https://doi.org/10.7840/kics.2024.49.11.1644

블라인드 통신 환경에서 CNN을 활용한 변조 및 채널 코딩 인식과 프로토콜 역공학 시뮬레이션 구현

조 현 우◆, 채 명 호*, 임 완 수°

Implementation of Modulation and Channel Coding Recognition Using CNN and Protocol Reverse Engineering Simulation in Blind Communication Environment

Hyunwoo Cho◆, Myoungho Chae*, Wansu Lim°

요 약

본 논문에서는 송수신기가 통신 제원을 공유하지 않는 블라인드 통신 환경에서 CNN (Convolutional Neural Network)을 활용한 변조 및 채널 코딩 인식과 프로토콜 역공학에 대한 시뮬레이션 구현에 대하여 설명한다. 통신 채널은 AWGN 채널로 가정하였으며 변조 방식으로는 BPSK, QPSK, 8PSK를 사용하고 채널 코딩은 (2, 1, 3), (2, 1, 4), (2, 1, 5) convolutional 코드를 사용한다. 딥러닝 모델 중 하나인 CNN을 사용하여 변조 및 채널 코딩 방식을 인식한다. 또한 프로토콜 역공학 알고리즘인 contiguous sequence pattern 알고리즘을 사용하여 프로토콜 분석을 수행한다. 본 논문의 시뮬레이션은 블라인드 통신 환경을 구현하여 변조 및 채널 코딩 인식, 프로토콜 역공학을 위한 데이터 생성 및 성능 평가 수단으로 활용 가능하다.

키워드 : 변조 인식, 채널 코딩 인식, Convolutional Neural Network (CNN), 프로토콜 역공학, 블라인드 통신
Key Words : modulation recognition, channel coding recognition, Convolutional Neural Network (CNN), protocol reverse engineering, blind communication

ABSTRACT

This paper describes the implementation of the simulation for modulation and channel coding recognition and protocol reverse engineering using CNN (Convolutional Neural Network) in the blind communication environment where transmitters and receivers do not share communication parameters. The communication channel is assumed to be AWGN channel, and BPSK, QPSK, and 8PSK are used as modulation schemes. For Channel coding, (2, 1, 3), (2, 1, 4), and (2, 1, 5) convolutional codes are used. CNN, a type of deep learning model, is utilized to recognize modulation and channel coding schemes. Additionally, the contiguous sequence pattern algorithm, a protocol revers engineering algorithm, is employed to analyze protocols. The simulation in this paper implements the blind communication environment and can be used as a means to generate data and evaluate the performance of modulation and channel coding recognition and protocol reverse engineering.

※ 본 연구는 방위사업청의 재원으로 국방과학연구소의 지원을 받아 수행된 연구임 (311JJ5-912967201).
◆ First Author : Kumoh National Institute of Technology, Department of Electronic Engineering, jm00020@kumoh.ac.kr, 학생회원
° Corresponding Author : Sungkyunkwan University, Electrical and Computer Engineering, wansu.lim@skku.edu, 정회원
* Agency for Defense Development, mhchae4940@naver.com
논문번호 : 202406-123-B-RN, Received June 25, 2024; Revised July 23, 2024; Accepted August 13, 2024

Ⅰ. 서 론

일반적인 통신 환경에서는 송수신기는 사전에 통신 제원을 공유하기 때문에 수신기는 송신기에서 전송한 신호를 복조 및 채널 디코딩을 통하여 정보를 복원할 수 있다. 통신 제원을 공유하는 방법은 신호 대역폭 일부를 통신 제원 등을 전송하는 제어 채널로 할당하거나 신호 일부에 통신 제원에 대한 정보를 담아 전송하는 방법이 있다[1-3]. 이러한 방법들은 대역폭 및 신호 일부를 사용하기에 정보 전송 효율이 감소한다. 따라서 정보 전송 효율을 위해 수신부에서 통신 제원 공유 없이 통신 제원을 추정하는 많은 연구가 진행되고 있다[4,5]. 또한 이러한 연구는 현대의 전자전에서 상대측의 통신 신호를 탈취하고 정보를 습득하기 위해 활용될 수 있다.

송수신기가 통신 제원을 공유하지 않는 블라인드 통신 환경에서 신호에서 정보를 복원하기 위해 변조 방식, 채널 코딩 방식 추정과 프로토콜 역공학을 수행한다. 변조는 신호의 진폭, 위상, 주파수 등을 적절한 형태로 변환하는 것으로 효율적인 정보 전송을 위해 사용된다. 채널 코딩은 통신 과정에서 발생하는 노이즈로 인한 오류를 검출 및 정정하는 기술이다. 변조 및 채널 코딩은 사용한 방법에 따라 복조 및 채널 디코딩 방법이 결정되기 때문에 변조 및 채널 코딩 방식을 알지 못하면 정보를 복원에 어려움이 발생한다. 따라서 블라인드 통신 환경에서 변조 및 채널 코딩 방식 추정이 필요하다. 최근 변조 및 채널 코딩 방식 추정 방법으로 다양한 딥러닝 모델을 사용하는 연구가 진행되고 있다[6-9]. [6]에서는 변조 인식을 위해 CNN (Convolutional Neural Network), LSTM (Long Short-Term Memory), FC (Fully-connected)를 활용한 모델인 MCLDNN (Multi-channel Convolutional Long Short-term Deep Neural Network)을 제안하였고 [7]에서는 ViT (vision transformer) 기반의 VT-MCNet (ViT-based Auto Modulation Classification Network)을 변조 인식에 사용하였다. [8]은 convolutional 코드, LDPC (Low Density Parity Check) 코드, polar 코드를 BiLSTM (Bi-directional LSTM)과 CNN을 결합한 모델인 BiLSTM-CNN으로 분류하였고 [9]는 3가지의 convolutional 코드를 textCNN으로 분류하였다. 복조 및 채널 디코딩을 통해 정보를 복원하여도 사용한 프로토콜을 모르면 필요한 정보 및 정보의 의미를 정확히 파악할 수 없다. 따라서 프로토콜 구조 및 시퀀스 탐색을 수행하는 프로토콜 역공학이 필요하다. 프로토콜 역공학 방법은 실행 트레이스 기반 분석과 네트워크 트레이스 기반 분석으로 분류된다. 실행 트레이스 기반 분석을 사용한 프로토콜 역공학 방법으로 [10]은 산업용 바이너리를 위한 역공학 프레임워크로 사전 지식 데이터베이스를 구축하여 코드시스 바이너리 파일에 대하여 자동화된 분석을 수행한다. [11]은 support vector machine을 이용하여 바이너리의 타입을 분류한다. 그리고 네트워크 트레이스 기반 분석을 사용한 자동 프로토콜 역공학 방법인 [12]는 랜덤 변수와 관측된 메시지의 joint distribution을 이용하여 메시지의 clustering을 수행하고 메시지 형식과 FSM (Finite State Machine)을 추정하고 [13]은 일반적인 시퀀스 탐색 알고리즘과 달리 연속적인 시퀀스만 고려하는 알고리즘이며 반복적인 CSP (Contiguous Sequence Pattern)를 이용하여 추출한 field의 속성을 정의하여 프로토콜의 구조를 확인할 수 있다.

본 논문에서는 통신 제원을 송수신기가 공유하는 일반적인 통신 환경이 아닌 블라인드 통신 환경에서 변조 및 채널 코딩 인식, 프로토콜 분석을 위해 블라인드 통신 환경을 구현한다. 구현한 블라인드 통신 환경에서 딥러닝 모델인 CNN을 사용하여 변조 및 채널 코딩 인식을 수행하며 CSP 알고리즘을 사용하여 프로토콜 분석을 수행한다. 본 논문에서는 송수신기가 통신 제원을 공유하지 않고 데이터 전송 프로토콜도 알지 못하는 환경을 가정하여 송수신기가 서로 아무런 정보를 공유하지 않은 환경을 구현하였다. 따라서 수신기가 수신된 정보를 파악하기 위해서는 변조 및 채널 코딩 인식 후 프로토콜 역공학을 통한 프로토콜 분석 과정이 필요하다. 본 논문의 2장에서는 변조 및 채널 코딩 인식에 사용되는 CNN 모델과 인식 방법에 대하여 설명하고 3장에서는 블라인드 통신 환경에서 변조 및 채널 코딩 인식과 프로토콜 역공학 시뮬레이션을 시뮬레이션 수행 단계에 따라 설명한다. 마지막으로 4장에서는 시뮬레이션 구현에 대한 정리와 활용 방안에 대하여 설명한다.

Ⅱ. 변조 및 채널 코딩 인식 방법

CNN은 딥러닝 모델 중 하나로 특히 이미지 인식 및 컴퓨터 비전 등에서 좋은 성능을 보여주는 모델이다. 그림 1은 간단한 CNN 모델 구조를 보여준다. CNN은 크게 convolutional 레이어와 pooling 레이어를 통하여 데이터의 특징을 추출하고 FC 레이어는 추출된 데이터의 특징으로 이미지 인식 및 분류를 수행한다[14]. 본 논문에서는 변조 인식을 위해 신호의 IQ (In-phase and Quadrature) 데이터를 활용한 성상도를 생성하여 CNN 모델의 입력으로 사용하고 채널 코딩 인식에서는 복조된 데이터를 CNN 모델의 입력으로 사용한다. 성상도

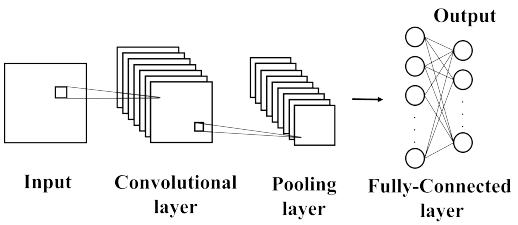


그림 1. 간단한 CNN 모델 구조
Fig. 1. Structure of simple CNN model

는 변조된 신호를 in-phase 성분과 quadrature 성분으로 나누어 2차원 평면에 나타낸 것으로 x축은 in-phase, y축은 quadrature를 의미한다. 주로 PSK (Phase Shift keying), QAM (Quadrature Amplitude Modulation) 변조 신호를 나타낼 때 성상도를 사용한다[21].

2.1 변조 인식

본 논문에서는 BPSK (Binary Phase Shift Keying), QPSK (Quadrature Phase Shift Keying), 8PSK (8-Phase Shift Keying) 3가지의 변조 방식을 CNN으로 인식하는 연구를 수행하였다. IQ 신호를 이용하여 성상도를 그렸을 때 M-PSK 변조 방식은 변조 방식에 따라 다른 성상도를 갖는 특징을 가지고 있다. 따라서 M-PSK 변조 방식의 성상도와 이미지 인식에서 좋은 성능을 보여주는 CNN을 활용하여 변조 분류를 수행하였다. 그림 2는 각 변조 방식에 따른 성상도이다. 그림 2의 (a), (c), (e)는 채널 노이즈가 추가되지 않은 성상도이고 (b), (d), (f)는 AWGN (Additive White Gaussian Noise) 채널 환경에서 SNR (Signal Noise Ratio)이 0dB인 경우의 성상도이다. 각 성상도는 64개의 IQ 데이터를 이용하여 생성하였으며 성상도 크기는 32 × 32 pixel이다. 그림 2의 (b), (d), (f)와 같이 채널 노이즈가 추가된 성상도를 CNN 모델 학습에 사용하였다.

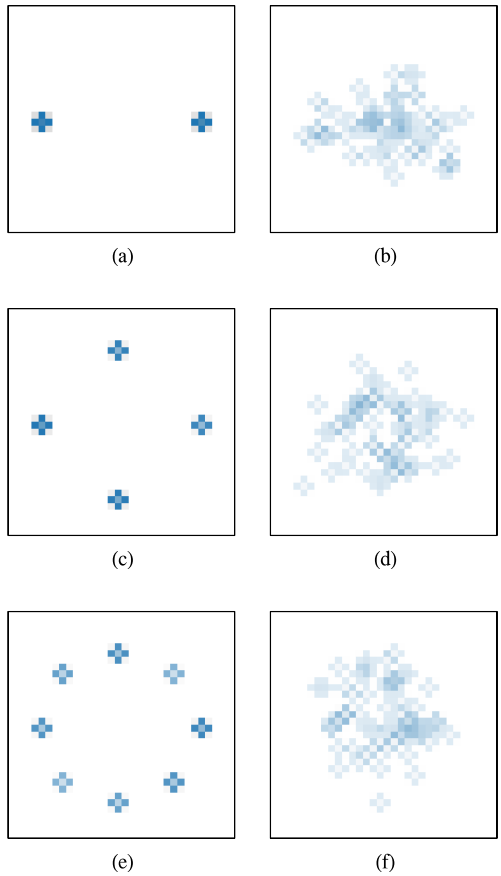


그림 2. 변조 방식에 따른 성상도: 채널 노이즈가 추가 되지 않은 (a) BPSK, (c) QPSK, (e) 8PSK, SNR이 0dB인 경우 (b) BPSK, (d) QPSK, (f) 8PSK
Fig. 2. Constellation diagrams according to modulation schemes: (a) BPSK, (c) QPSK, (e) 8PSK without channel noise, and (b) BPSK, (d) QPSK, (f) 8PSK with SNR of 0dB

표 1. 변조 인식을 위한 CNN 모델 구조
Table 1. CNN model architecture for modulation recognition

Layer	Output shape	Parameters
Input	32 × 32 × 3	0
Conv2D	32 × 32 × 8	224
MaxPooling2D	16 × 16 × 8	0
Conv2D	16 × 16 × 16	1,168
MaxPooling2D	8 × 8 × 16	0
Conv2D	8 × 8 × 32	4,640
MaxPooling2D	4 × 4 × 32	0
Flatten	512	0
Dense	100	51,300
Dense	10	1,010
Output	3	33

Total parameters: 58,375

표 1은 변조 인식을 위한 CNN 모델 구조를 보여준다. 변조 인식을 위한 CNN 모델은 이미지 특징 추출을 위한 3개의 Conv2D 레이어와 MaxpPoling2D 레이어로 구성되어 있고 2개의 Dense 레이어와 Output 레이어를 통하여 변조 방식을 인식하도록 구성되어 있다. 표 1의 Conv2D와 MaxPooling2D는 각각 convolutional 레이어와 maxpooling 레이어로 이미지와 같은 2차원 데이터에서 주로 사용된다. Conv2D의 커널 크기는 3 × 3이고 stride는 1이며 MaxPooling2D의 커널

크기는 2 × 2이고 stride는 2이다. Flatten은 다차원의 데이터를 1차원의 데이터로 변형하는 레이어고 Dense는 FC 레이어이다. Input은 입력 레이어고 Output은 FC 레이어이며 최종적으로 분류를 수행한다. Convolutional 레이어와 Dense 레이어의 활성화 함수는

$$f_{\mathrm{ReLU}}(x)=\max(0,x) \tag{1}$$

인 ReLU (Rectified Linear Unit) 함수를 사용하고 output 레이어의 활성화 함수는

$$f_{\mathrm{softmax}}(x)=\frac{e^{x_j}}{\sum_{k=1}^{K}e^{x_k}} \quad \text{for } j=1,\cdots,k \tag{2}$$

인 softmax 함수를 사용하였다[15, 16]. 변조 인식을 위한 CNN 모델 학습을 위한 loss 함수는 분류 모델에 많이 사용되는

$$f_{\mathrm{cross-entropy}}=-\sum_{i=1}^{N}y_i\log(\hat{y}_i) \tag{3}$$

인 cross-entropy 함수를 사용하였고 optimizer로는 Adam (Adaptive Moment Estimation)을 사용하여 학습을 진행하였다[17]. Adam은

$$\begin{aligned} m_t&=\beta_1 m_{t-1}+(1-\beta_1)\nabla_\theta \boldsymbol{J}(\theta)\\ v_t&=\beta_2 v_{t-1}+(1-\beta_2)\nabla_\theta(\boldsymbol{J}(\theta))^2 \end{aligned} \tag{4}$$

를 통해 기울기의 평균 m_t과 v_t분산을 계산한다. θ는 CNN 모델의 가중치이고 $\boldsymbol{J}(\theta)$는 loss 함수이다. β_1와 β_2는 하이퍼 파라미터로 변화량을 조절한다. 또한 Adam은

$$\theta=\theta-m_t\frac{\eta}{\sqrt{v_t+\epsilon}} \tag{5}$$

를 통하여 CNN 모델의 가중치를 갱신한다. ϵ는 0으로 나누어지는 것을 방지하고 수치 안정성을 유지하기 위한 하이퍼 파라미터이고 η는 learning rate이다[18]. 학습에 사용한 데이터는 AWGN 채널을 가정하여 SNR이 0dB인 64개의 IQ 데이터를 사용하여 그림 2와 같은 100,000개의 성상도를 생성하였다. 따라서 총 학습 데이터는 변조 종류 × 100,000 = 3 × 100,000 = 300,000개의 데이터를 사용하였다.

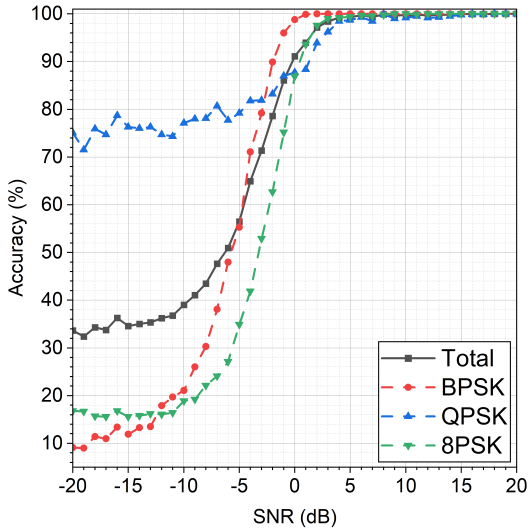
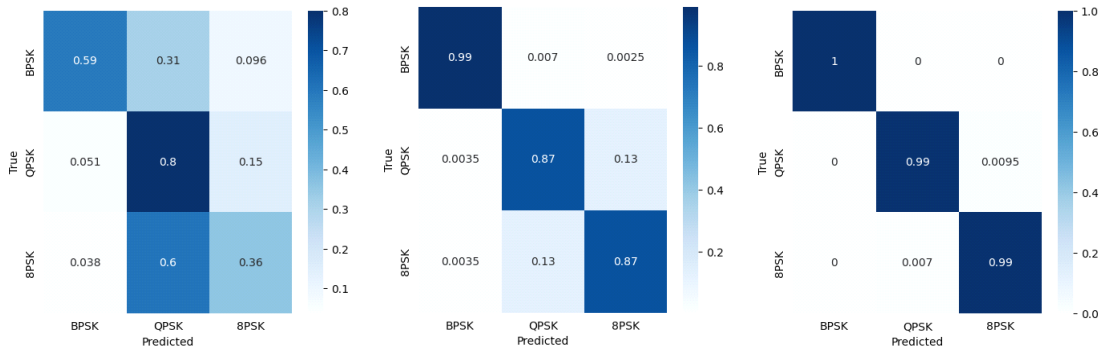


그림 3. 변조 인식을 위한 CNN 모델 성능
Fig. 3. Performance of the CNN model for modulation recognition

그림 3은 변조 인식을 위한 CNN 모델 성능 그래프이다. 그림 3의 성능을 구하기 위해 각 SNR당 변조 방식마다 1,000개의 성상도 데이터를 사용하였다. 그림 3의 검은 실선은 3가지 변조 방식에 대한 정확도를 보여주고 빨간 점선은 BPSK, 파란 점선은 QPSK, 초록 점선은 8PSK에 대한 정확도를 보여준다. 그림 3에서 볼 수 있듯이 변조 인식을 위한 CNN 모델은 0dB의 SNR에서 변조 분류 정확도가 약 90%이고 5dB이상에서는 100%에 근접한 성능을 보여준다. 또한 QPSK의 경우, 0dB보다 작은 SNR에서도 70% 이상의 정확도는 보이지만 BPSK와 8PSK는 비교적 낮은 정확도를 보여준다. 그림 4는 SNR이 −5, 0, 5dB일 때 변조 인식을 위한 CNN 모델의 혼동 행렬을 보여준다. 그림 4의 (a)와 그림 3에서 변조 방식에 따른 정확도를 비교하였을 때, 낮은 SNR 구간에서는 BPSK와 8PSK를 QPSK로 많이 혼동하는 것을 확인할 수 있으며 그림 4의 (b)와 (c)를 보면 SNR이 커져도 QPSK와 8PSK를 혼동하는 것을 확인할 수 있다.

그림 5는 표 1의 CNN 모델에서 첫 번째 Dense 레이어의 output shape를 50, 100, 200으로 한 CNN 모델들의 성능을 비교한 그래프이다. 첫 번째 Dense 레이어의 output shape가 50인 경우, 총 파라미터 수는 32,225이고 200인 경우는 110,675개이며 100인 경우는 표 1에서와 같이 58,375개이다. 그림 5에서 볼 수 있듯이 3개



(a) (b) (c)

그림 4. 변조 인식을 위한 CNN 모델 혼동 행렬: (a) SNR = -5dB, (b) SNR = 0dB, (c) SNR = 5dB
Fig. 4. Confusion matrix of the CNN model for modulation recognition: (a) SNR = -5dB, (b) SNR = 0dB, (c) SNR = 5dB

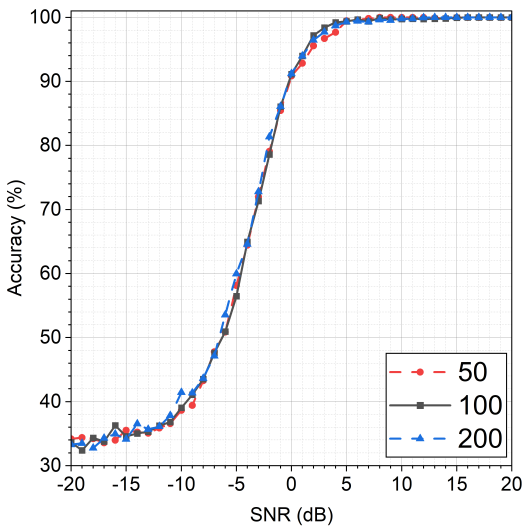


그림 5. CNN에서 첫 번째 Dense 레이어의 output shape에 따른 변조 인식 성능
Fig. 5. Modulation recognition performance based on the output shape of the first Dense layer in CNN

의 모델은 유의미한 성능 차이를 보여주지 않지만 모델의 파라미터 수가 많은 모델일수록 좋은 성능을 보여주는 것을 확인할 수 있다. 본 논문에서는 3가지 모델 중 모든 구간에 양호한 성능을 보여주며 파라미터 수가 너무 많지 않은 첫 번째 Dense 레이어의 output shape가 100인 CNN 모델을 사용한다.

2.2 채널 코딩 인식

본 논문에서는 다양한 채널 코딩 방법 중에서 3가지 종류의 convolutional 코드를 사용하였다. Convolutional 코드는 블록 코드와 달리 전송하는 데이터의 길이가 유동적으로 변형될 수 있는 코드로 메모리를 사용하여 이전 입력 비트들과의 연산을 통해 인코딩을 수행한다. Convolutional 코드는 (n, k, L) convolutional 코드로 표현할 수 있으며 n은 코드 워드 길이, k는 입력 비트 길이, L은 구속장(constraint length)으로 $m+1$이며, m은 메모리 수이다. 본 논문에서는 (2, 1, 3), (2, 1, 4), (2, 1, 5) convolutional 코드를 사용한다. Convolutional 코드는 메모리 수만큼의 이전 입력이 현재 출력에 영향을 주는 특징을 갖는 채널 코딩 방법으로 다른 채널 코딩 방법과 달리 인접한 코드 워드간 높은 연관성을 갖는다. 인접한 데이터를 고려하는 대표적인 딥러닝 모델로 RNN (Recurrent Neural Network)과 CNN이 있다. CNN은 커널 크기에 따라 인접한 데이터와의 연산을 수행하기 때문에 비교적 작은 계산복잡도로 RNN처럼 인접한 데이터를 활용할 수 있다. 따라서 채널 코딩 인식을 위한 딥러닝 모델로 CNN 모델을 사용하고 AWGN 채널 환경과 hard decision 복조를 고려하였다.

표 2는 채널 코딩 인식을 위한 CNN 모델 구조를 보여준다. 채널 코딩 인식에서는 변조 인식에서처럼 이미지가 아닌 복조 신호를 사용하여 모델의 입력이 1차원 데이터이다. Convolutional 레이어를 사용하기 위해 Reshape 레이어로 입력 데이터의 차원을 변형한다. Conv1D와 MaxPooling1D는 1차원 형태의 데이터를 위한 convolutional 레이어와 pooling 레이어이다. Conv1D의 커널 크기는 7이고 stride는 1이며 MaxPooling1D의 커널 크기는 2이고 stride는 2이다. Convolutional 레이어와 Dense 레이어의 활성화 함수는 ReLU 함수를 사용하였고 Output 레이어의 활성화 함수는 softmax이다. 채널 코딩 인식을 위한 CNN 모델 학습을 위한 loss 함수는 cross-entropy 함수, optimizer

표 2. 채널 코딩 인식을 위한 CNN 모델 구조
Table 2. CNN model architecture for channel coding recognition

Layer	Output shape	Parameters
Input	64	0
Reshape	64 × 1	0
Conv1D	64 × 16	128
MaxPooling1D	32 × 16	0
Conv1D	32 × 32	3,616
MaxPooling1D	16 × 32	0
Conv1D	16 × 64	14,400
MaxPooling1D	8 × 64	0
Flatten	512	0
Dense	100	51,300
Dense	10	1,010
Output	3	33

Total parameters: 70,487

로는 Adam을 사용하여 학습을 진행하였다. 학습에 사용한 데이터는 AWGN 채널을 가정하여 SNR이 0dB인 64비트의 길이를 갖는 복조 신호 1,000,000개를 생성하여 사용하였다. 따라서 총 학습 데이터는 채널 코딩 종류 × 1,000,000 = 3 × 1,000,000 = 3,000,000개의 데이터를 사용하였다.

그림 6은 채널 코딩 인식을 위한 CNN 모델 성능 그래프이다. 그림 6의 성능을 구하기 위해 각 SNR당 채널 코딩 방식마다 10,000개의 64비트 코드 워드를 사용하였다. 그림 6의 검은 실선은 3가지 채널 코딩 방식에 대한 정확도를 보여주고 빨간 점선은 (2, 1, 3), 파란 점선은 (2, 1, 4), 초록 점선은 (2, 1, 5) convolutional 코드에 대한 정확도를 보여준다. 채널 코딩 인식을 위한 CNN 모델은 0dB의 SNR에서 채널 코딩 분류 정확도가 약 60%이고 5dB이상에서는 100%에 근접한 성능을 보여준다. 채널 코딩 인식 정확도는 변조 인식의 경우와는 달리 3가지 채널 코딩 방식에 따른 정확도 차이가 크지 않지만 (2, 1, 5), (2, 1, 4), (2, 1, 3) convolutional 코드 순으로 높은 정확도를 보여준다. 그림 7은 SNR이 -5, 0, 5dB일 때 변조 인식을 위한 CNN 모델의 혼동 행렬을 보여준다. 그림 7의 (a)와 (b)에서 볼 수 있듯이 낮은 SNR에서는 (2, 1, 3), (2, 1, 4) convolutional 코드를 (2, 1, 5) convolutional 코드로 많이 혼동하고 그림 7의 (c)에서는 (2, 1, 4), (2, 1, 5) con-

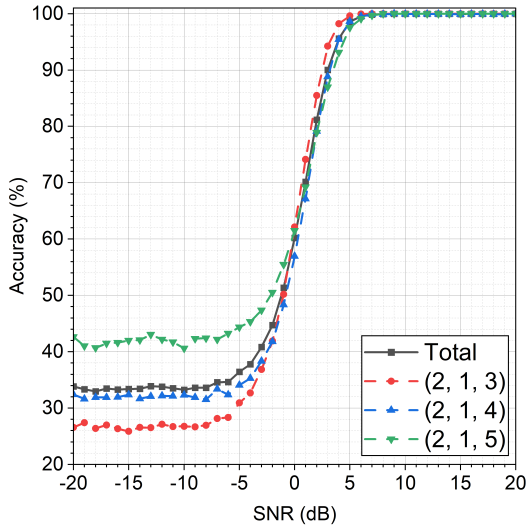
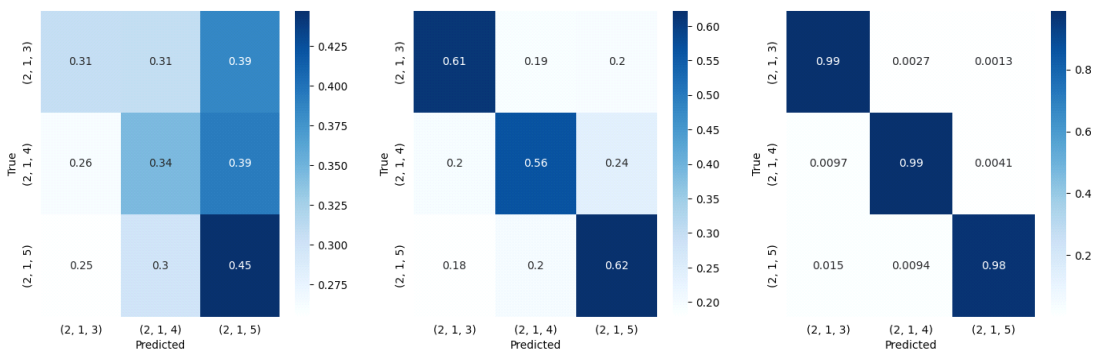


그림 6. 채널 코딩 인식을 위한 CNN 모델 성능
Fig. 6. Performance of the CNN model for channel coding recognition



(a) (b) (c)

그림 7. 채널 코딩 인식을 위한 CNN 모델 혼동 행렬: (a) SNR = -5dB, (b) SNR = 0dB, (c) SNR = 5dB
Fig. 7. Confusion matrix of the CNN model for channel coding recognition: (a) SNR = -5dB, (b) SNR = 0dB, (c) SNR = 5dB

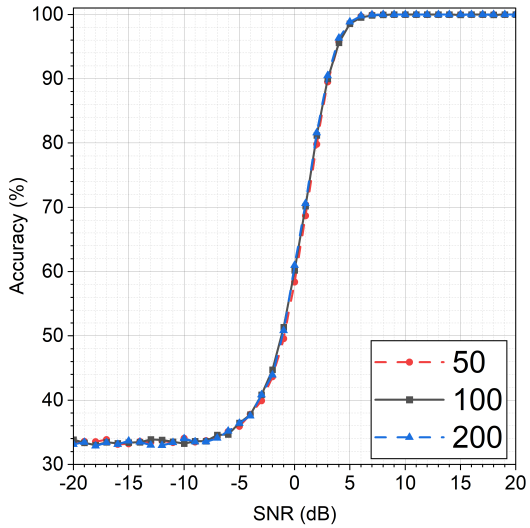


그림 8. CNN에서 첫 번째 Dense 레이어의 output shape에 따른 채널 코딩 인식 성능
Fig. 8. Channel coding recognition performance based on the output shape of the first Dense layer in CNN

volutional 코드를 (2, 1, 3) convolutional 코드로 혼동한다.

그림 8은 표 2의 CNN 모델에서 첫 번째 Dense 레이어의 output shape를 50, 100, 200으로 한 CNN 모델들의 성능을 비교한 그래프이다. 첫 번째 Dense 레이어의 output shape가 50인 경우, 총 파라미터 수는 44,337이고 200인 경우는 122,787개이며 100인 경우는 표 2에서와 같이 70,487개이다. 변조 인식과 마찬가지로 그림 5에서 볼 수 있듯이 3개의 모델은 유의미한 성능 차이를 보여주지 않았다. 본 논문에서는 3가지 모델 중 모든 구간에 양호한 성능을 보여주며 파라미터 수가 너무 많지 않은 첫 번째 Dense 레이어의 output shape가 100인 CNN 모델을 사용한다.

Ⅲ. 변조 및 채널 코딩 인식과 프로토콜 역공학 시뮬레이션

그림 9는 본 논문에서 구현한 시뮬레이션 overview를 보여준다. 블라인드 통신 환경에서 CNN을 활용한 변조 및 채널 코딩 인식 시뮬레이션은 크게 송신부, 수신부로 나누어진다. 송신부는 송신 데이터 생성, 채널 인코딩, 변조를 수행하고 수신부는 변조 인식과 복조, 채널 코딩 인식과 채널 디코딩을 수행하고 수신 데이터에 대하여 프로토콜 분석을 수행한다. 또한 본 논문에서는 변조 및 채널 코딩 인식과 프로토콜 역공학 결과 분석을 위해 다양한 통신 채널 모델 중 페이딩, 간섭

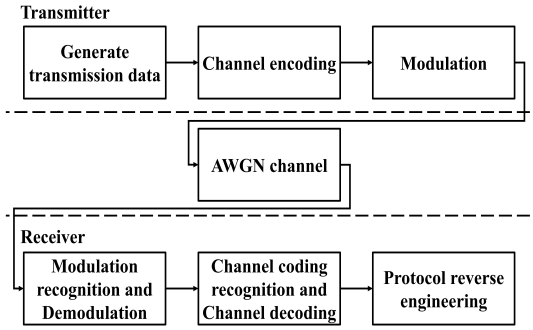


그림 9. 변조 및 채널 코딩 인식과 프로토콜 역공학 시뮬레이션 overview
Fig. 9. Overview of modulation and channel coding recognition and protocol reverse engineering simulation

등을 고려하지 않는 통신 채널인 AWGN 채널만을 고려하여 시뮬레이션을 진행한다. 시뮬레이션 구현을 위한 프로그래밍 언어로는 Python을 사용하였다.

3.1 송신부

송신부에서는 송신 데이터를 생성하고 생성된 데이터로 채널 인코딩을 수행하여 코드 워드를 생성한다. 코드 워드를 전송하기 위해 변조 과정을 수행한다.

3.1.1 송신 데이터 생성

본 논문에서 사용하는 송신 데이터는 Wireshark를 통해서 생성한 .pcap 파일이다. .pcap 파일을 생성하기 위해 TCP (Transmission Control Protocol)를 이용하여 파일을 전송하는 서버와 클라이언트를 실행 후 Wireshark를 사용하여 서버와 클라이언트간의 통신 패킷을 수집한다. 이러한 방법으로 생성한 .pcap 파일을 시뮬레이션에서 송신 데이터로 사용한다. 그림 10은 본 논문에서 사용하는 서버와 클라이언트간의 TCP 시퀀스 다이어그램을 보여준다. 그림 10의 SYN (synchronize), SYN-ACK (synchronize-acknowledgment) 패킷을 사용하여 서버와 클라이언트는 통신을 위해 서로 연결된다. 클라이언트는 PSH-ACK (push-acknowledgment) 패킷을 사용하여 데이터를 전송하고 서버는 ACK (acknowledgment) 패킷을 사용하여 데이터를 수신 여부를 알려준다. FIN-ACK (finish-acknowledgment) 패킷을 사용하여 서버와 클라이언트는 통신을 종료한다. Wireshark로 생성한 .pcap 파일은 그림 10과 같은 패킷들로 구성된다.

3.1.2 채널 인코딩

본 논문에서는 채널 코딩 방법으로 convolutional 코드만을 고려하며 서로 다른 파라미터를 갖는 (2, 1, 3),

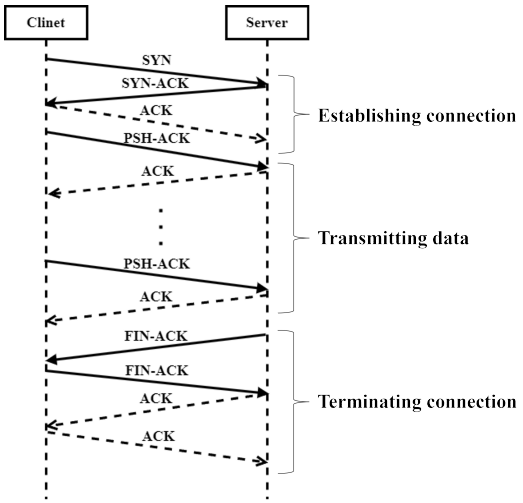


그림 10. TCP 시퀀스 다이어그램
Fig. 10. TCP sequence diagram

표 3. Convolutional 코드의 생성 시퀀스
Table 3. Generator sequences of convolutional code

Convolutional code	Generator sequences
(2, 1, 3)	$g^{(1)}=(1\,0\,1), g^{(2)}=(1\,1\,1)$
(2, 1, 4)	$g^{(1)}=(1\,0\,1\,1), g^{(2)}=(1\,1\,1\,1)$
(2, 1, 5)	$g^{(1)}=(1\,0\,1\,1\,1), g^{(2)}=(1\,1\,0\,0\,1)$

(2, 1, 4), (2, 1, 5) convolutional 코드를 사용하여 시뮬레이션을 수행한다. 그림 11은 (2, 1, 3) convolutional 코드 인코더를 보여주고 표 3은 3가지 convolutional 코드의 생성 시퀀스를 보여준다. b_l은 l번째 입력이며 b_{l-i}는 $l-i$번째 입력을 의미한다. $g_0^{(1)}=1$, $g_1^{(1)}=0$, $g_2^{(1)}=1$이고 $g_j^{(2)}$는 모두 1이다. $c_l^{(1)}$은 $g^{(1)}$으로 생성한 코드 워드이고 $c_l^{(2)}$은 $g^{(2)}$으로 생성한 코드 워드이다. 그림 11에서 볼 수 있듯이 convolutional 코드 인코더는 생성 시퀀스를 기반으로 코드 워드를 생성한다. (2, 1, 4)와 (2, 1, 5) convolutional 코드 역시 표 3의 생성 시퀀스를 활용하여 인코더를 구성하였다.

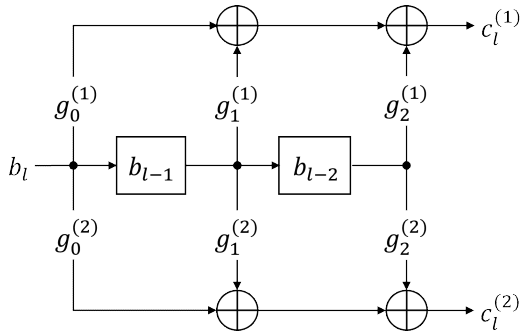


그림 11. (2, 1, 3) convolutional 코드 인코더
Fig. 11. (2, 1, 3) convolutional code encoder

3.1.3 변조

본 논문에서는 변조 방식으로 BPSK, QPSK, 8PSK 3가지를 사용한다. 변조 인식에서 수신 신호의 성상도를 이용하기 때문에 성상도로 표현할 수 있는 변조 방식인 M-PSK 방식을 사용하였다. 그림 12는 변조 방식에 따른 심볼 매핑 성상도를 보여준다. 그림 10의 (a), (b), (c)는 각각 BPSK, QPSK, 8PSK이며 비트에 따른 심볼 매핑 결과를 확인할 수 있다.

3.2 수신부

AWGN 채널 환경 구현을 위해 송신부에서 채널 인

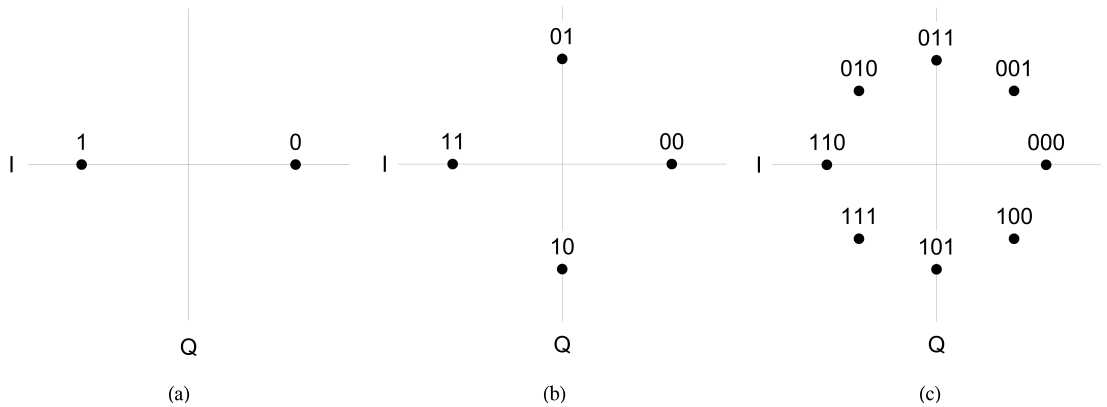


그림 12. 심볼 매핑 성상도: (a) BPSK, (b) QPSK, (c) 8PSK
Fig. 12. Symbol mapping constellation diagrams: (a) BPSK, (b) QPSK, (c) 8PSK

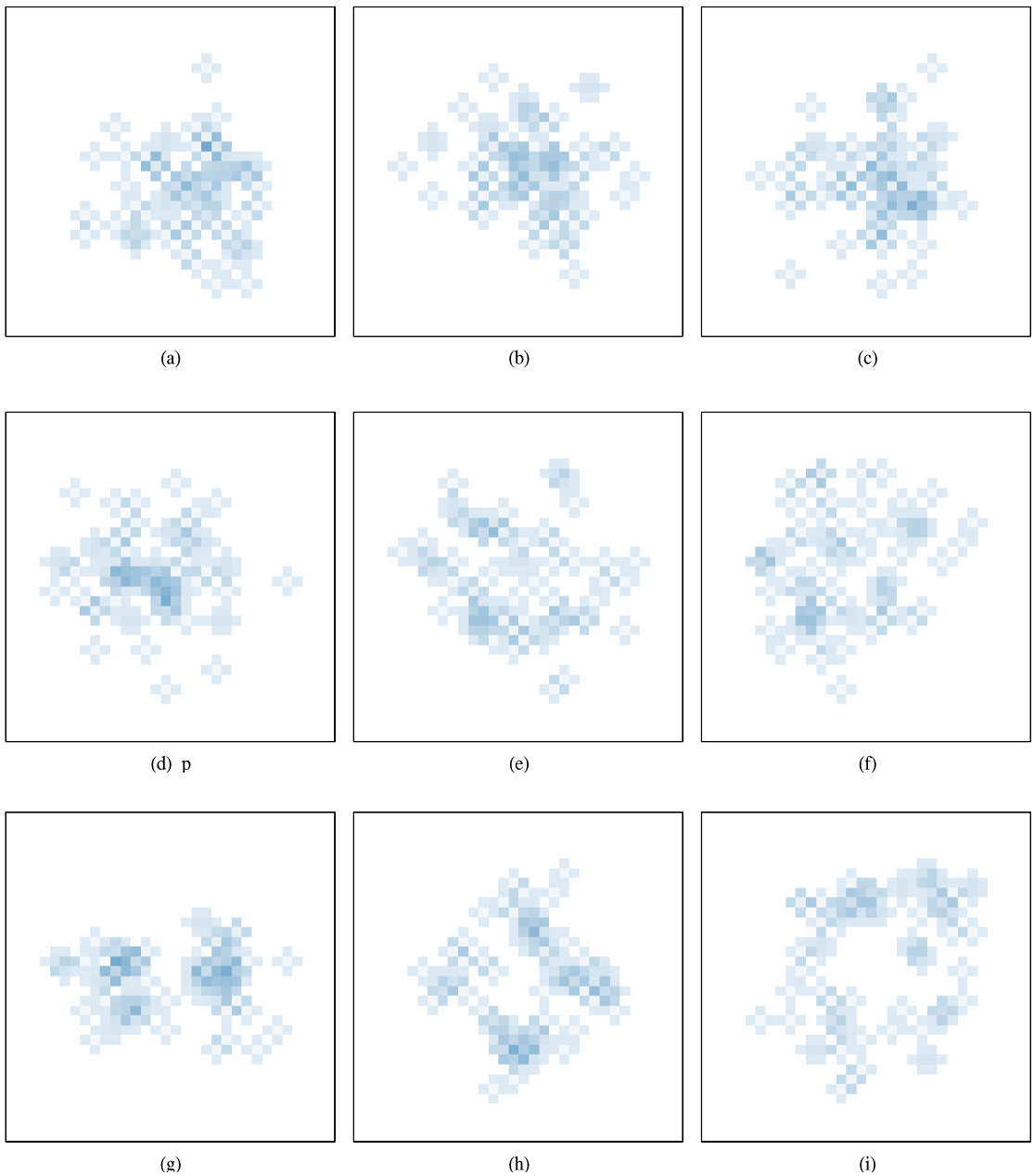


그림 13. SNR에 따른 성상도: (a) SNR = -5dB, BPSK, (b) SNR = -5dB, QPSK, (c) SNR = -5dB, 8PSK, (d) SNR = 0dB, BPSK, (e) SNR = 0dB, QPSK, (f) SNR = 0dB, 8PSK, (g) SNR = 5dB, BPSK, (h) SNR = 5dB, QPSK, (i) SNR = 5dB, 8PSK

Fig. 13. Constellation diagrams for different SNR values: (a) SNR = -5dB, BPSK, (b) SNR = -5dB, QPSK, (c) SNR = -5dB, 8PSK, (d) SNR = 0dB, BPSK, (e) SNR = 0dB, QPSK, (f) SNR = 0dB, 8PSK, (g) SNR = 5dB, BPSK, (h) SNR = 5dB, QPSK, (i) SNR = 5dB, 8PSK

코딩 및 변조를 수행한 송신 데이터에 SNR에 따른 Gaussian 노이즈를 $r = x + n$과 같이 추가하였다. r은 수신 벡터, x는 송신 벡터, n은 평균이 0이고 분산이 $N_o/2$인 Gaussian 노이즈이며 N_o는 노이즈 전력 스펙트럼 밀도이다. 수신부에서는 Gaussian 노이즈가 추가된 송신 데이터를 사용하여 변조 인식과 복조를 수행하고 채널 코딩 인식과 채널 디코딩을 수행한다. 또한 프로토콜 분석을 위해 CSP 알고리즘을 활용하여 프로토

콜의 message format을 추출하고 FSM을 생성한다.

3.2.1 변조 인식과 복조

본 논문에서는 변조 인식을 위해 수신 벡터를 사용하여 성상도를 생성하고 이를 CNN의 입력으로 사용한다. 그림 13은 시뮬레이션에서 SNR이 -5, 0, 5dB인 경우, BPSK, QPSK, 8PSK 성상도를 보여준다. 그림 13의 (a), (b), (c)는 SNR -5dB일 때 각각 BPSK, QPSK, 8PSK의 성상도이고, (d), (e), (f)는 SNR이 0dB, (g), (h), (i)는 SNR이 5dB인 경우이다. 2.1에서 설명한 CNN 모델은 64개의 IQ 데이터를 사용한 성상도를 사용하였기 때문에 시뮬레이션에서도 64개의 IQ 데이터를 사용하고 그림 13도 64개의 IQ 데이터를 사용한 결과이다. 그림 13에서 볼 수 있듯이 SNR이 -5, 0dB인 경우에는 변조 방식을 육안으로 쉽게 구별할 수 없고 5dB에서는 구별 가능하다. 그림 13과 같은 성상도는 2.1에서 설명한 변조 인식을 위한 CNN 모델의 입력으로 사용된다.

그림 14는 SNR이 0dB일 때 변조 인식 결과를 보여준다. 그림 14의 (a)는 BPSK를 사용한 신호에 대한 CNN 출력이고 그림 14의 (b)는 QPSK, 그림 14의 (c)는 8PSK인 경우이다. 또한 그림 14는 채널 코딩이 (2, 1, 3) convolutional 코드로 설정하였을 때의 결과이다. 그림 14의 real M은 실제 변조 방식을 나타내고 estimation M은 CNN 모델을 사용하여 추정한 변조 방식을 나타낸다. estimation M은 수신 데이터에 대한 CNN 모델 출력 중 가장 많은 수의 변조 방식을 선택한다. 그림 14에서 변조 방식 뒤의 숫자가 CNN 모델 출력 수이다. 시뮬레이션 수행에 있어 수신 데이터가 커질수록 수행 시간이 증가하는 문제를 예방하기 위해 최대 100개의 성상도만을 사용하도록 하였다. 그림 14의 (a)와 (b)는 변조 방식을 정확히 추정하였지만, 8PSK의 경우에는 정확한 변조 방식을 추정하지 못하였다. 시뮬레이션에서는 그림 14와 같이 변조 인식을 위한 CNN으로 변조 방식을 추정하고 복조를 수행한다. 복조는 hard decision 방법을 사용하여 수신된 IQ 데이터를 0과 1인 디지털 신호로 복조한다.

```
Modulation recognition...
real M: 1 estimation M: 1 [BPSK, QPSK, 8PSK]: [69. 24.  7.]
```
(a)

```
Modulation recognition...
real M: 2 estimation M: 2 [BPSK, QPSK, 8PSK]: [15. 73. 12.]
```
(b)

```
Modulation recognition...
real M: 3 estimation M: 2 [BPSK, QPSK, 8PSK]: [12. 45. 43.]
```
(c)

그림 14. SNR = 0dB일 때 변조 인식 결과: (a) BPSK, (b) QPSK, (c) 8PSK
Fig. 14. Modulation recognition results at SNR = 0dB: (a) BPSK, (b) QPSK, (c) 8PSK

3.2.2 채널 코딩 인식과 채널 디코딩

채널 코딩 인식에서는 3.2.1의 복조 결과를 채널 코딩 인식을 위한 CNN의 입력으로 사용하여 채널 코딩 방법을 추정한다. 복조 결과를 사용하기 위해서는 2.2에서 설명한 채널 코딩 인식을 위한 CNN은 64비트의 데이터를 입력으로 사용하기 때문에 복조된 신호를 64비트의 길이로 자르는 과정이 필요하다.

그림 15는 SNR이 0dB일 때 채널 코딩 인식 결과를 보여준다. 그림 15의 (a), (b), (c)는 각각 (2, 1, 3), (2, 1, 4), (2, 1, 5) convolutional 코드인 경우이다. 또한 그림 15는 BPSK 변조를 사용한 결과이다. 그림 15의 real C는 실제 채널 코딩 방식을 나타내고 estimation C는 CNN 모델을 사용하여 추정한 채널 코딩 방식을 나타낸다. estimation C도 변조 인식에서와 동일하게 수신 데이터에 대한 CNN 모델 출력 중 가장 많은 수의 채널 코딩 방식을 선택한다. 그림 15에서 채널 코딩 방식 뒤의 숫자가 CNN 모델 출력 수이다. 시뮬레이션 수행에 있어 수신 데이터가 커질수록 수행 시간이 증가하는 문제를 예방하기 위해 최대 100개의 복조 신호만을 사용하였다. 채널 코딩 인식은 0dB에서 3가지 방식 모두 정확히 추정할 수 있었다. Convolutional 코드의 디코딩 방법으로는 Viterbi 디코딩 사용하였다[19]. 채널 코딩 인식을 위한 CNN으로 채널 코딩 방법을 예측하고 이를 활용하여 Viterbi 디코딩을 통해 전송 메시지를 추정한다.

```
Channel coding recognition...
real C: 1 estimation C: 1 [C(2,1,3), C(2,1,4), C(2,1,5)]: [55. 14. 31.]
```
(a)

```
Channel coding recognition...
real C: 2 estimation C: 2 [C(2,1,3), C(2,1,4), C(2,1,5)]: [22. 48. 30.]
```
(b)

```
Channel coding recognition...
real C: 3 estimation C: 3 [C(2,1,3), C(2,1,4), C(2,1,5)]: [24. 22. 54.]
```
(c)

그림 15. SNR = 0dB일 때 채널 코딩 인식 결과: (a) (2, 1, 3), (b) (2, 1, 4), (c) (2, 1, 5) convolutional 코드
Fig. 15. Channel coding recognition results at SNR = 0dB: (a) (2, 1, 3), (b) (2, 1, 4), (c) (2, 1, 5) convolutional code

3.2.3 프로토콜 역공학

본 논문에서는 프로토콜 분석을 위해 빈번한 시퀀스 탐색 및 프로토콜 구조를 파악하는 알고리즘인 CSP 알고리즘을 사용한다. CSP 알고리즘은 일반적인 시퀀스 탐색 알고리즘과 달리 연속적인 시퀀스를 추출하는 알고리즘으로써 추출한 시퀀스의 field type을 결정하고 field의 정보를 이용하여 프로토콜의 구조를 추정한다. CSP 알고리즘은 그림 16과 같은 방법으로 빈번한 시퀀스를 추출하고 프로토콜 구조를 추정한다. CSP 알고리즘은 첫 번째 CSP를 통해 field format을 추출한다. field format은 하나의 빈번한 시퀀스로 SF(v) (Static Field(v)), DF(v) (Dynamic Field(v)), DF (Dynamic Field), GAP와 같은 type을 갖는다. SF(v)는 시퀀스의 값과 길이가 변하지 않는 경우이고 DF(v)는 시퀀스의 값, 길이가 변하지만 예측 가능한 경우, DF는 시퀀스의 값을 예측할 수 없지만 길이는 예측 가능한 경우, GAP은 시퀀스의 값, 길이 모두 예측 불가능한 field를 나타낸다. 첫 번째 CSP로는 SF(v) type의 field만 추출한다.

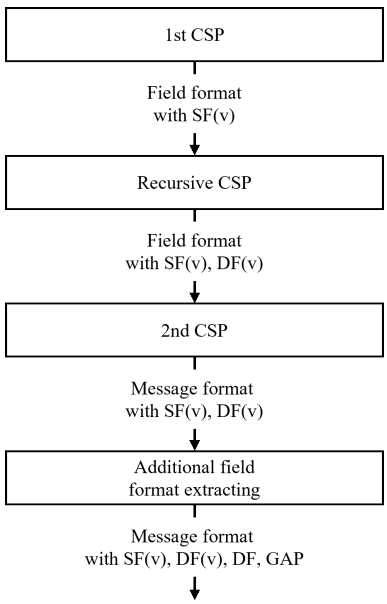


그림 16. CSP 알고리즘 동작 방식
Fig. 16. Process of CSP algorithm

표 4. 데이터 타입에 따른 프로토콜 패킷 수와 시퀀스 추출률
Table 4. Number of protocol packet and sequence extraction ratio by data type

Data type	Number of protocol packet	sequence extraction ratio
Text	12	0.553
Image	249	0.449

그 후 Recursive CSP로 DF(v) type의 field를 추출한다. 두 번째 CSP는 SF(v), DF(v) field format을 사용하여 field format의 집합인 message format을 추출한다. 마지막으로는 CSP로 추출하지 못한 field format의 type을 DF나 GAP으로 결정한다[20].

표 4는 2가지의 데이터를 전송하였을 때의 프로토콜 패킷 수와 시퀀스 추출률을 보여준다. 시퀀스 추출률은 추출된 시퀀스 길이와 전체 프로토콜 패킷의 메시지 길이의 비로 1에 가까울수록 많은 시퀀스를 추출했음을 보여준다. 본 논문에서는 SF, SF(v) type을 갖는 field의 시퀀스를 추출한 시퀀스로 결정하였다. 표 4에서는 0.5 KB 크기의 텍스트 데이터와 120KB 크기의 이미지 데이터를 사용하였고 각각 12개와 249개의 프로토콜 패킷이 생성되었다는 것을 확인할 수 있다. 또한 시퀀스 추출률은 텍스트 데이터를 사용한 경우 0.553으로 이미지 데이터를 사용한 0.449보다 높은 추출률을 보여주었다. 하지만 텍스트 데이터의 경우, 프로토콜 패킷이 매우 적어 비교적 많은 시퀀스를 추출할 수 있어 높은 추출률을 보여주었다.

그림 17과 그림 18은 표 4의 텍스트 데이터를 사용하여 CSP 알고리즘을 수행한 결과로 그림 17은 CSP 알고리즘으로 추출한 message format을 시각화한 것이며 그림 18은 message format을 활용하여 생성한 FSM을 보여준다. 그림 17에서 빨간색은 SF(v), 노란색은 DF(v), 초록색은 DF, 파란색은 GAP을 나타낸다. 그림 8의 TCP 시퀀스 다이어그램에서 볼 수 있듯이 SYN, SYN-ACK, ACK, PSH-ACK, FIN-ACK로 5가지 패킷이 있다. 그림 17에서는 5가지 패킷 중 2가지를 추출한 것으로 보인다. 그림 18은 그림 17의 message format을 이용하여 생성한 FSM으로 message format의 순서 및 흐름을 파악할 수 있다. 그림 18의 Initial은

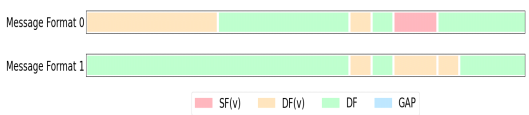


그림 17. 텍스트 데이터를 사용한 Message format
Fig. 17. Message format using text data

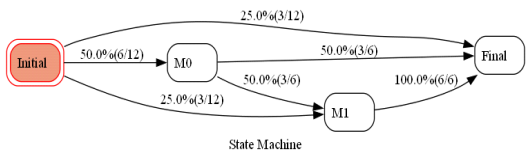


그림 18. 텍스트 데이터를 사용한 Finite state diagram
Fig. 18. Finite state diagram using text data

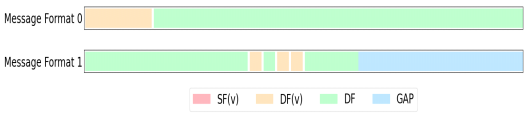


그림 19. 이미지 데이터를 사용한 Message format
Fig. 19. Message format using image data

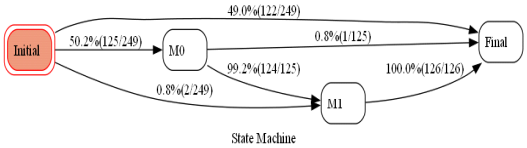


그림 20. 이미지 데이터를 사용한 Finite state diagram
Fig. 20. Finite state diagram using image data

시작, Final은 끝을 나타내고 M0는 그림 17의 Message Format 0, M1은 그림 17의 Message Format 1을 나타낸다. 그림 16을 보면 Initial에서 Final로 연결된 패킷이 전체 패킷 중 25%(3/12)로 3개 패킷에서 빈번한 시퀀스를 추출하지 못했음을 의미한다. 또한 6개 패킷은 M0를 포함하고 있고 그중 3개 패킷은 M1도 포함하고 있고 M1만을 포함하는 패킷은 3개가 있다. Message format을 시각화한 그림 19와 FSM을 나타낸 그림 20은 표 4의 이미지 데이터를 사용하여 CSP 알고리즘을 수행한 결과이다. 이미지 데이터를 사용한 그림 19의 경우, 텍스트 데이터를 사용한 그림 17과 달리 SF type의 field는 추출하지 못하였고 GAP type의 field가 있는 것을 확인할 수 있다. 그림 20은 그림 18과 유사한 형태를 보여주지만 Initial에서 Final로 연결된 패킷이 49%로 증가한 것을 확인할 수 있다.

3.3 시뮬레이션 실행 결과

시뮬레이션은 Python을 사용하여 구현하였으며 CLI (Command Line Interface)를 제공한다. 그림 21은 시뮬레이션의 CLI를 보여준다. 그림 21에서 볼 수 있듯이 사용자에게 전송할 파일 이름을 입력받아 .pcap 파일을 생성한다. 그 후 사용자는 채널 코딩 방법과 변조 방법을 선택하고 AWGN 채널의 SNR을 선택한다. 시뮬레이션은 사용자가 모든 선택이 끝나면 앞서 설명한 채널 인코딩 및 변조를 수행하고 AWGN 채널을 통과한 데이터에 대한 변조 인식 및 채널 코딩 인식 결과를 출력하고 변조 및 채널 디코딩을 수행한다. 마지막으로 프로토콜 분석 후 그 결과를 저장한다.

그림 22는 다양한 환경의 시뮬레이션 결과로 (a)는 채널 코딩 방법으로 (2, 1, 5) convoutional 코드, 변조 방식으로 BPSK, SNR이 -10dB인 경우, (b)는 (2, 1, 3) convoutional 코드, QPSK, 0dB인 경우, (c)는 (2, 1, 4) convoutional 코드, 8PSK, 10dB인 경우이다. 그림 18은 시뮬레이션 CLI 출력을 최소화하여 변조 및 채널 코딩 인식 결과만 출력하도록 하였다. 그림 22에서 볼 수 있듯이 변조 및 채널 코딩 방식 변경과 AWGN 채널 환경을 변경함으로써 다양한 환경에서 변조 및 채널 코딩 인식 결과를 확인할 수 있다. 추후 변조 및 채널 코딩 유형을 추가하는 등에 방식을 통해 다양한 실험과 성능 비교를 할 수 있는 시뮬레이션으로 발전할 수 있다. 또한 통신 채널도 AWGN 채널이 아닌 페이딩 채널 등으로 변경할 수 있으며 프로토콜 역공학 알고리즘도 CSP 알고리즘이 아닌 다른 알고리즘을 변경하는 등의

```
Please enter data path: data/test.txt
Creating .pcap...

Channel code method(1. C(2,1,3), 2. C(2,1,4), 3.C(2,1,5)): 1
Modulation method(1. BPSK, 2. QPSK, 3. 8PSK): 1
Select SNR(-20 ~ 20dB): 10

Channel encoding...

Modulation...

AWGN Channel...

Modulation recognition...
real M: 1 estimation M: 1 [BPSK, QPSK, 8PSK]: [96.  3.  1.]

Demodulation...

Channel coding recognition...
real C: 1 estimation C: 1 [C(2,1,3), C(2,1,4), C(2,1,5)]: [100.   0.   0.]

Channel decoding...

Protocol reverse engineering...
save received data
save messageF.png and state_diagram.png
```

그림 21. 시뮬레이션 CLI
Fig. 21. CLI of simulation

```
Channel code method(1. C(2,1,3), 2. C(2,1,4), 3.C(2,1,5)): 3
Modulation method(1. BPSK, 2. QPSK, 3. 8PSK): 1
Select SNR(-20 ~ 20dB): -10
real M: 1 estimation M: 2 [BPSK, QPSK, 8PSK]: [ 9. 81. 10.]
real C: 3 estimation C: 3 [C(2,1,3), C(2,1,4), C(2,1,5)]: [25. 35. 40.]
```

(a)

```
Channel code method(1. C(2,1,3), 2. C(2,1,4), 3.C(2,1,5)): 1
Modulation method(1. BPSK, 2. QPSK, 3. 8PSK): 2
Select SNR(-20 ~ 20dB): 0
real M: 2 estimation M: 2 [BPSK, QPSK, 8PSK]: [23. 63. 14.]
real C: 1 estimation C: 1 [C(2,1,3), C(2,1,4), C(2,1,5)]: [56. 18. 26.]
```

(b)

```
Channel code method(1. C(2,1,3), 2. C(2,1,4), 3.C(2,1,5)): 2
Modulation method(1. BPSK, 2. QPSK, 3. 8PSK): 3
Select SNR(-20 ~ 20dB): 10
real M: 3 estimation M: 3 [BPSK, QPSK, 8PSK]: [ 8. 14. 78.]
real C: 2 estimation C: 2 [C(2,1,3), C(2,1,4), C(2,1,5)]: [ 8. 92.  0.]
```

(c)

그림 22. 다양한 환경의 시뮬레이션 결과: (a) (2, 1, 5), BPSK, -10dB, (b) (2, 1, 3), QPSK, 0dB, (c) (2, 1, 4), 8PSK, 10dB
Fig. 22. Simulation results in various conditions: (a) (2, 1, 5), BPSK, -10dB, (b) (2, 1, 3), QPSK, 0dB, (c) (2, 1, 4), 8PSK, 10dB

활용이 가능하다.

Ⅳ. 결 론

본 논문에서는 송수신기가 통신 제원을 공유하지 않는 블라인드 통신 환경에서 통신 신호를 복원하기 위해 CNN을 사용하여 변조 및 채널 코딩을 인식하고 CSP 알고리즘을 활용하여 프로토콜 분석을 수행하는 시뮬레이션을 수행하였다. 시뮬레이션은 크게 송신부, 수신부로 나누어진다. 송신부에서는 채널 인코딩과 변조를 수행하고 수신부에서는 CNN을 사용하여 변조 방식을 추정하여 복조를 수행하며 채널 코딩 방법도 CNN을 사용하여 추정하여 채널 디코딩을 수행한다. 또한 수신부에서는 프로토콜 분석을 위해 CSP 알고리즘을 활용하여 프로토콜의 message format을 추출하고 FSM을 생성한다.

본 논문에서 제안한 변조 및 채널 코딩 인식을 위한 CNN 모델은 학습에 사용한 3가지 방식만을 분류할 수 있으며 학습에 사용한 방식이 아닌 다른 입력으로 들어오면 분류하지 못하는 단점이 있다. 이를 해결하기 위해서는 다양한 변조 및 채널 코딩 방법에 대한 추가적인 학습이 필요하다. 또한 본 논문에서 사용한 CNN 모델은 변조 및 채널 코딩 인식의 딥러닝 모델 활용 가능성을 확인하기 위해 간단한 구조를 갖도록 구성하였기에 변조 및 채널 코딩 인식에 대한 최적의 성능을 보여주는 모델이 아니다. 하지만 본 논문에서는 필요에 따라 최적의 성능을 갖는 딥러닝 모델을 CNN 모델 대신 사용할 수 있는 블라인드 통신 환경 시뮬레이션을 구현하였다.

이러한 블라인드 통신 환경에 대한 시뮬레이션을 통해 변조 및 채널 코딩 인식 기술과 프로토콜 역공학 기술의 성능 평가 및 활용 가능성을 확인할 수 있다. 또한 CNN과 같은 딥러닝의 학습을 위해서는 많은 수의 데이터가 필요하지만, 블라인드 통신 환경은 일반적인 통신 환경이 아니기에 데이터를 수집하는 것에 어려움이 있다. 본 논문의 시뮬레이션이 이러한 데이터 부족을 해결할 수 있다고 생각된다. 본 논문의 시뮬레이션에 사용된 변조 방법이나 채널 코딩 방법이 각각 3가지뿐이지만 필요에 따라 증가시킬 수 있으며 프로토콜 종류도 변경할 수 있으며 프로토콜 역공학 알고리즘도 변경할 수 있다. 이처럼 유동적으로 변형 가능한 시뮬레이션은 변조 및 채널 코딩 인식, 프로토콜 역공학에 대한 연구에 있어 데이터 생성 및 성능 평가에 도움을 줄 수 있다.

References

[1] S. Ye, S. H. Wong, and C. Worrall, "Enhanced physical downlink control channel in LTE advanced release 11," *IEEE Commun. Mag.*, vol. 51, no. 2, pp. 82-89, 2013.

[2] A. Jalali and Z. Ding, "Joint detection and decoding of polar coded 5G control channels," *IEEE Trans. Wireless Commun.*, vol. 19, no. 3, pp. 2066-2078, 2020.

[3] S. Ahmadi, *5G NR: Architecture, Technology, Implementation, and Operation of 3GPP New Radio Standards*, Cambridge, MA, USA: Academic Press, 2019.

[4] S. Peng, S. Sun, and Y. D. Yao, "A survey of modulation classification using deep learning: Signal representation and data preprocessing," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7020-7038, 2021.

[5] A. Bonvard, S. Houcke, R. Gautier, and M. Marazin, "Classification based on euclidean distance distribution for blind identification of error correcting codes in noncooperative contexts," *IEEE Trans. Signal Process.*, vol. 66, no. 10, pp. 2572-2583, 2018.

[6] G. Sumen, B. A. Celebi, G. K. Kurt, A. Gorcin, and S. T. Başaran, "Multi-channel learning with preprocessing for automatic modulation order separation," *IEEE Symp. Comput. and Communi.* (ISCC), pp. 1-5, 2022.

[7] T. T. Dao, D. I. Noh, Q. V. Pham, M. Hasegawa, H. Sekiya, and W. J. Hwang, "VT-MCNet: High-accuracy automatic modulation classification model based on vision transformer," *IEEE Commun. Lett.*, vol. 28, no. 1, pp. 98-102, 2024.

[8] X. Huang, S. Sun, X. Yang, and S. Peng, "Recognition of channel codes based on BiLSTM-CNN," *WOCC*, pp. 151-154, 2022.

[9] X. Qin, S. Peng, X. Yang, and Y. D. Yao, "Deep learning based channel code recognition using TextCNN," *IEEE Int. Symp. DySPAN*, pp. 1-5, 2019.

[10] K. Anastasis and M. Michail, "ICSREF: A framework for automated reverse engineering

of industrial control systems binaries," 2018. (http://dx.doi.org/10.14722/ndss.2019.23271)

[11] Z. Xu, C. Wen, and S. Qin, "Learning types of binaries," *Int. Conf. Formal Eng. Methods*, pp. 430-446, 2017.

[12] Y. Ye, Z. Zhang, F. Wang, X. Zhang, and D. Xu, "NETPLIER: Probabilistic network protocol reverse engineering from message traces," *Netw. and Distrib. Syst. Secur. Symp.*, pp. 1-18, 2021.

[13] Y.-H. Goo, K.-S. Shim, M.-S. Lee, and M.-S. Kim, "Protocol specification extraction based on contiguous sequential pattern algorithm," *IEEE Access*, vol. 7, pp. 36057-36074, 2019. (http://dx.doi.org/10.1109/ACCESS.2019.2905353)

[14] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Advances in NIPS*, pp. 1097-1105, 2012.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," *ICML*, pp. 807-814, 2010.

[16] J. Han and C. Moraga, "The influence of the sigmoid function parameters on the speed of backpropagation learning," *Int. Wkshp. Artificial Neural Netw.*, pp. 195-201, 1995.

[17] A. Mao, M. Mohri, and Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," *ICML*, pp. 23803-23828, 2023.

[18] B. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014. (https://doi.org/10.48550/arXiv.1412.6980)

[19] F. Chan and D. Haccoun, "Adaptive viterbi decoding of convolutional codes over memoryless channels," *IEEE Trans. Commun.*, vol. 45, no. 11, pp. 1389-1400, 1997.

[20] H. Cho, J. Park, M. Chae, H. Lee, and W. Lim, "Protocol structure and sequence detection method for multi-protocol analysis," *J. KICS*, vol. 49, no. 4, pp. 556-566, 2024.

[21] T. S. Rappaport, *Wireless communications: Principles and Practice*, 2nd Ed., Prentice Hall, pp. 267-272, 2002.

조 현 우 (Hyunwoo Cho)
2021년 3월~현재 : 금오공과대학교 전자공학과 박사과정
<관심분야> 머신 러닝, 채널 코딩, 자동변조인식
[ORCID:0000-0003-4504-0965]

채 명 호 (Myoungho Chae)
2014년 2월~현재 : 국방과학연구소 선임연구원
<관심분야> 통신 프로토콜, 기계학습, 자동변조인식
[ORCID:0000-0001-7741-1818]

임 완 수 (Wansu Lim)
2024년 3월~현재 : 성균관대학교 전자전기공학부 교수
<관심분야> 통신 프로토콜, 기계학습, 자동변조인식
[ORCID:0000-0003-2533-3496]