End-to-End Learning From Spectrum Data: A Deep Learning Approach for Wireless Signal Identification in Spectrum Monitoring Applications, IEEE Access 2018.

유비쿼터스네트워크 프로젝트 구현

한승우

Index

Introduction & Data acquisition

▶ 개발환경

- Anconda3, Python 3.7
- IDE = Spyder4

▶ 본 구현에 사용된 라이브러리

- Tensorflow-gpu 1.15.0, Keras 2.2.5
- Numpy 1.17.4
- Pandas 0.25.3
- Pickle, math, cmath (기본 모듈)
- Matplotlib 3.1.2 Seaborn 0.9.0

Introduction & Data acquisition

- 본 논문은 RadioML 데이터 (IQ vector)를 phase/amplitude vector, FFT vector로 만든 뒤
- IQ vector / phase/amplitude vector / FFT vector와 주파수*를 이용하여 CNN으로 분류
- 본 논문은 2D CNN이나, Input size 오류 발생으로 부득이하게 1D CNN으로 진행
- Dataset : RadioML 2016.10a Modulation dataset (https://www.deepsig.io/datasets)
- 데이터는 128개의 sample vector로 이루어져 있음. vector 총 개수는 총 220,000 data vector
- 11가지의 주파수를 원-핫 인코딩으로 변환
- 본 구현은 Radio signal modulation recognition만 진행, (Wireless interference identification은 구현하지 않음)
- 구현에 참고한 Python 2 기반으로 작성된 코드
- https://github.com/radioML/examples/blob/master/modulation_recognition/RML2016.10a_VTCNN2_example.ipynb

Experimental setting

- Keras의 to_categorical 모듈을 이용하여 주파수를 원-핫 인코딩으로 변환 (아래 자료 참고하여 활용)
- https://github.com/radioML/examples/blob/master/modulation_recognition/RML2016.10a_VTCNN2_example.ipynb

```
# 원핫 인코딩으로 11가지 주파수를 변환

y_train = to_categorical(list(map(lambda x: mods.index(lbl[x][0]), train_idx)))
y_test = to_categorical(list(map(lambda x: mods.index(lbl[x][0]), test_idx)))
```

	0	1	2	3	4
0	0	0	0	0	1
1	0	0	1	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	0
5	0	0	0	0	0
6	0	0	0	0	0
7	0	0	0	0	0
8	0	0	0	0	0
9	0	0	0	1	0
10	0	0	0	0	0
11	0	0	0	0	0

Experimental setting

- Cmath와 Numpy 라이브러리를 이용하여 phase vector와 amplitude vector, FFT vector를 계산

```python
for i in range(len(x_i_flat)):
    x_phase.append(cmath.atan(x_r_flat[i]/x_i_flat[i]))
```

```python
x_amp = (x_r**2+x_i**2)**1/2
```

```python
x_fft_r = np.fft.fft(x_r)
x_fft_i = np.fft.fft(x_i)
```

*Transformation 2 ($\mathbf{A}/\boldsymbol{\phi}$ vector): The **$\mathbf{A}/\boldsymbol{\phi}$ vector** is a mapping from the raw complex data vector $\boldsymbol{r}_k \in \mathbb{C}^N$ into two real-valued vectors, one that represents its phase, $\boldsymbol{\phi}$, and one that represents its magnitude $\mathbf{A}$,* i.e.

$$\mathbf{x}_k^{\mathbf{A}/\boldsymbol{\phi}} = \begin{bmatrix} \mathbf{x}_A{}^T \\ \mathbf{x}_\phi{}^T \end{bmatrix} \tag{26}$$

$$\begin{aligned} x_{\phi_n} &= \arctan\left(\frac{r_{q_n}}{r_{i_n}}\right), \\ x_{A_n} &= (r_{q_n}^2 + r_{i_n}^2)^{1/2}, \quad n = 0, \ldots, N-1 \end{aligned} \tag{27}$$

*Transformation 3 (FFT vector): The **FFT vector** is a mapping from the raw time-domain complex data vector $\boldsymbol{r}_k \in \mathbb{C}^N$ into its frequency-domain representation vector consisting of two sets of real-valued data vectors, one that carries the real component of its complex FFT $\boldsymbol{x}_{F_{re}}$ and one that holds the imaginary component of its FFT $\boldsymbol{x}_{F_{im}}$.* That is

$$\mathbf{x}_k^{\mathcal{F}} = \begin{bmatrix} \mathbf{x}_{F_{re}}{}^T \\ \mathbf{x}_{F_{im}}{}^T \end{bmatrix} \tag{30}$$

Experimental setting

- valiation set 설정

```
n_train = n_examples * 0.67
train_idx = np.random.choice(220000, size=int(n_train), replace=False)
test_idx = list(set(range(0,n_examples))-set(train_idx))
```

- Train (67%)를 제외한 나머지 33%을 validation set으로 설정함

and 33% for testing and validation. Hence, for modulation recognition 147,400 examples are used for training, while 72,600 examples for testing and validation. For the task

Experimental setting

- Keras 라이브러리를 이용하여 논문과 같이 레이어 구현 (단, 1D CNN으로 구현함)

```
model.add(Conv1D(filters=256, kernel_size=3, activation='relu', input_shape=(128,2)))
model.add(Dropout(0.6))
model.add(Conv1D(filters=80, kernel_size=3, activation='relu'))
model.add(Dropout(0.6))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.6))
model.add(Dense(11, activation='softmax'))
model.summary()
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy',  metrics=["accuracy", precision_metric, recall_metric, f1_metric])

history = model.fit(x_train, y_train, validation_data=(x_test, y_test), batch_size=1024, epochs=70, verbose=1)

loss, accuracy, precision, recall, f1 = model.evaluate(test_X_i, test_Y_i, verbose=1, batch_size=1024)
```

The CNNs were trained on 70 epochs and the model with the lowest validation loss is selected for evaluation. In total,

(GPU) Nvidia Tesla K80. For both use cases, 67% randomly selected examples are used for training in batch sizes of 1024,

We selected the Adaptive moment estimation (Adam) optimizer [30] to estimate the model parameters with a learning rate $\alpha = 0.001$ to ensure convergence. To speed up the model learning and convergence procedure, the input data was normalized and the ReLU activation units are selected.

TABLE 1. CNN structure.

Layer type	Input size	Parameters	Activation function
Convolutional layer	2×128	256 filters, filter size 1×3, dropout=0.6	ReLU
Convolutional layer	$256\times2\times128$	80 filters, filter size 2×3, dropout=0.6	ReLU
Fully connected layer	10240×1	256 neurons, dropout=0.6	ReLU
Fully connected layer	256×1	11 neurons or 15 neurons	Softmax

Experimental setting

- Keras 라이브러리를 이용하여 논문과 같이 레이어 구현
- Model.summary를 통한 레이어 요약

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d_1 (Conv1D)            (None, 126, 256)          1792
_________________________________________________________________
dropout_1 (Dropout)          (None, 126, 256)          0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 124, 80)           61520
_________________________________________________________________
dropout_2 (Dropout)          (None, 124, 80)           0
_________________________________________________________________
flatten_1 (Flatten)          (None, 9920)              0
_________________________________________________________________
dense_1 (Dense)              (None, 256)               2539776
_________________________________________________________________
dropout_3 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_2 (Dense)              (None, 11)                2827
=================================================================
Total params: 2,605,915
Trainable params: 2,605,915
Non-trainable params: 0
_________________________________________________________________
Train on 147400 samples, validate on 72600 samples
```

Experimental setting

- Performance results for modulation recognition (high = 18, medium=0, low=-8db) 1 running

Model	SNR	Precision	Recall	F1 score
CNN IQ	High	0.81798	0.66675	0.72566
	Medium	0.76640	0.58014	0.64758
	Low	0.61448	0.07282	0.12853
CNN A/P	High	0.85018	0.68897	0.76113
	Medium	0.74292	0.58999	0.65768
	Low*	-	-	-
CNN FFT	High	0.84119	0.41355	0.51482
	Medium	0.62613	0.30409	0.37437
	Low	0.39932	0.04052	0.0731

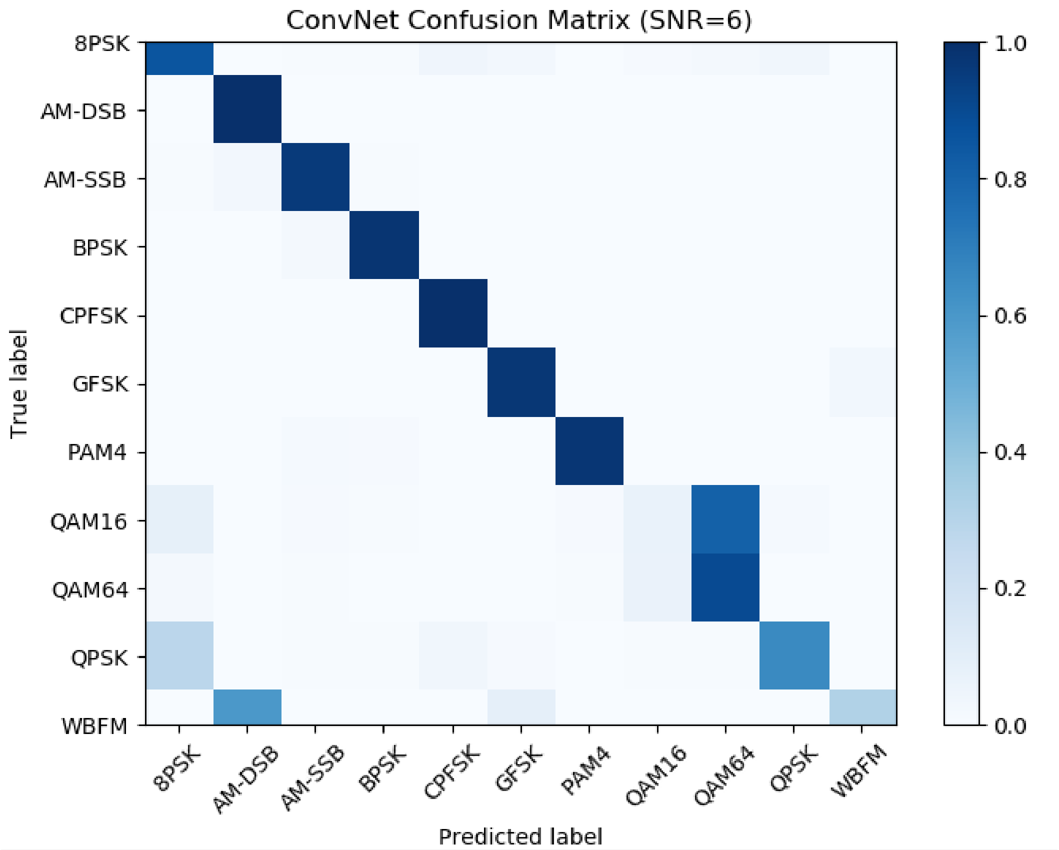
Case	Model	SNR	$\mathbf{P}_{avg}$	$\mathbf{R}_{avg}$	F_1 $\mathbf{score}_{avg}$
Mod. recognition	CNN_{IQ}^{M}	High	0.83	0.82	0.79
		Medium	0.75	0.75	0.72
		Low	0.36	0.32	0.30
	$\mathrm{CNN}_{A/\phi}^{M}$	High	0.86	0.84	0.82
		Medium	0.70	0.70	0.69
		Low	0.33	0.29	0.26
	$\mathrm{CNN}_{\mathcal{F}}^{M}$	High	0.71	0.68	0.67
		Medium	0.63	0.6	0.59
		Low	0.28	0.25	0.22

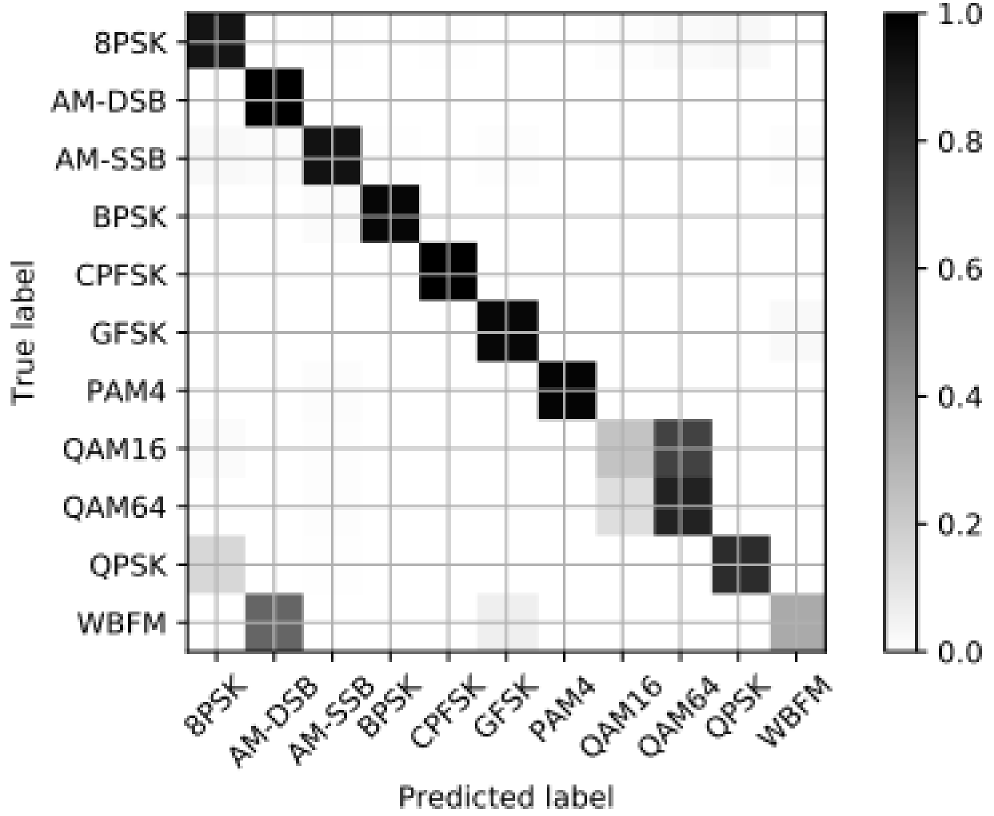
* 세 값 모두 0으로 나오는 결과가 있음

Analysis and results

- Performance results for modulation recognition classifiers (SNR=6db, IQ vector)

FIGURE 7. **Confusion matrices for the modulation recognition data for SNR 6dB. (a) $CNN^{M}_{\mathcal{IQ}}$. (b) $CNN^{M}_{\mathbf{A}/\phi}$. (c) $CNN^{M}_{\mathcal{F}}$.**



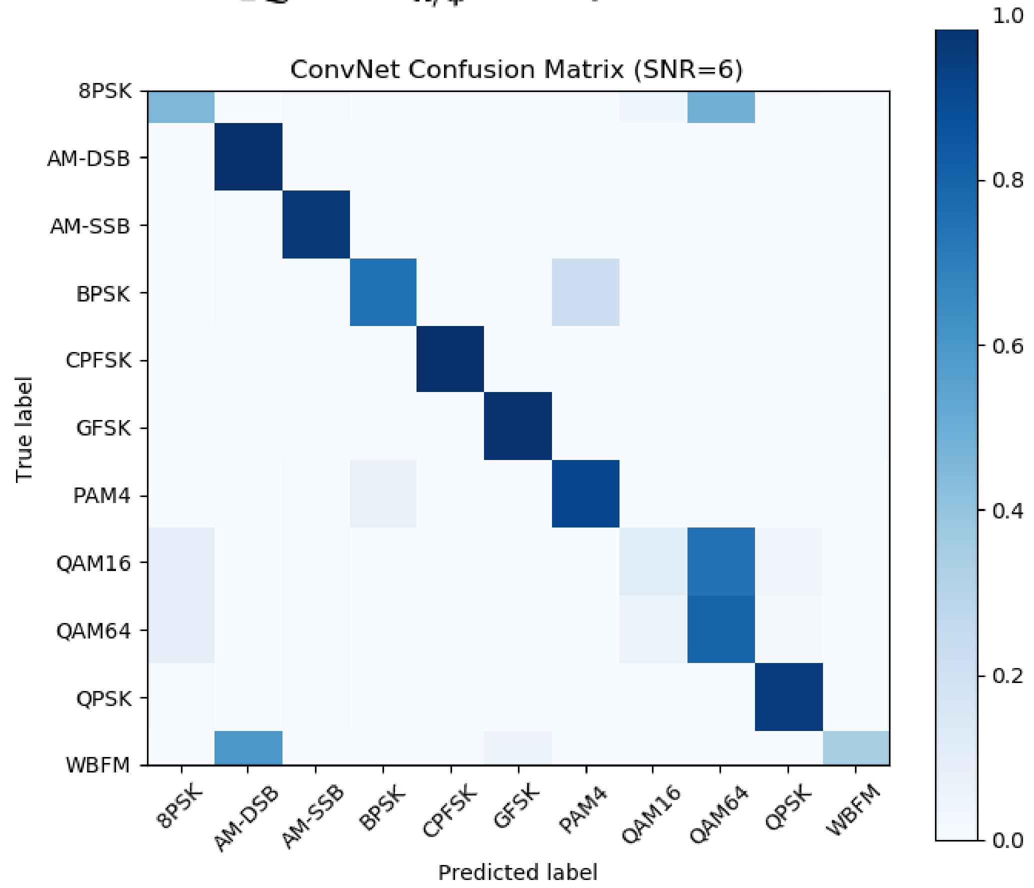


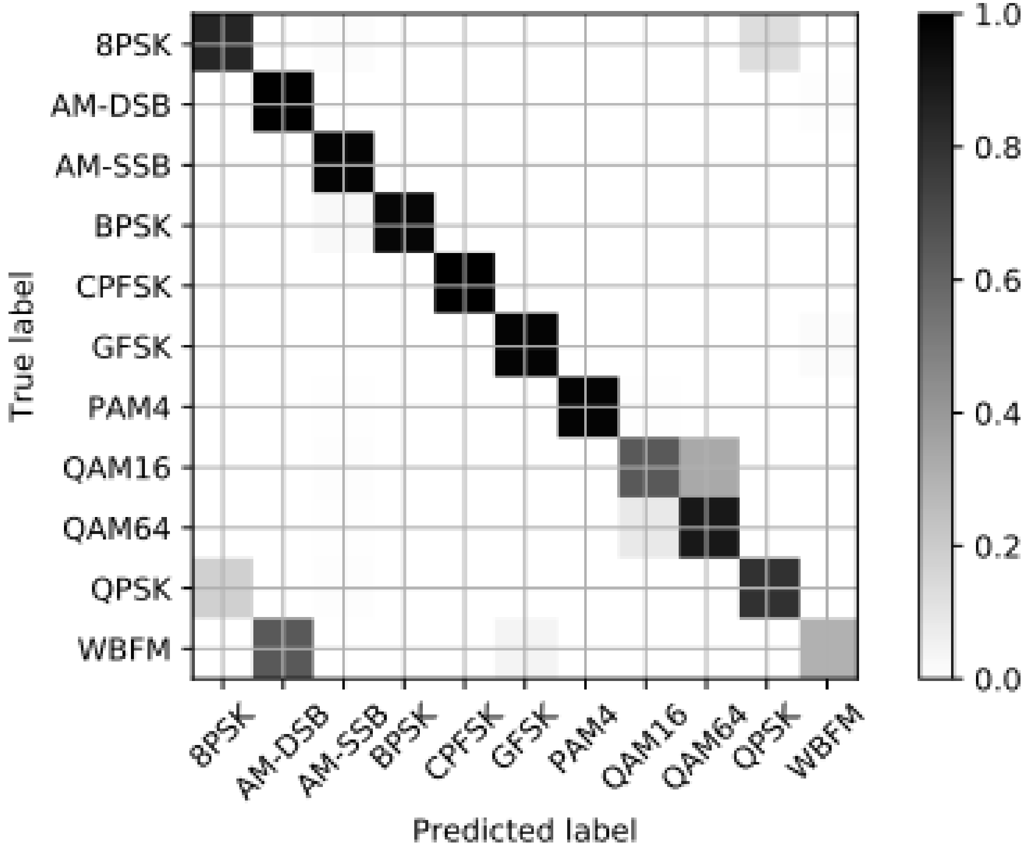
(a)

Analysis and results

- Performance results for modulation recognition classifiers (SNR=6db, phase amplitude vector)

FIGURE 7. Confusion matrices for the modulation recognition data for SNR 6dB. (a) $\text{CNN}^{M}_{\mathcal{IQ}}$. (b) $\text{CNN}^{M}_{\mathbf{A}/\Phi}$. (c) $\text{CNN}^{M}_{\mathcal{F}}$.



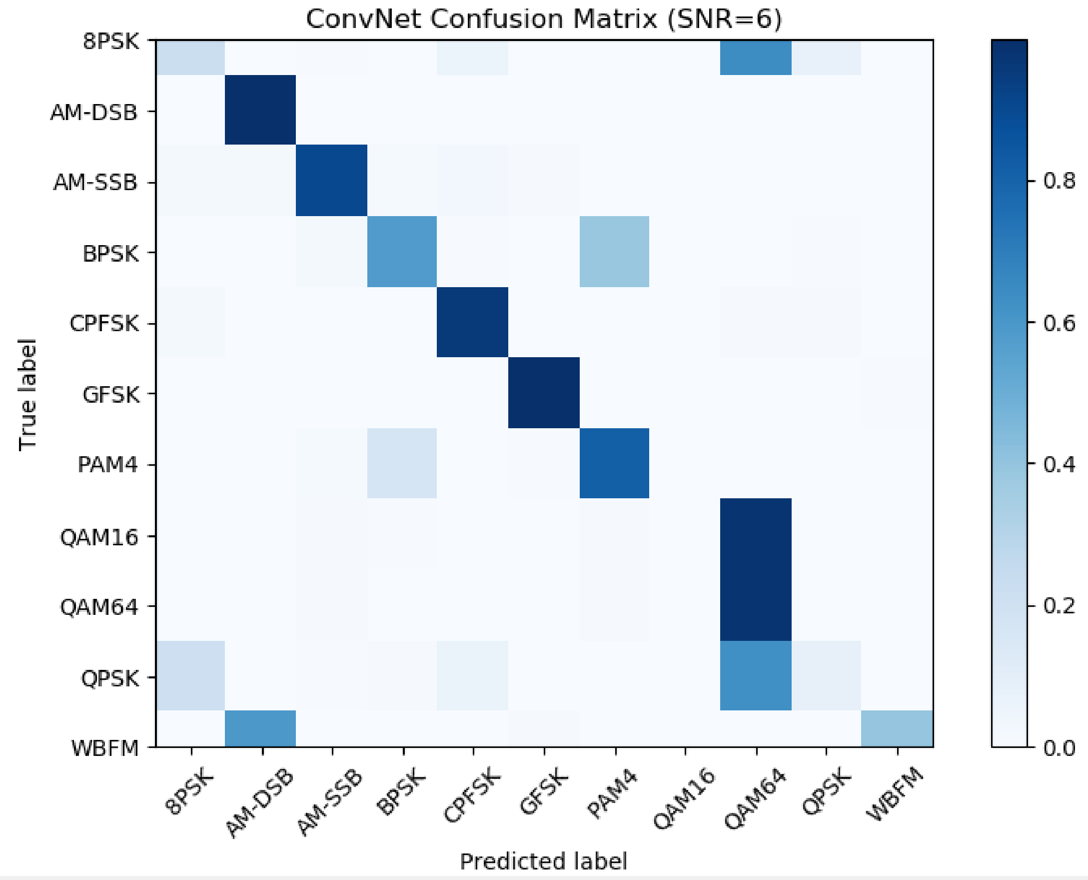


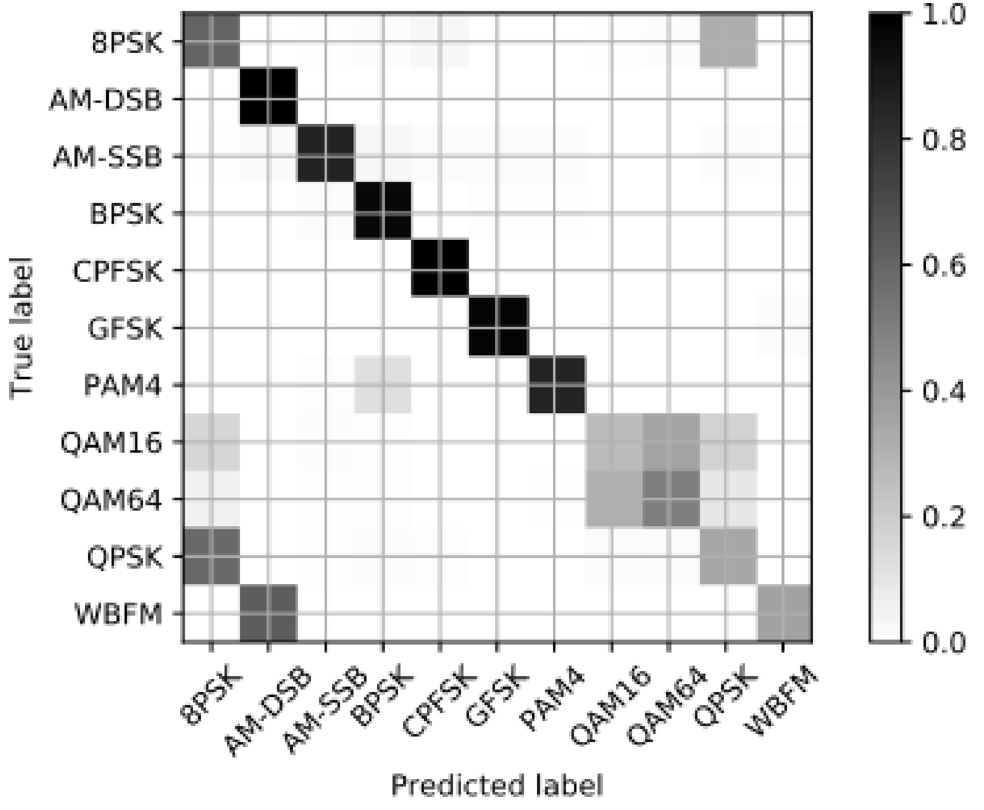
(b)

Analysis and results

- Performance results for modulation recognition classifiers vs. SNR.

FIGURE 7. **Confusion matrices for the modulation recognition data for SNR 6dB. (a) $CNN^{M}_{\mathcal{IQ}}$. (b) $CNN^{M}_{\mathbf{A}/\phi}$. (c) $CNN^{M}_{\mathcal{F}}$.**

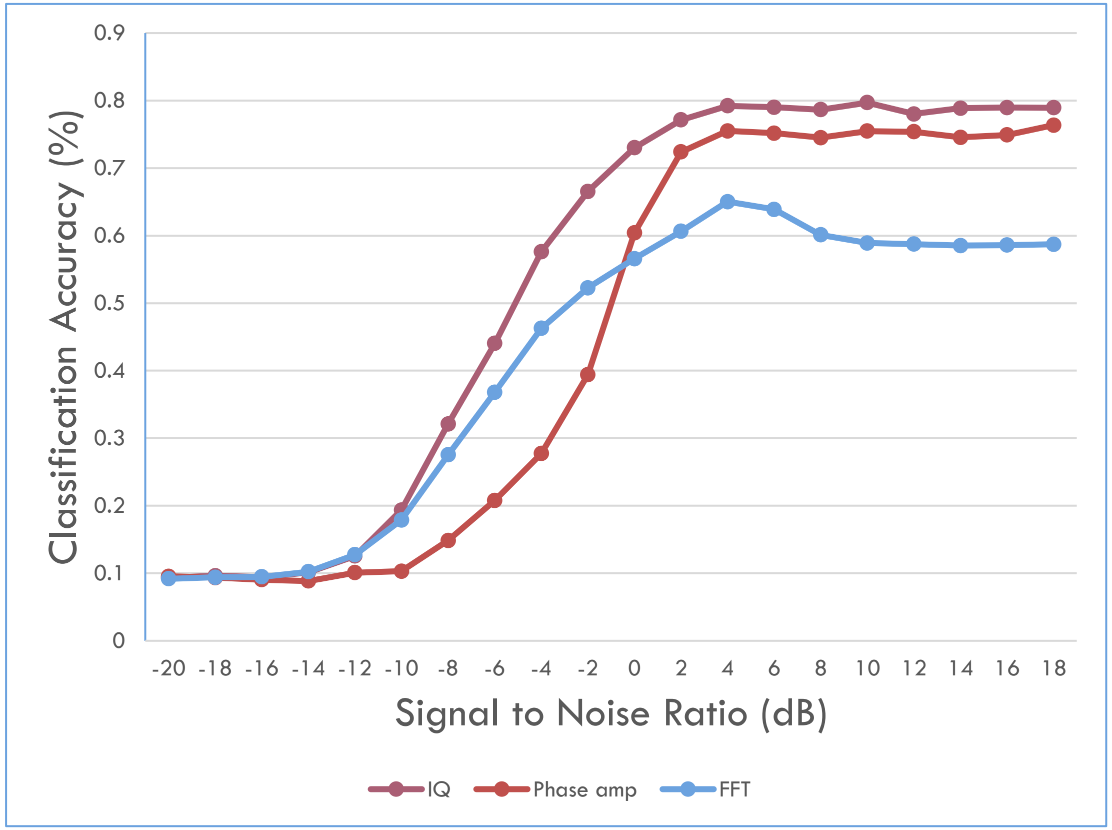


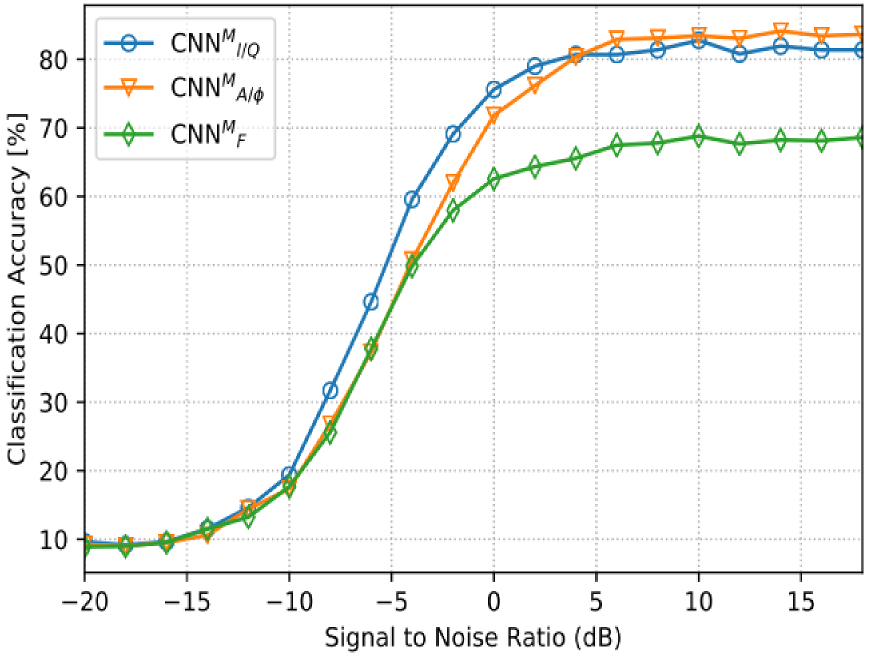


(c)

Analysis and results

- Performance results for modulation recognition classifiers vs. SNR.
- (정확도 지표는 종합 결과를 엑셀로 모두 취합하여 따로 그림)





느낀점

- 통신 도메인 분야는 처음이라 관련 내용부터 이해의 어려움이 있었음
- 구현하면서 다양한 분야에 활용될 수 있다는 것을 다시 한번 깨닫게 됨