종합설계 1

거대무선 채널 기반 미래 채널 예측 및 통신 환경 분류 연구

지도교수님 | 양희철 교수님

컴퓨터융합학부 | 이호윤 | 202002541

인공지능학과 | 김가현 | 202202469

연구 배경

1. 차세대 무선 통신의 기술 트렌드

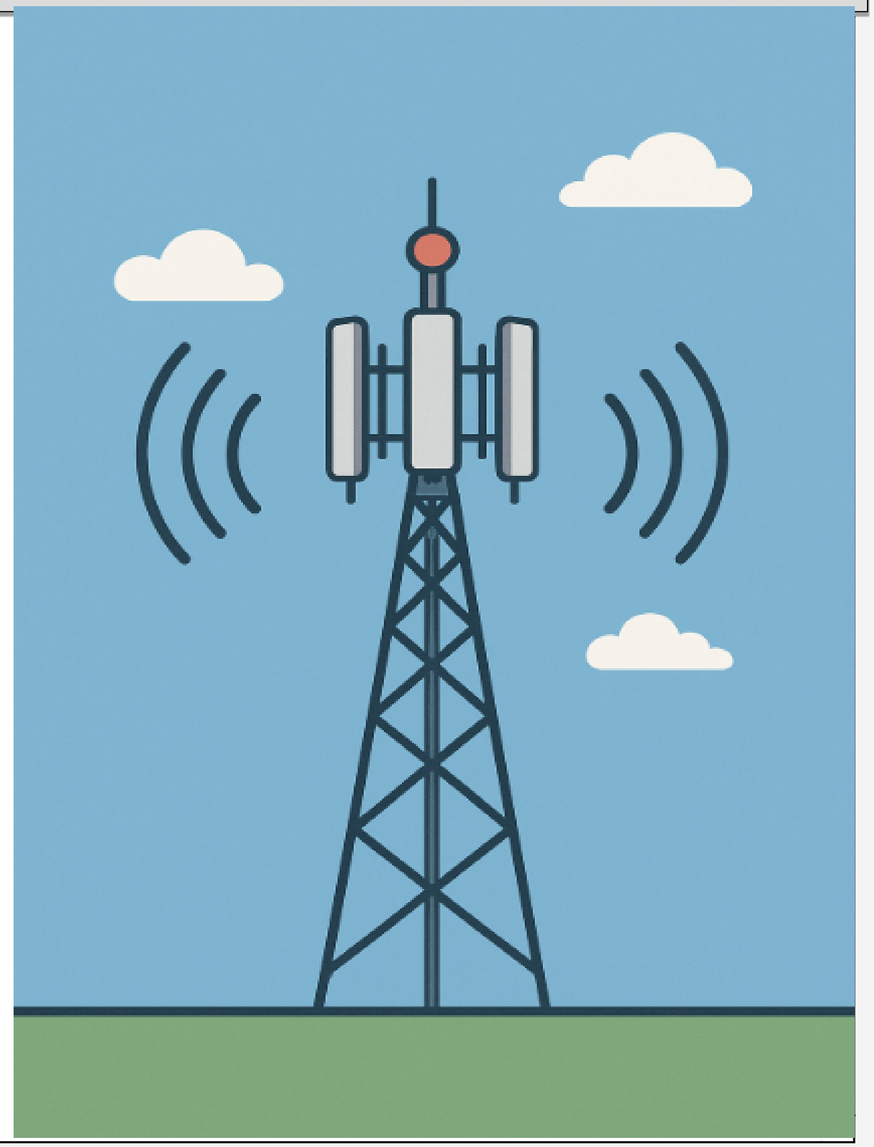
mmWave-sub-THz 대역, Massive MIMO,
네트워크 고밀도화에 따라 초고속, 초저지연 서비스 요구 증가

2. 기존 접근 방식의 한계

고차원 채널 처리와 실시간 자원 관리의 복잡성으로 인해
기존 딥러닝 모델이나 최적화 기법으로는
환경 일반화에 한계를 보인다.

3. Foundation 도입 필요성

LWM은 대규모 채널 시뮬레이션 데이터 기반으로
self-supervised 방식의 transformer를 학습해
다양한 다운스트림 과제를 수행할 수 있다.



연구 목적

$$y = Hx + n$$

LWM 파운데이션 모델을 활용하여 동적 무선 시나리오에서
수신 신호 y = Hx + n의 시변 복소 채널 행렬 H를
정밀하게 예측하는 것이 목적이다

연구 목적

$$y = Hx + n$$

y : 수신신호
X: 송신신호(pilot 신호)
n: 노이즈
H: 채널 행렬

세부단계

1. 데이터 구축

DeepMIMO Dynamic Scenario (3.5 GHz)를 활용하여
시변 복소 채널 학습을 위한 연속 채널 샘플 생성

2. 모델 사전학습 및 전이학습

Masked Channel Modeling (MCM) 방식으로 LWM 사전학습
전이학습을 통해 시변 채널 행렬 H 예측

3. 성능 비교 및 정량 평가

CNN, LSTM 등 기존 방법과 예측 정확도, 처리 지연, 파라미터 수 비교
정량적 비교로 LWM의 효율성 정확성 검증

데이터 구축

scene_0	2025-03-24 오후 6:20	파일 폴더
scene_1	2025-03-24 오후 6:20	파일 폴더
scene_2	2025-03-24 오후 6:22	파일 폴더
scene_3	2025-03-24 오후 6:24	파일 폴더
scene_4	2025-03-24 오후 6:26	파일 폴더
scene_5	2025-03-24 오후 6:29	파일 폴더
scene_6	2025-03-24 오후 6:31	파일 폴더
scene_7	2025-03-24 오후 6:33	파일 폴더

O2_dyn_3p5.1.CIR.BSBS.mat	2025-03-24 오후 6:20	MAT 파일	1KB
O2_dyn_3p5.1.CIR.mat	2025-03-24 오후 6:20	MAT 파일	3,366KB
O2_dyn_3p5.1.DoA.BSBS.mat	2025-03-24 오후 6:20	MAT 파일	1KB
O2_dyn_3p5.1.DoA.mat	2025-03-24 오후 6:20	MAT 파일	3,026KB
O2_dyn_3p5.1.DoD.BSBS.mat	2025-03-24 오후 6:20	MAT 파일	1KB
O2_dyn_3p5.1.DoD.mat	2025-03-24 오후 6:20	MAT 파일	3,040KB
O2_dyn_3p5.1.LoS.BSBS.mat	2025-03-24 오후 6:20	MAT 파일	1KB
O2_dyn_3p5.1.LoS.mat	2025-03-24 오후 6:20	MAT 파일	4KB
O2_dyn_3p5.1.PL.BSBS.mat	2025-03-24 오후 6:20	MAT 파일	1KB
O2_dyn_3p5.1.PL.mat	2025-03-24 오후 6:20	MAT 파일	626KB
O2_dyn_3p5.2.CIR.BSBS.mat	2025-03-24 오후 6:20	MAT 파일	1KB

DeepMIMO

Home Versions Scenarios Applications Forum License

Figure 1. The top view of the Blender design for the ray-tracing scenario

Download a high resolution photo

Download the ray-tracing data files of the O2 scenario:

- 3.5 GHz operating frequency: 'O2_dyn_3p5' scenario (zipped file) or 'O2_dyn_3p5' scenario (folder)
- 3.4 GHz operating frequency: 'O2_dyn_3p4' scenario (zipped file) or 'O2_dyn_3p4' scenario (folder)

데이터 확인

```
{'OFDM': {'RX_filter': 0,
          'bandwidth': 0.05,
          'selected_subcarriers': array([0]),
          'subcarriers': 512},
 'OFDM_channels': 1,
 'active_BS': array([1]),
 'bs_antenna': {'FoV': array([360, 180]),
                'radiation_pattern': 'isotropic',
                'rotation': array([0, 0, 0]),
                'shape': array([8, 4]),
                'spacing': 0.5},
 'dataset_folder': './Raytracing_scenarios',
 'dynamic_scenario_scenes': array([1]),
 'enable_BS2BS': 1,
 'enable_doppler': 0,
 'enable_dual_polar': 0,
 'num_paths': 5,
 'scenario': 'O1_60',
 'ue_antenna': {'FoV': array([360, 180]),
                'radiation_pattern': 'isotropic',
                'rotation': array([0, 0, 0]),
                'shape': array([4, 2]),
                'spacing': 0.5},
 'user_rows': array([1]),
 'user_subsampling': 1}
```

```
# scenario = 02_dyn_3p5 <- 다운받은 파일(동적시나리오)
parameters['scenario'] = '02_dyn_3p5'
parameters['dynamic_scenario_scenes'] = np.arange(10) #scene 0~9

# 각 사용자-기지국 채널에 대해 최대 10개 멀티패스 경로 사용
parameters['num_paths'] = 10

# User rows 1-100
parameters['user_rows'] = np.arange(100)

# Activate only the first basestation
parameters['active_BS'] = np.array([1])

parameters['activate_OFDM'] = 1

parameters['OFDM']['bandwidth'] = 0.05 # 50 MHz
parameters['OFDM']['subcarriers'] = 512 # OFDM with 512 subcarriers
parameters['OFDM']['selected_subcarriers'] = np.arange(0, 64, 1)
#parameters['OFDM']['subcarriers_limit'] = 64 # Keep only first 64 subcarriers

parameters['ue_antenna']['shape'] = np.array([1, 1]) # Single antenna
parameters['bs_antenna']['shape'] = np.array([1, 32]) # ULA of 32 elements
#parameters['bs_antenna']['rotation'] = np.array([0, 30, 90]) # ULA of 32 elements
#parameters['ue_antenna']['rotation'] = np.array([[0, 30], [30, 60], [60, 90]]) # ULA of 32 elements
#parameters['ue_antenna']['radiation_pattern'] = 'isotropic'
#parameters['bs_antenna']['radiation_pattern'] = 'halfwave-dipole'
```

채널 정보 확인

$$y = Hx + n$$

사용자 채널 정보 확인

In [12]:
```
# subcarries = 나눈 각각의 주파수 채널
# Channel = H <- 채널 벡터
# 채널 형태
# (user, UE antenna, Bs antenna, subcarrier)
channel = dataset[0][0]['user']['channel']
print(channel.shape)
```

```
(69040, 1, 32, 64)
```

```
print(dataset[0][0]['user']['channel'][100])
```

```
[[[-4.9276509e-06+6.0179661e-07j -4.7681883e-06+1.3829425e-06j
   -4.4857170e-06+2.1285541e-06j ...  4.3711116e-06-2.4074948e-06j
    3.9297033e-06-3.0764131e-06j  3.3868639e-06-3.6660977e-06j]
  [-4.7825752e-06+1.6998159e-06j -4.4491662e-06+2.4436672e-06j
   -4.0009481e-06+3.1246111e-06j ...  3.8229477e-06-3.3767817e-06j
    3.2333687e-06-3.9455144e-06j  2.5602983e-06-4.4125736e-06j]
  [-4.3861578e-06+2.7614337e-06j -3.8878534e-06+3.4282084e-06j
   -3.2891933e-06+4.0066625e-06j ...  3.0543217e-06-4.2173128e-06j
    2.3400164e-06-4.6522073e-06j  1.5652473e-06-4.9671735e-06j]
  ...
  [-5.9705985e-06+1.8381670e-06j -5.5992200e-06+2.7702260e-06j
   -5.0834296e-06+3.6307945e-06j ...  4.8462462e-06-3.9304277e-06j
    4.1544481e-06-4.6554678e-06j  3.3555179e-06-5.2603964e-06j]
  [-5.3850690e-06+3.0986103e-06j -4.8194606e-06+3.9206407e-06j
   -4.1295657e-06+4.6415066e-06j ...  3.8328362e-06-4.8787911e-06j
    3.0023016e-06-5.4293314e-06j  2.0943648e-06-5.8398036e-06j]
  [-4.5357333e-06+4.1878652e-06j -3.8067144e-06+4.8598408e-06j
   -2.9795442e-06+5.4064294e-06j ...  2.6393784e-06-5.5698706e-06j
    1.7136289e-06-5.9204085e-06j  7.4372792e-07-6.1182200e-06j]]]
```

모델 사전학습 및 전이학습

$$y = Hx + n$$

전체 채널 데이터 = User * Scene * antenna * subcarrier

모델 사전학습 및 전이학습

```
channel_sequence = []

for scene in dataset:
    H = scene[0]['user']['channel'][100]
    H = H.squeeze(0)
    # 채널은 복소수 형태
    H_real = H.real # (32, 64) = (BS antenna, subcarrier)
    H_imag = H.imag
    H_concat = np.concatenate([H_real, H_imag], axis=0)
    channel_sequence.append(H_concat)

# T는 sence
channel_sequence = np.stack(channel_sequence)  # shape: (T, 64, 64) = (real/imag쌍, subcarrier)
```

복소수 채널 실수와 허수 부분 합친 후
시계열 데이터로

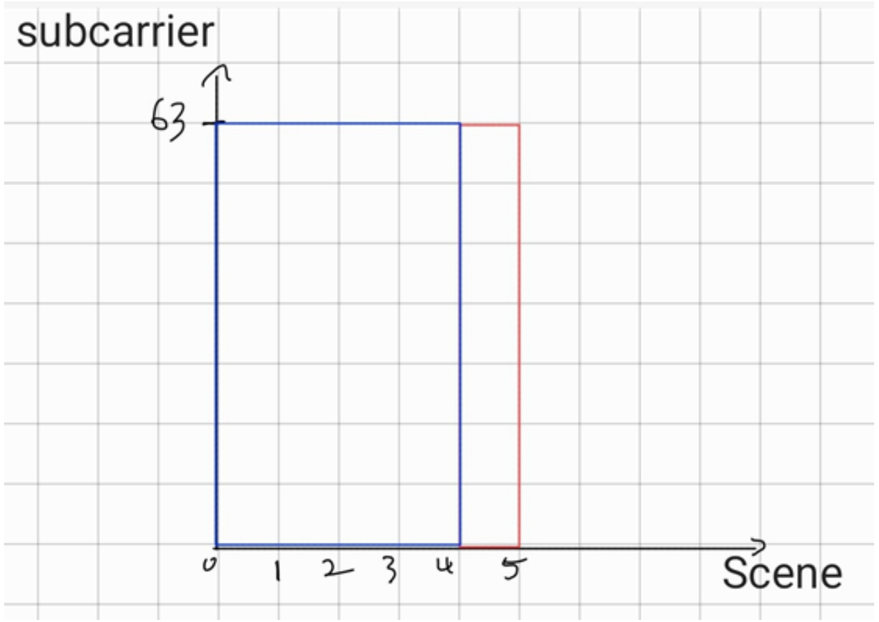
```
# 데이터 만들기
# seq_len = 시퀀스를 잘라서 사용
# [0,1,2,3,4], [1,2,3,4,5], [2,3,4,5,6],[3,4,5,6,7],[4,5,6,7,8],[5,6,7,8,9]
seq_len = 5
X = []
y = []

for i in range(len(channel_sequence) - seq_len):
    input_seq = channel_sequence[i:i+seq_len-1] # t=0~3
    target = channel_sequence[i+seq_len-1] # t = 4

    X.append(input_seq)
    y.append(target)

# T = scene case
# samples = T - seq_len + 1
X = np.array(X) # shape: (samples, seq_len-1, 64, 64)
y = np.array(y) # shape: (samples, 64, 64)
```

앞 4개 training data
뒤 1개 test data

채널 예측 방법(1)

$$y = Hx + n$$

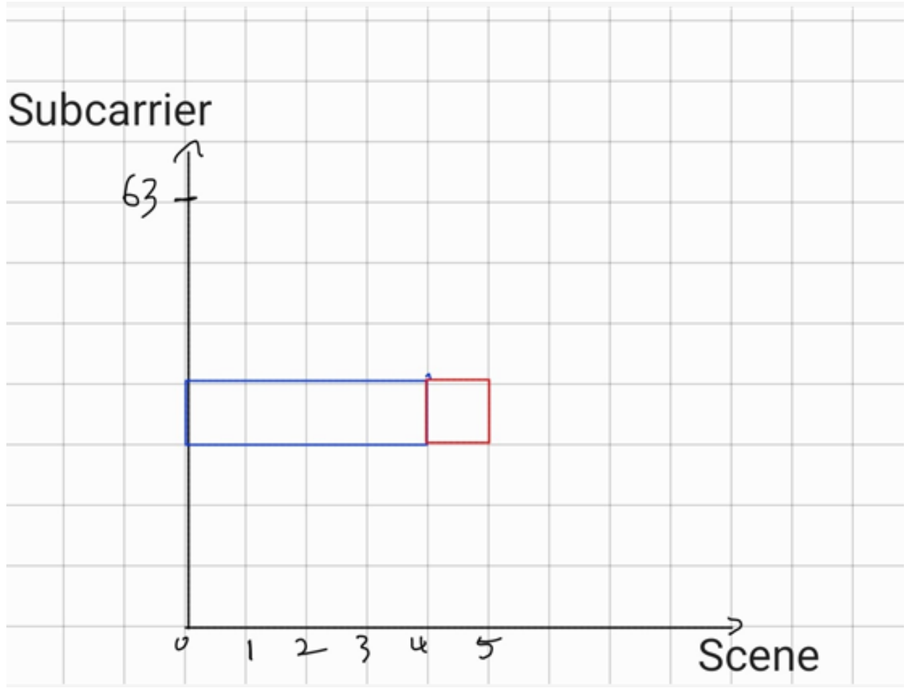
채널 데이터 = user * scene * antenna * subcarrier



채널 예측 방법(2)

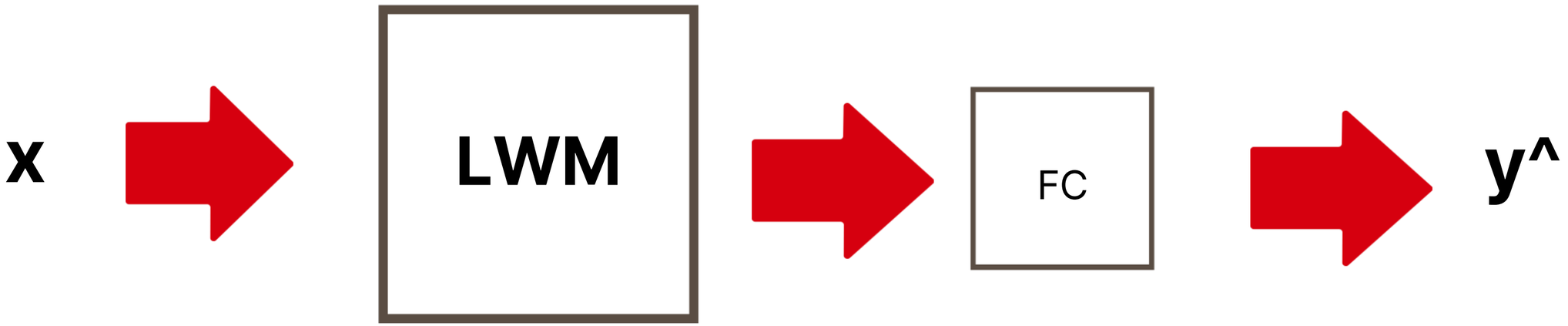
$$y = Hx + n$$

채널 데이터
= user * scene * antenna * subcarrier[1]

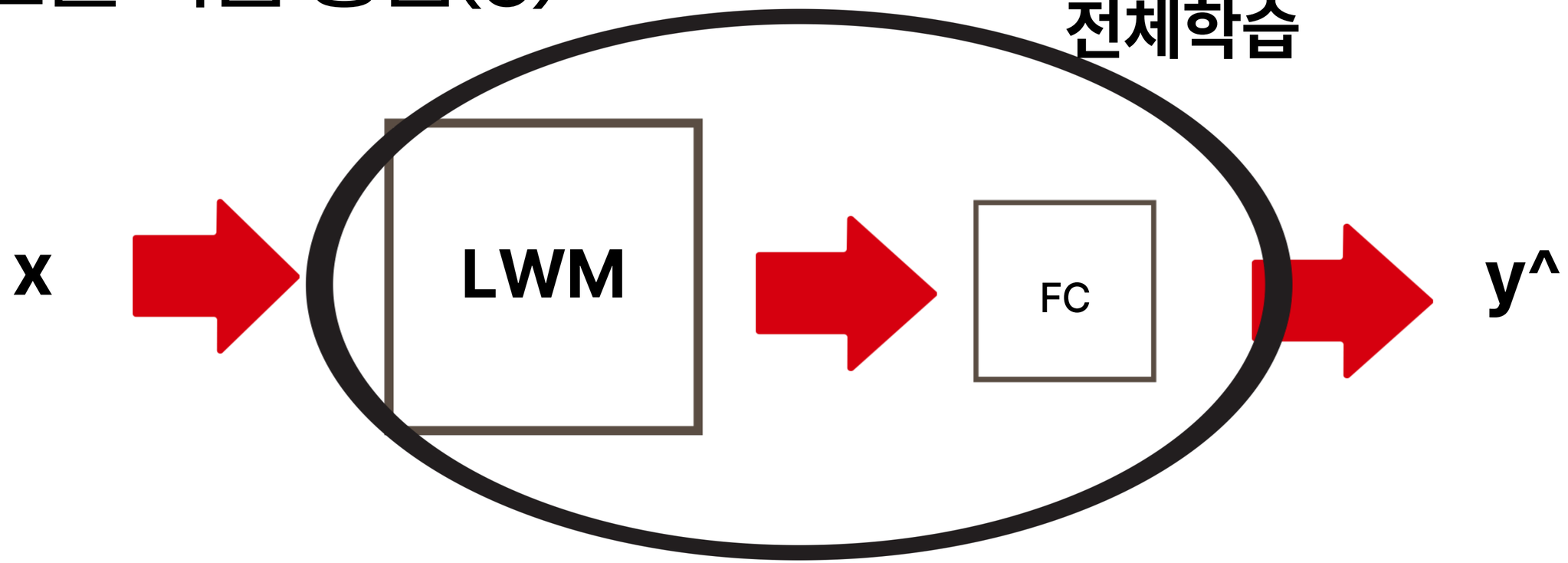


모델 학습 방법(1)

모델 학습 방법(2)



모델 학습 방법(3)



요약

$$y = Hx + n$$

전체 채널 데이터 = User * Scene * antenna * subcarrier

subcarrier를 전체 64개가 아닌 각자로도 볼 수 있다.

LWM에 FC를 추가하는 형식으로 모델을 학습한다.

위의 방식으로 진행한 예측을 CNN, LSTM 등 기존 모델들과 비교하여 성능 평가를 한다.

감사합니다