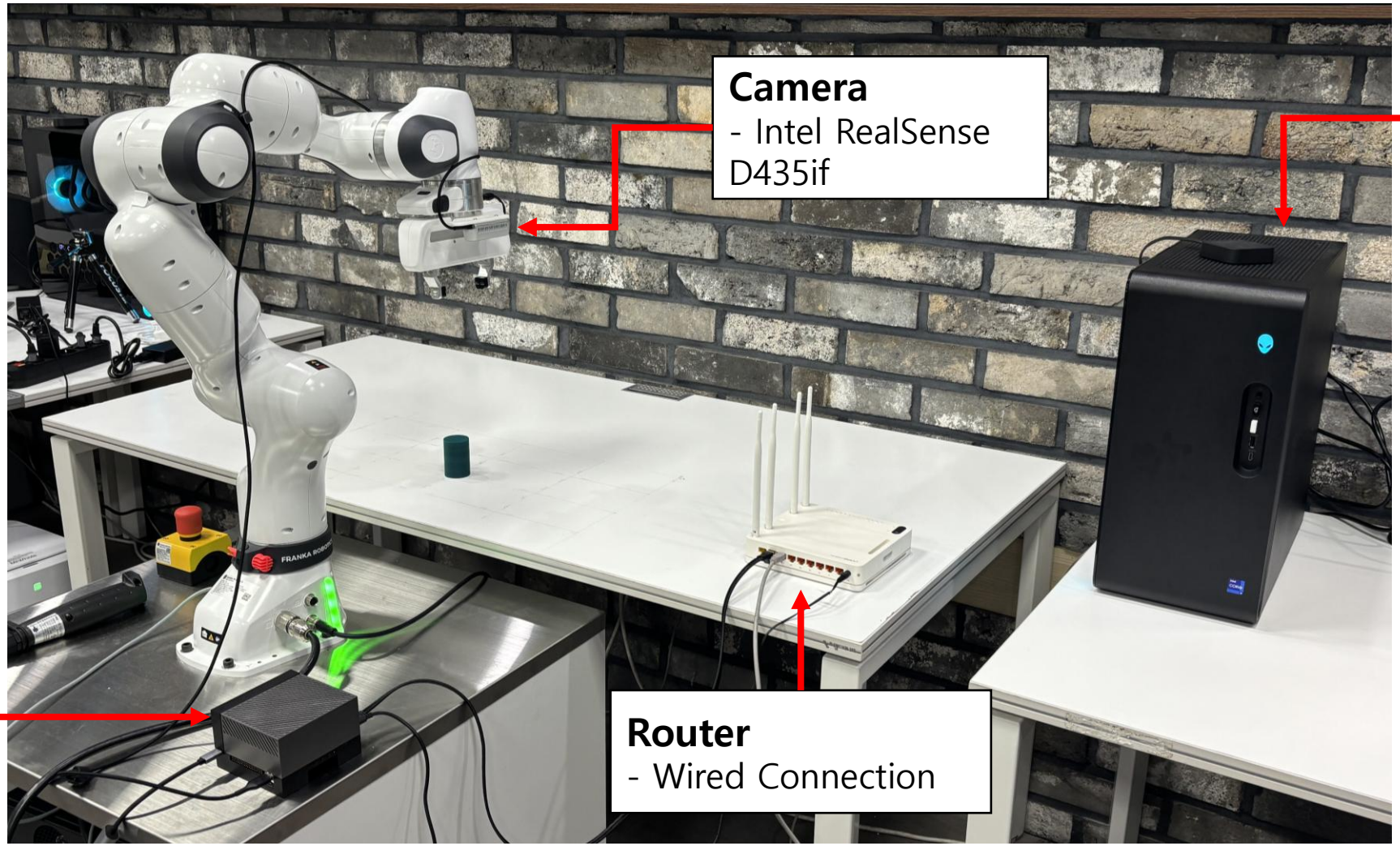


- 지능분할 시험용 모델로는 PyTorch에서 제공하는 ResNet-50 모델 선정
- ResNet-50 모델은 스킵 연결(skip connection)이 포함된 residual block으로 구성되는데, 이 구간 내에서 모델을 분할할 경우 전송량이 크게 증가할 수 있음
- 이를 위해 모델을 분석하여 단일 특징 맵을 전송할 수 있는 분할 지점 후보 추출 기술을 개발

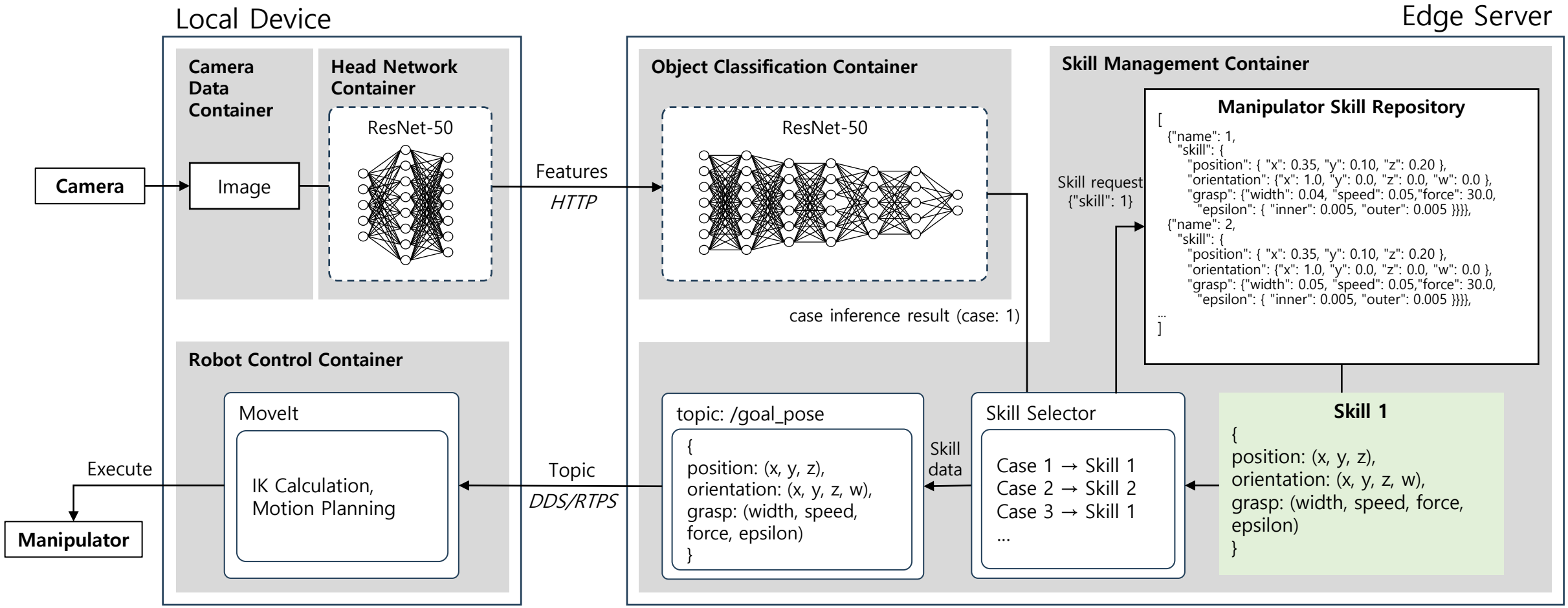
모델 분할 → MS → 배포



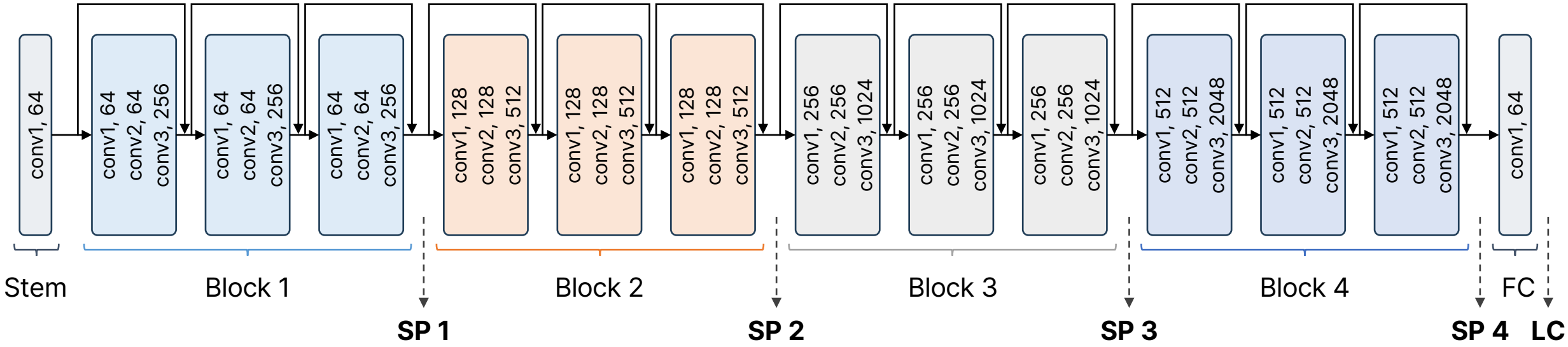
테스트베드 구현


Camera
- Intel RealSense D435if
Edge Server
- i7-14900k
- RTX 4090
Local Device
- Nvidia Jetson AGX Orin
Router
- Wired Connection

지능 분할실행을 활용한 로봇 태스크 수행 시스템 아키텍처

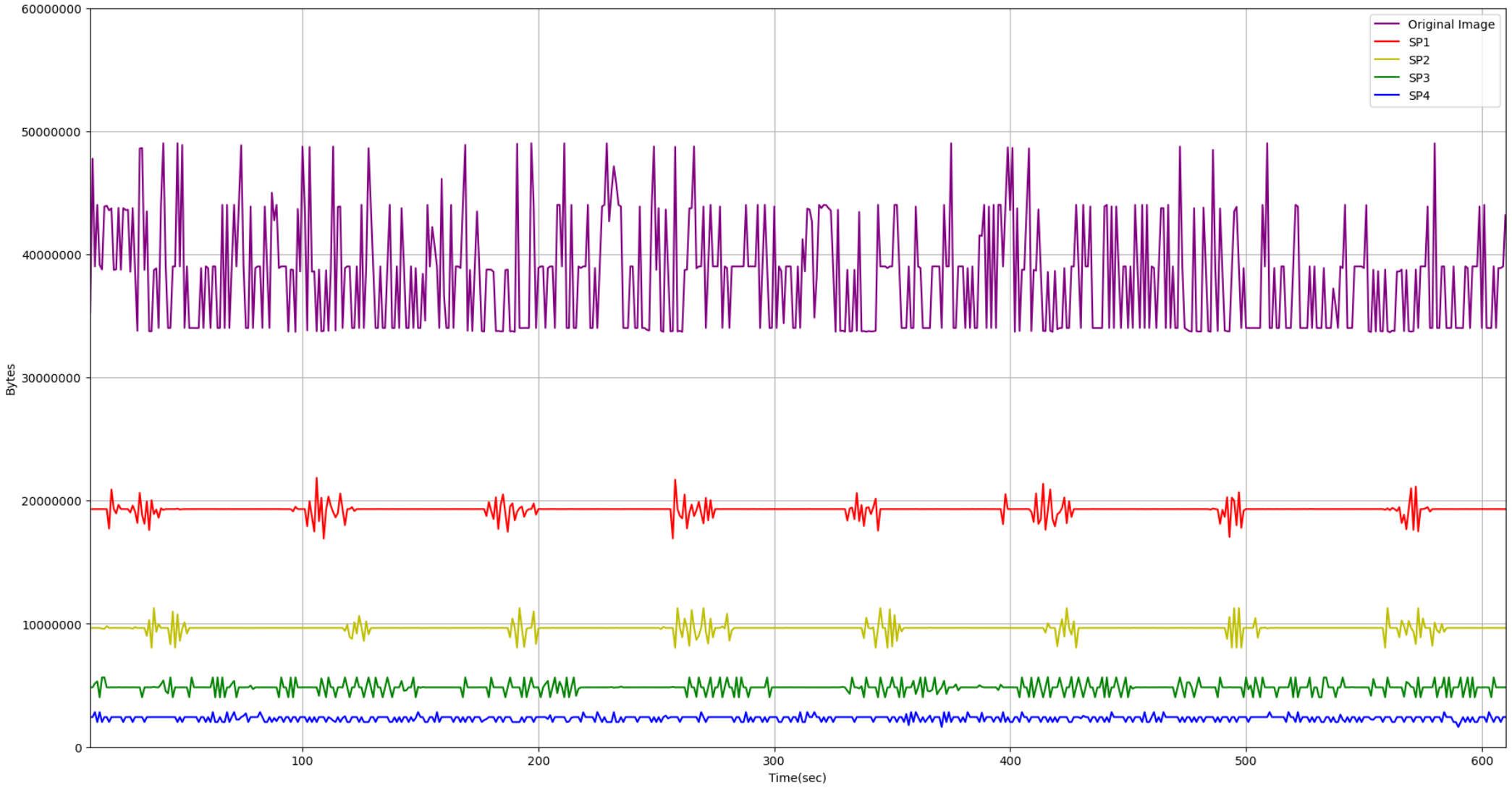


ResNet-50 분할 후보 지점 선정



Split Point	Output Shape	Output Size	Data Size (MB)	FLOPs	Params
SP 1	[1, 256, 56, 56]	802,816	3.16	806,830,080	225,344
SP 2	[1, 512, 28, 28]	401,408	1.53	1,040,048,128	1,219,584
SP 3	[1, 1024, 14, 14]	200,704	0.77	1,472,966,656	7,098,368
SP 4	[1, 2048, 7, 7]	100,352	0.38	811,747,328	14,964,736
LC	[1,1000]	1,000	0.00	2,150,400	2,049,000

SP별 초당 전송량 그래프 (6 FPS)



원본 이미지 대비

SP1 : 49.7% 감소

SP2 : 74.8% 감소

SP3 : 87.4% 감소

SP4 : 93.9% 감소

분할 지점/스킬별 매니퓰레이터 작업 수행

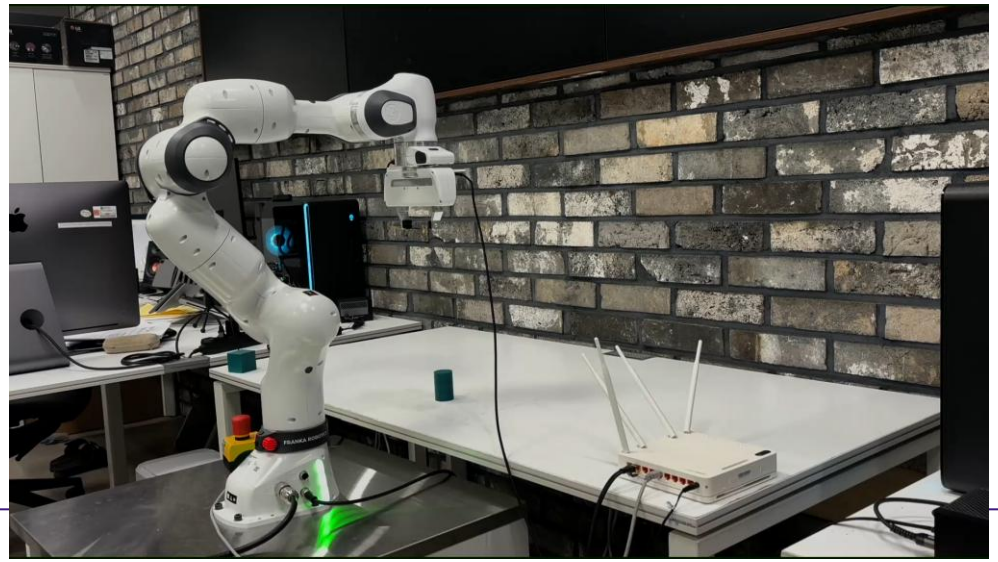
SP: 1
Skill: 1



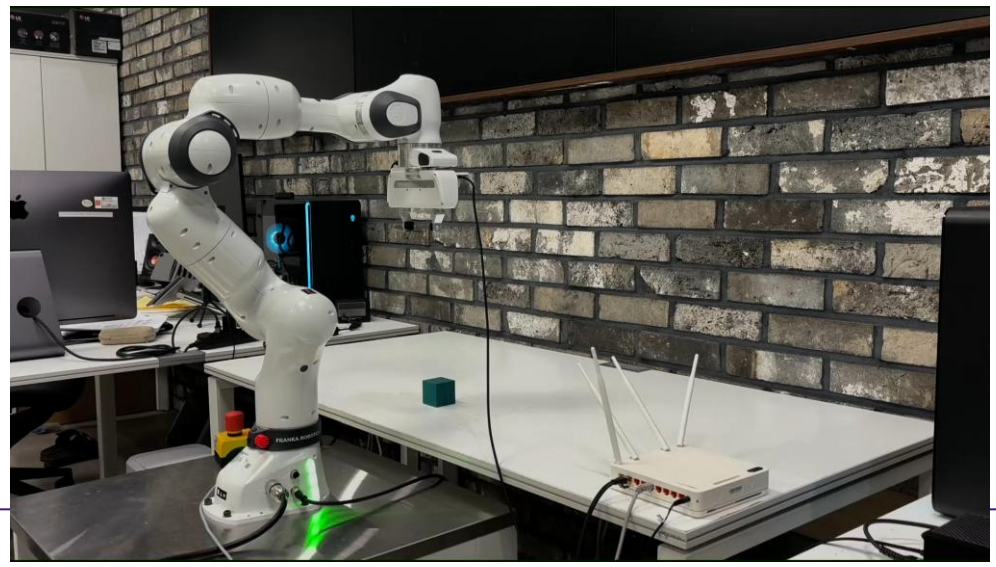
SP: 2
Skill: 2



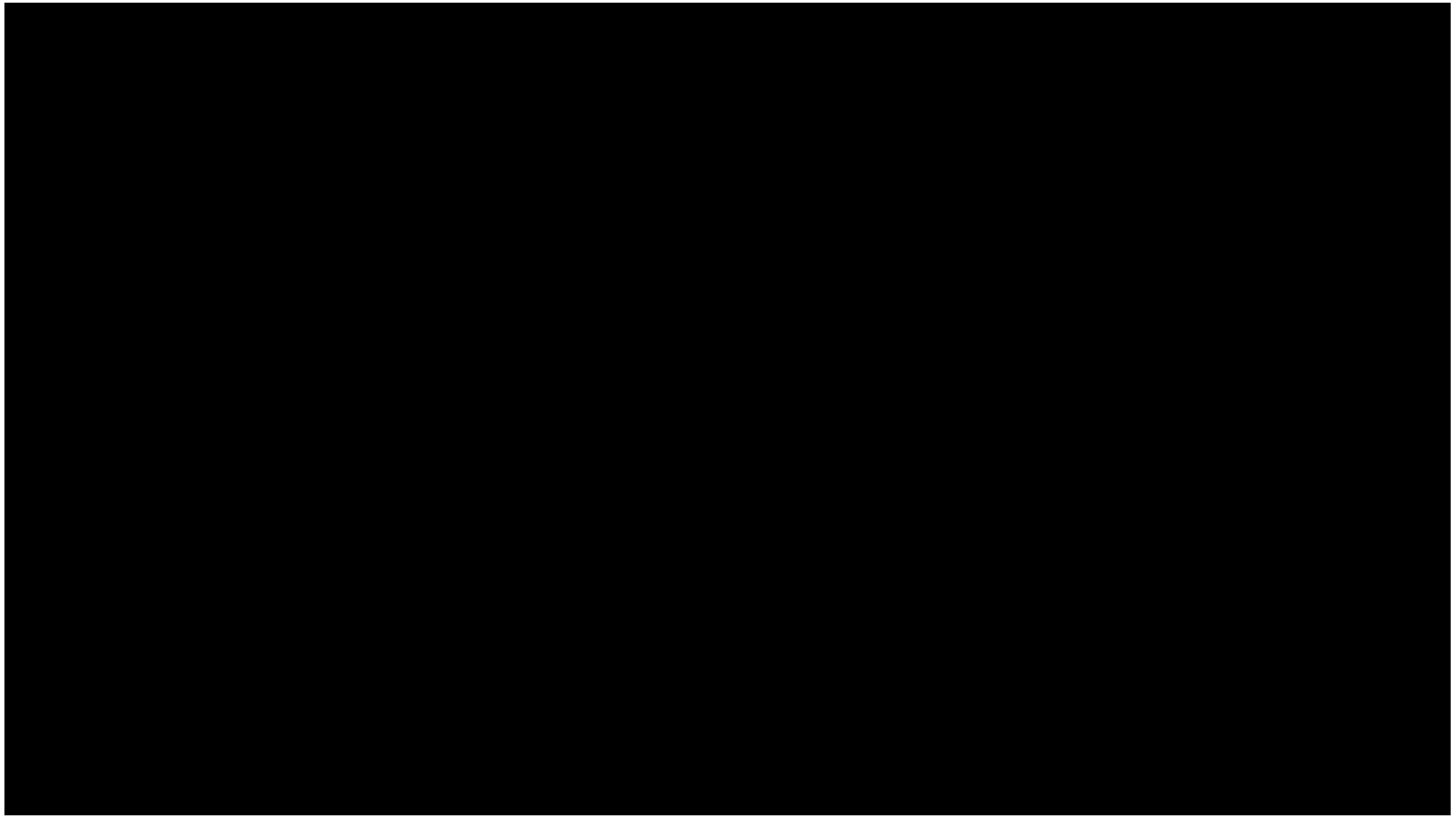
SP: 3
Skill: 3



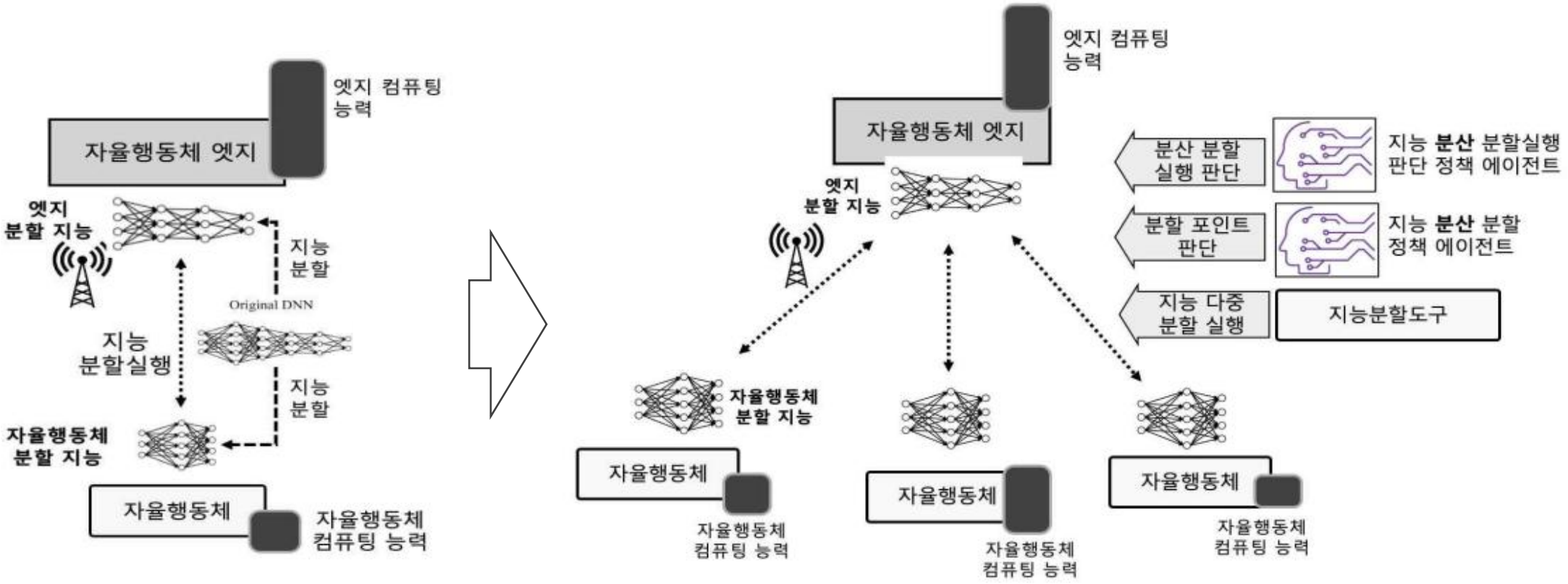
SP: 4
Skill: 4



데모 영상

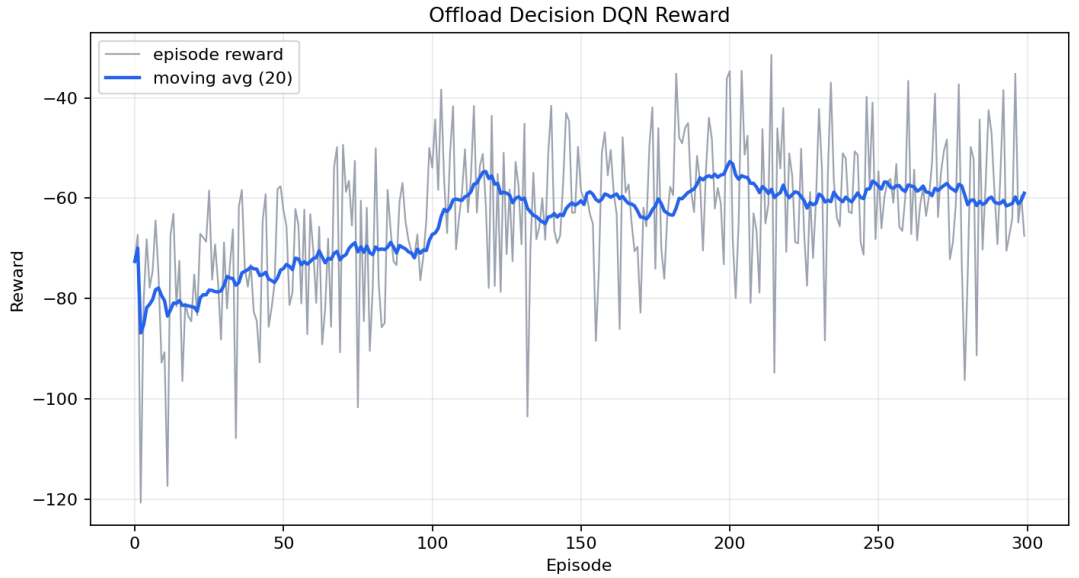


자율행동체 (로봇지능) 분산·분할실행 판단 정책 모델 확장 개발



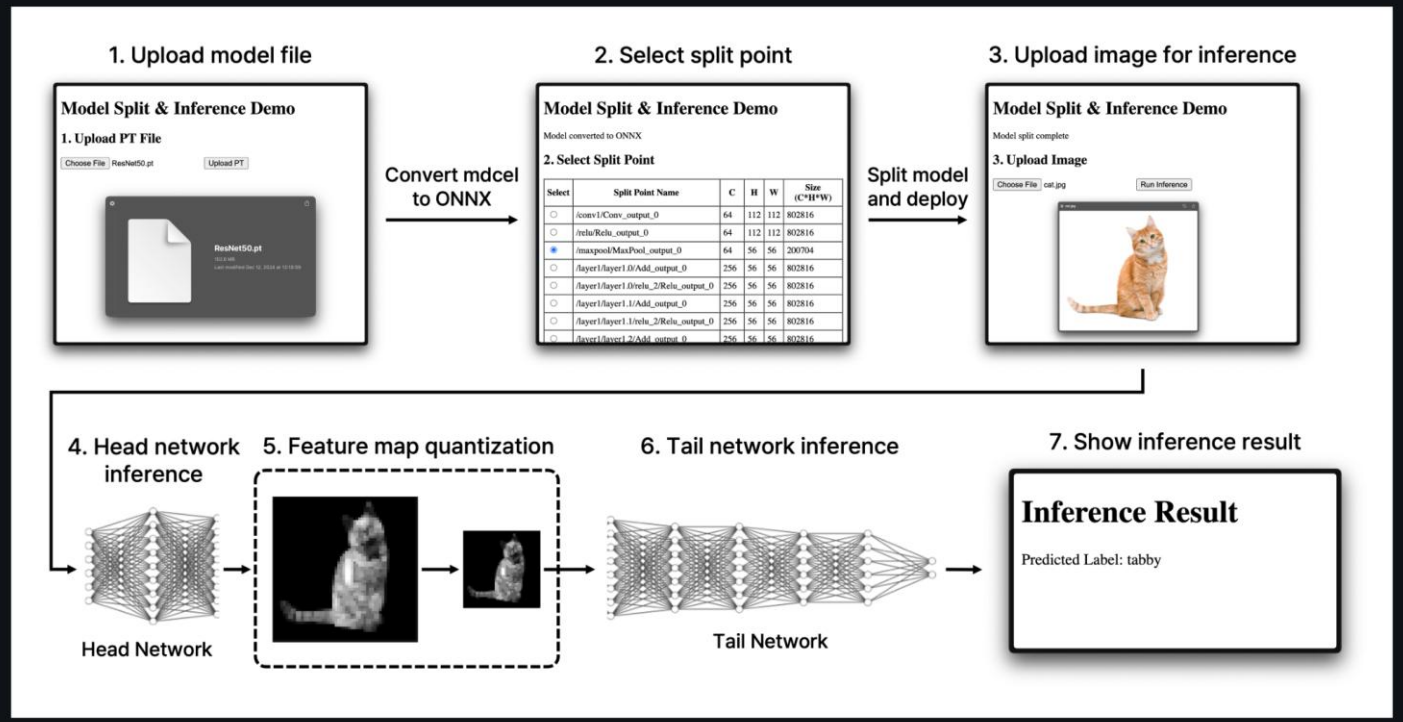
강화학습 기반 분할 지점 선정 정책 성능

Policy	Mean latency	Action distribution
Random	73.9 ms	Full: 50.13%, Split: 49.87%
Full offload only	56.6 ms	Full: 100%
Split computing only	61.5 ms	Split 100%
DQN	**53.8 ms**	Full 44.84%, Split 55.16%



A.EdgeAI-Split-Computing

A.EdgeAI-Split-Computing is a framework designed for the **split execution of deep learning models** between autonomous robots and edge servers. The system optimizes **computational efficiency**, reduces **latency**, and ensures **network traffic minimization** by leveraging distributed AI computing techniques.



A.EdgeAI-Split-Computing / model_splitter / **model_ops.py**

junhodev Create model_splitter/model_ops.py

Code Blame 54 lines (49 loc) · 1.89 KB

```python
import torch
import onnx
import onnx.utils
import onnx.shape_inference

def pt_to_onnx(pt_model_path, onnx_model_path):
    model = torch.load(pt_model_path, map_location='cpu')
    model.eval()
    dummy_input = torch.randn(1, 3, 224, 224)
    torch.onnx.export(model, dummy_input, onnx_model_path)

def get_split_points(model_path):
    model = onnx.load(model_path)
    graph = model.graph
    inputs = set()
    split_points = []
    flag = False
    for i, node in reversed(list(enumerate(graph.node))):
        if flag:
            flag = False
            split_points.append(node.output)
        for input_ in node.input:
            if all(x not in input_ for x in ['onnx::', 'fc.weight', 'fc.bias']):
                inputs.add(input_)
        inputs = inputs.difference(node.output)
        if len(inputs) == 1:
            flag = True
    split_points.reverse()
    return split_points

def infer_output_shapes(model_path, outputs):
    inferred_model = onnx.shape_inference.infer_shapes(onnx.load(model_path))
    name_to_dim = {}
    for v in inferred_model.graph.value_info:
        dims = [d.dim_value for d in v.type.tensor_type.shape.dim]
        name_to_dim[v.name] = dims
    for o in inferred_model.graph.output:
        dims = [d.dim_value for d in o.type.tensor_type.shape.dim]
        name_to_dim[o.name] = dims

    shapes = []
    for out_name in outputs:
        shape = name_to_dim.get(out_name[0], None)
        if shape is None:
            shape = [1,64,56,56]
        shapes.append((out_name[0], shape))
    return shapes

def split_model(model_path, split_point, head_path, tail_path):
    model = onnx.load(model_path)
    input_name = model.graph.input[0].name
    output_name = model.graph.output[0].name
    onnx.utils.extract_model(model_path, head_path, [input_name], [split_point], check_model=True)
    onnx.utils.extract_model(model_path, tail_path, [split_point], [output_name], check_model=True)
```

https://github.com/Open-Edge-Robotics/A.EdgeAI-Split-Computing

10th Anniversary

GISC 2026

Global ICT Standards Conference 2026

- 감사합니다 -

윤주상 교수 동의대학교 물리지능융합연구실

joosang.youn@gmail.com, jsyoun@deu.ac.kr

ICT Standards and Intellectual Property:
Shaping the Physical AI

주최 과학기술정보통신부 지식재산처 주관 국립전파연구원 TTA IITP KEA kista ETRI