Introduction to Deep Learning for the Physical Layer

Emmanuel Obeng Frimpong

Intelligent Communications and Information Security Lab

22nd July 2021

HANBAT NATIONAL UNIVERSITY

CONTENTS

Introduction

- This paper present several novel applications of deep learning for the physical layer.
- In a field of communications there is rich expert knowledge and mathematical model that can achieve relatively optimal performance.
- There is a high performance over which any Machine/Deep Learning based approach must pass in order to provide tangible new benefits.

Paper Contributions

- It is possible to learn full transmitter and receiver implementations for a given channel model which are optimized for a chosen loss function (e.g., minimizing block error rate (BLER)).
- Extended this concept to an adversarial network of multiple transmitter-receiver pairs competing for capacity.
- Introduced radio transformer networks (RTNs) to integrate expert knowledge into the DL model.
- CNN based modulation classification

Deep Learning Basics

- A feedforward NN with L layers describes a mapping $f(r_o; \theta)$ of an input vector r_o to an output vector r_L through L iterative processing steps:
  - $r_L = f_l(r_{l-1}; \theta_l), \quad l = 1, \dots, L$
- The lth layer is called dense or fully connected if
  - $f_l(r_{l-1}; \theta_l) = \sigma(W_l r_l + b_l)$
- $\sigma(\cdot)$ is activation function

Deep Learning Basics

- The goal of the training process is to minimize the loss:

$$L(\boldsymbol{\theta}) = \frac{1}{S}\sum_{i=1}^{S} l(\mathbf{r}^{\star}_{L,i}, \mathbf{r}_{L,i})$$

Table I: List of layer types

Name	$f_\ell(\mathbf{r}_{\ell-1};\theta_\ell)$	θ_ℓ
Dense	$\sigma\left(\mathbf{W}_\ell \mathbf{r}_{\ell-1} + \mathbf{b}_\ell\right)$	$\mathbf{W}_\ell, \mathbf{b}_\ell$
Noise	$\mathbf{r}_{\ell-1} + \mathbf{n}$, $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \beta\mathbf{I}_{N_{\ell-1}})$	none
Dropout [36]	$\mathbf{d} \odot \mathbf{r}_{\ell-1}$, $d_i \sim \text{Bern}(\alpha)$	none
Normalization	e.g., $\frac{\sqrt{N_{\ell-1}}\mathbf{r}_{\ell-1}}{\lVert\mathbf{r}_{\ell-1}\rVert_2}$	none

Table II: List of activation functions

Name	$[\sigma(\mathbf{u})]_{\mathbf{i}}$	Range
linear	u_i	$(-\infty, \infty)$
ReLU [37]	$\max(0, u_i)$	$[0, \infty)$
tanh	$\tanh(u_i)$	$(-1, 1)$
sigmoid	$\frac{1}{1+e^{-u_i}}$	$(0, 1)$
softmax	$\frac{e^{u_i}}{\sum_j e^{u_j}}$	$(0, 1)$

Table III: List of loss functions

Name	$l(\mathbf{u}, \mathbf{v})$
MSE	$\lVert\mathbf{u} - \mathbf{v}\rVert_2^2$
Categorical cross-entropy	$-\sum_j u_j \log(v_j)$

Autoencoders for end-to-end communications system

- From a DL point of view, this simple communications system can be seen as a particular type of autoencoder.

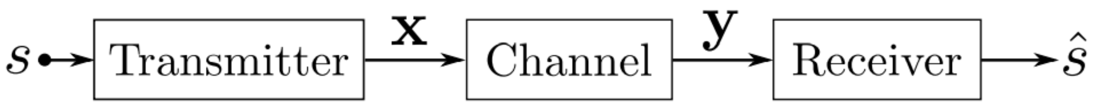


Figure 1: A simple communications system consisting of a transmitter and a receiver connected through a channel

Autoencoders for end-to-end communications system

5

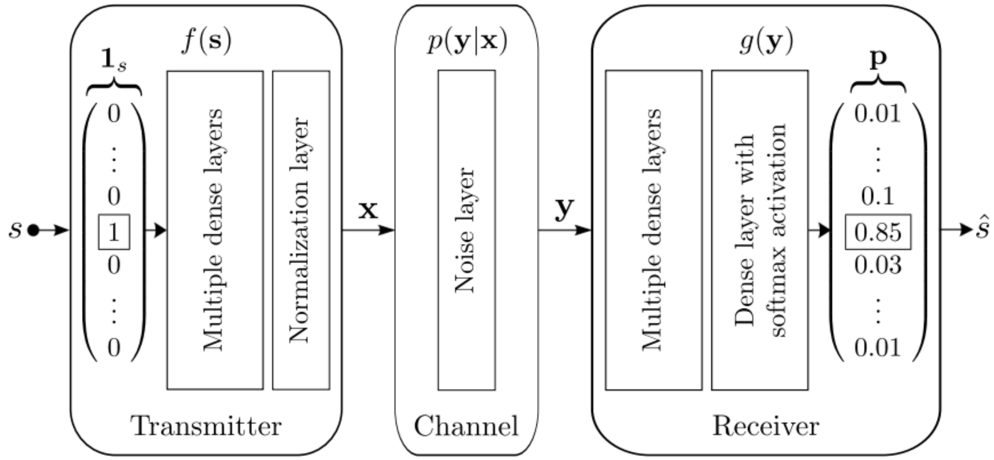


Figure 2: A communications system over an AWGN channel represented as an autoencoder. The input s is encoded as a one-hot vector, the output is a probability distribution over all possible messages from which the most likely is picked as output $\hat{s}$.

Table IV: Layout of the autoencoder used in Figs. 3a and 3b. It has $(2M+1)(M+n)+2M$ trainable parameters, resulting in 62, 791, and 135,944 parameters for the (2,2), (7,4), and (8,8) autoencoder, respectively.

Layer	Output dimensions
Input	M
Dense + ReLU	M
Dense + linear	n
Normalization	n
Noise	n
Dense + ReLU	M
Dense + softmax	M

SIMULATION RESULTS / ANALYSIS

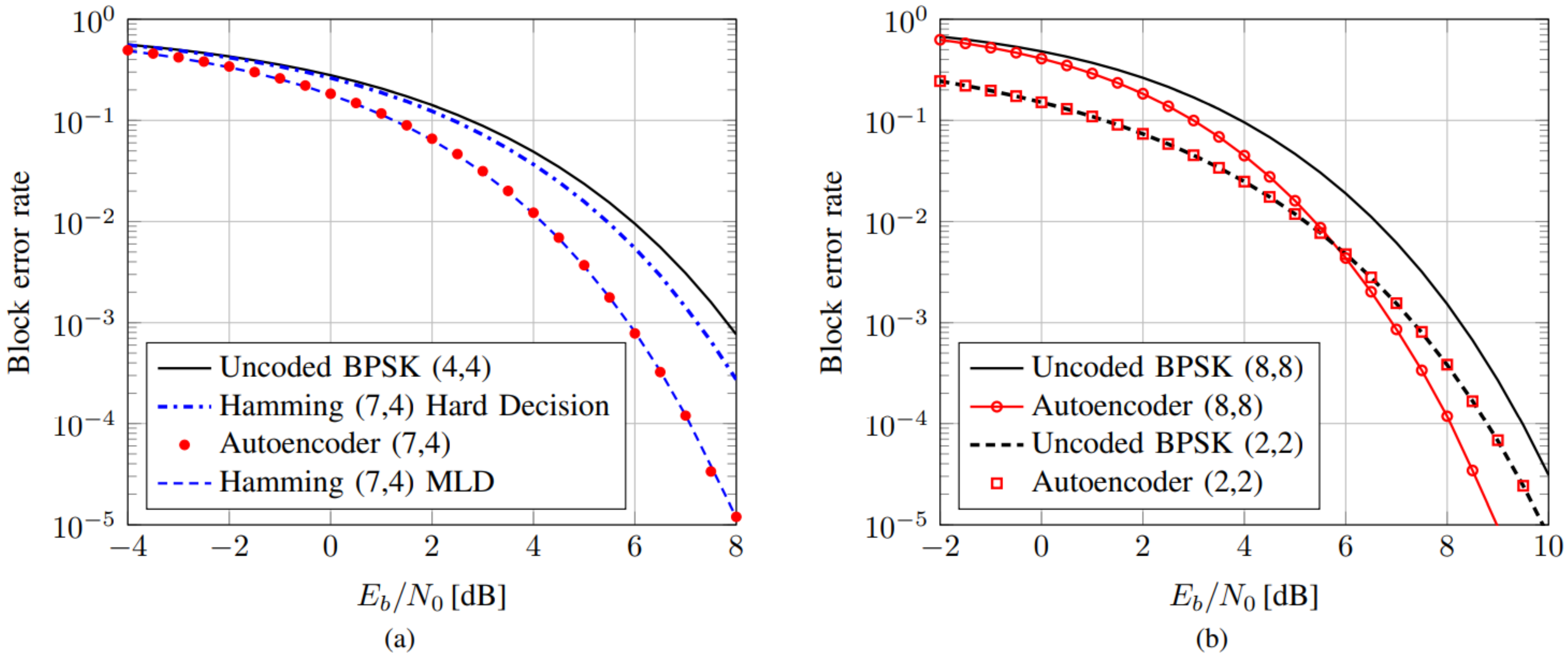


Figure 3: BLER versus E_b/N_0 for the autoencoder and several baseline communication schemes

SIMULATION RESULTS / ANALYSIS

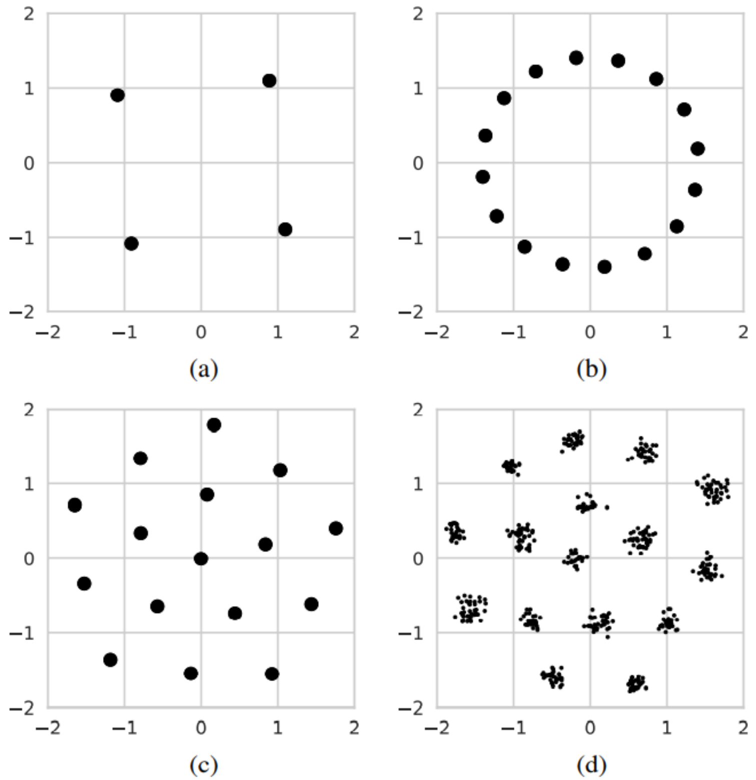


Figure 4: Constellations produced by autoencoders using parameters (n, k): (a) $(2, 2)$ (b) $(2, 4)$, (c) $(2, 4)$ with average power constraint, (d) $(7, 4)$ 2-dimensional t-SNE embedding of received symbols.

Autoencoders for multiple transmitters and receivers

- The autoencoder concept can be readily extended to multiple transmitters and receivers that share a common channel.

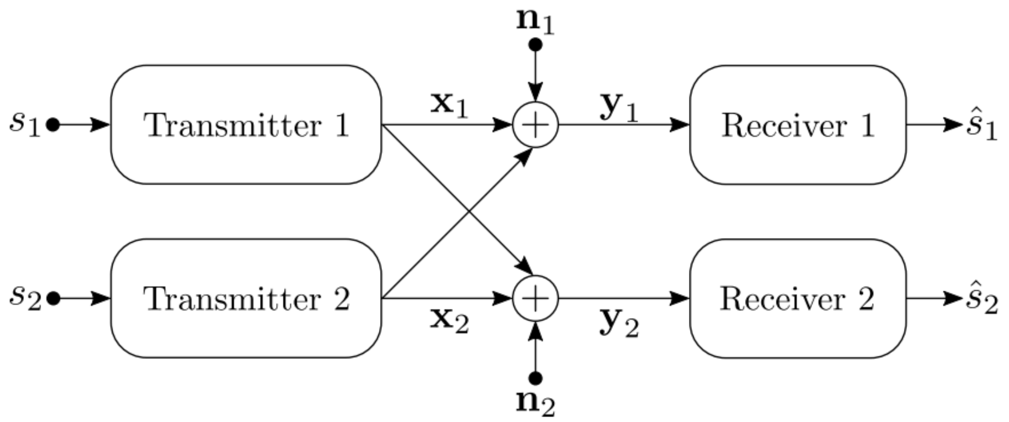


Figure 5: The two-user interference channel seen as a combination of two interfering autoencoders that try to reconstruct their respective messages

$$\mathbf{y}_1 = \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{n}_1$$
$$\mathbf{y}_2 = \mathbf{x}_2 + \mathbf{x}_1 + \mathbf{n}_2$$

Loss Function Analysis

- The individual cross-entropy loss functions of the first and second transmitter-receiver pair:

$$l_1 = -\log\left([\hat{\mathbf{s}}_1]_{s_1}\right), \quad l_2 = -\log\left([\hat{\mathbf{s}}_2]_{s_2}\right)$$

- It is less clear how one should train two coupled autoencoders with conflicting goals. One approach consists of minimizing a weighted sum of both losses.

$$\tilde{L} = \alpha \tilde{L}_1 + (1-\alpha)\tilde{L}_2 \text{ for some } \alpha \in [0,1]$$

$$\alpha_{t+1} = \frac{\tilde{L}_1(\boldsymbol{\theta}_t)}{\tilde{L}_1(\boldsymbol{\theta}_t) + \tilde{L}_2(\boldsymbol{\theta}_t)}, \quad t > 0$$

SIMULATION RESULTS / ANALYSIS

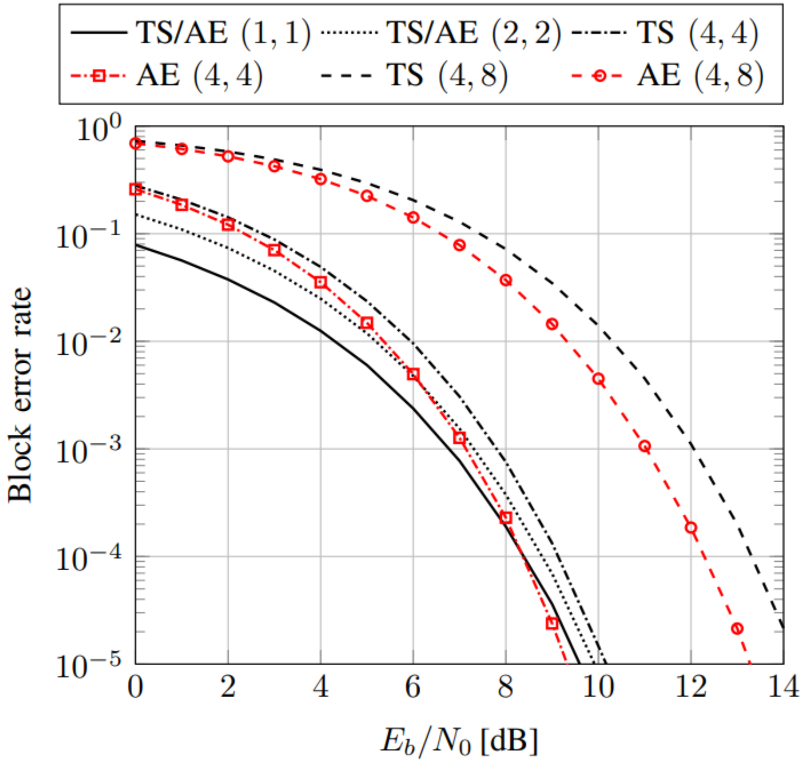


Figure 6: BLER versus E_b/N_0 for the two-user interference channel achieved by the autoencoder (AE) and $2^{2k/n}$-QAM time-sharing (TS) for different parameters (n, k)

SIMULATION RESULTS / ANALYSIS

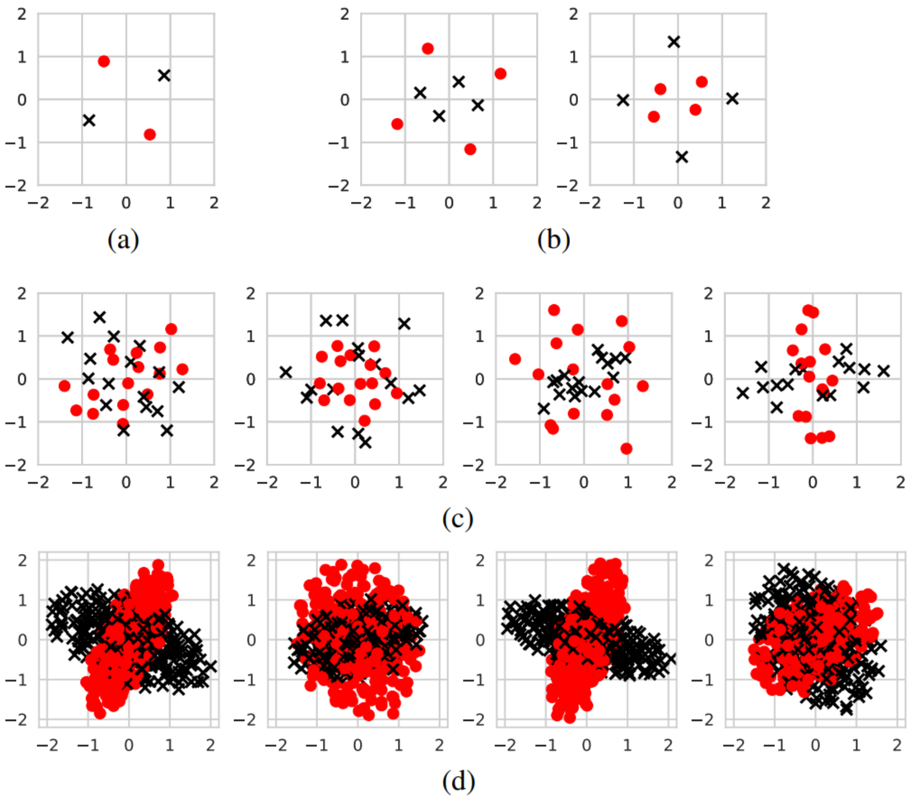


Figure 7: Learned constellations for the two-user interference channel with parameters (a) $(1, 1)$, (b) $(2, 2)$, (c) $(4, 4)$, and (d) $(4, 8)$. The constellation points of Transmitter 1 and 2 are represented by red dots and black crosses, respectively.

RTNs for augmented signal processing algorithms

- Many of the physical phenomena undergone in a communications channel and in transceiver hardware can be inverted using compact parametric models/transformations.
- The estimation processes for parameters to seed these transformations is often very involved and specialized based on signal specific properties and/or information from pilot tones.
- One way of augmenting DL models with expert propagation domain knowledge but not signal specific assumptions is using an RTN.

RTNs for augmented signal processing algorithms

- This RTN is described for receiver-side processing, but it can similarly be used wherever parametric transformations seeded by estimated parameters are needed.

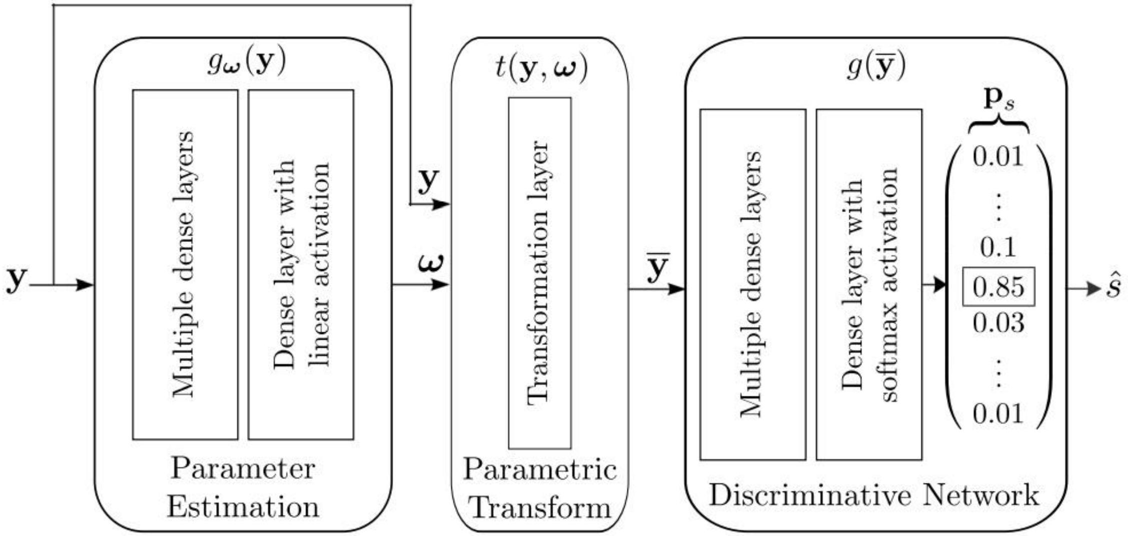


Figure 8: A radio receiver represented as an RTN. The input $\mathbf{y}$ first runs through a parameter estimation network $g_\omega(\mathbf{y})$, has a known transform $t(\mathbf{y}, \boldsymbol{\omega})$ applied to generate the canonicalized signal $\overline{\mathbf{y}}$, and then is classified in the discriminative network $g(\overline{\mathbf{y}})$ to produce the output $\hat{s}$.

RTN OPERATION

- The basic functioning of an RTN is best understood from a simple example, such as the problem of phase offset estimation and compensation

Let $\mathbf{y}_c = e^{j\varphi}\tilde{\mathbf{y}}_c \in \mathbb{C}^n$ be a vector of IQ samples that have undergone a phase rotation by the phase offset φ, and let $\mathbf{y} = [\Re\{\mathbf{y}\}^\mathsf{T}, \Im\{\mathbf{y}\}^\mathsf{T}]^\mathsf{T} \in \mathbb{R}^{2n}$. The goal of $g_{\boldsymbol{\omega}}$ is to estimate a scalar $\hat{\varphi} = \boldsymbol{\omega} = g_{\boldsymbol{\omega}}(\mathbf{y})$ that is close to the phase offset φ, which is then used by the parametric transform t to compute $\bar{\mathbf{y}}_c = e^{-j\hat{\varphi}}\mathbf{y}_c$. The canonicalized signal $\bar{\mathbf{y}} = [\Re\{\bar{\mathbf{y}}_c\}^\mathsf{T}, \Im\{\bar{\mathbf{y}}_c\}^\mathsf{T}]^\mathsf{T}$ is thus given by

$$\bar{\mathbf{y}} = t(\hat{\varphi}, \mathbf{y}) = \begin{bmatrix} \cos(\hat{\varphi})\Re\{\bar{\mathbf{y}}_c\} + \sin(\hat{\varphi})\Im\{\bar{\mathbf{y}}_c\} \\ \cos(\hat{\varphi})\Im\{\bar{\mathbf{y}}_c\} - \sin(\hat{\varphi})\Re\{\bar{\mathbf{y}}_c\} \end{bmatrix} \qquad (11)$$

and then fed into the discriminative network g for further processing, such as classification.

SIMULATION RESULTS / ANALYSIS

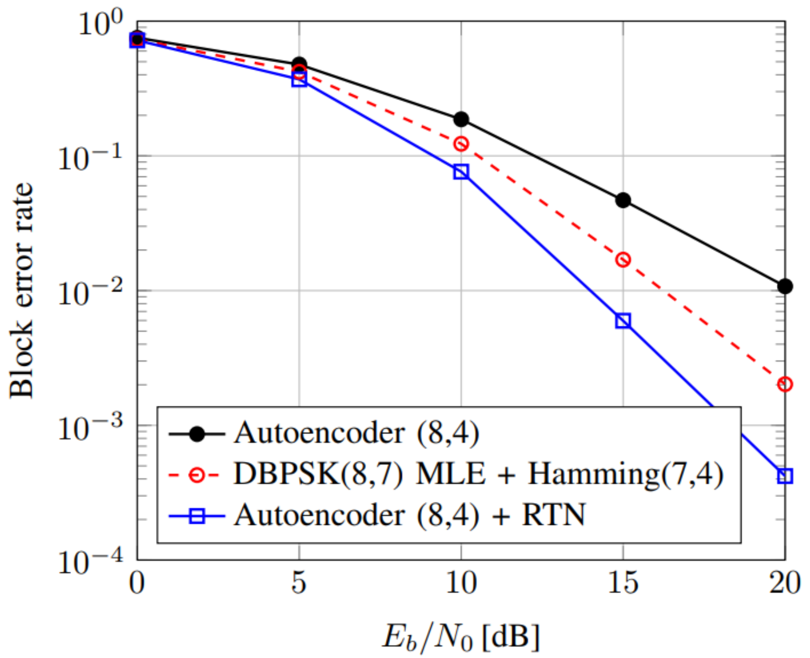


Figure 9: BLER versus E_b/N_0 for various communication schemes over a channel with $L = 3$ Rayleigh fading taps

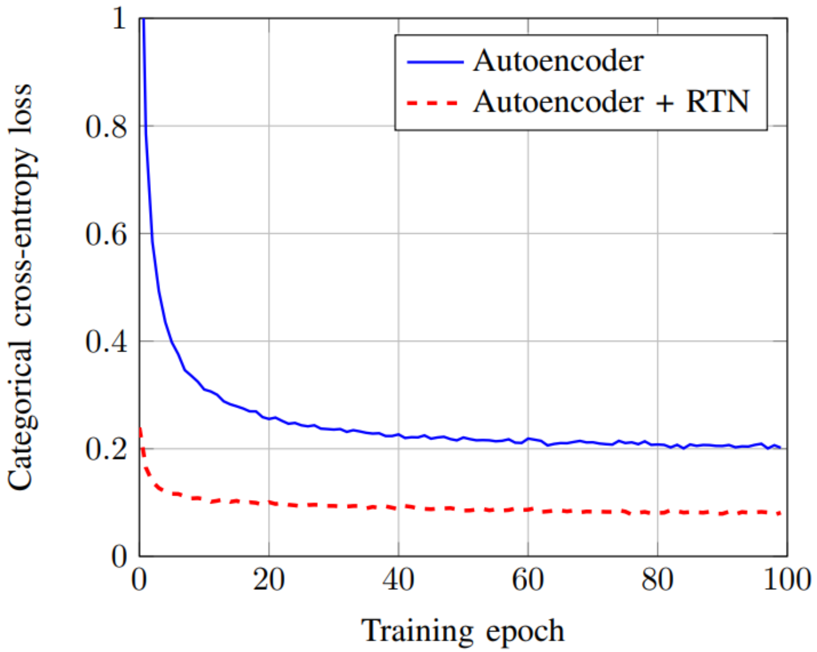


Figure 10: Autoencoder training loss with and without RTN

CNNs for classification tasks

- Paper looks at the well-known problem of modulation classification of single carrier modulation schemes based on sampled radio frequency time-series data.
- This task has been accomplished through support vector machines, random forests, or small feedforward NNs.
- None have sought to use feature learning on raw time-series data in the radio domain.
- This is however now the norm in computer vision which motivates our approach here.

CNNs Layout And Simulation Results

Table V: Layout of the CNN for modulation classification with 324,330 trainable parameters

Layer	Output dimensions
Input	2×128
Convolution (128 filters, size 2×8) + ReLU	128×121
Max Pooling (size 2, strides 2)	128×60
Convolution (64 filters, size 1×16) + ReLU	64×45
Max Pooling (size 2, strides 2)	64×22
Flatten	1408
Dense + ReLU	128
Dense + ReLU	64
Dense + ReLU	32
Dense + softmax	10

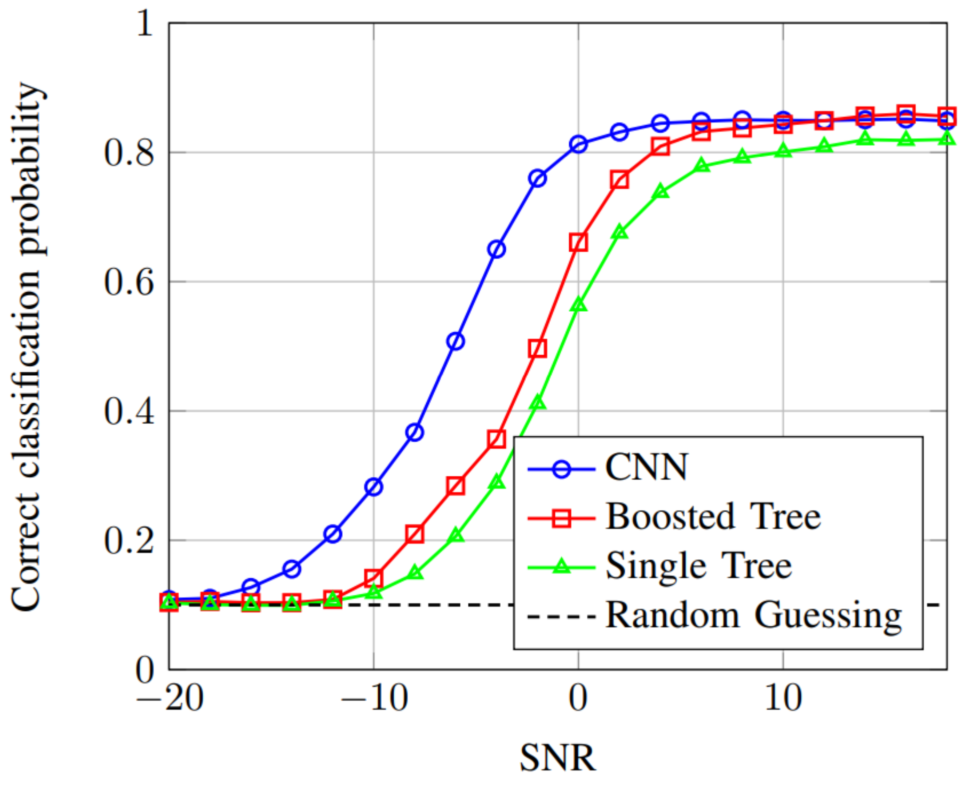


Figure 11: Classifier performance comparison versus SNR

CNN Metric ~ Confusion Matrix

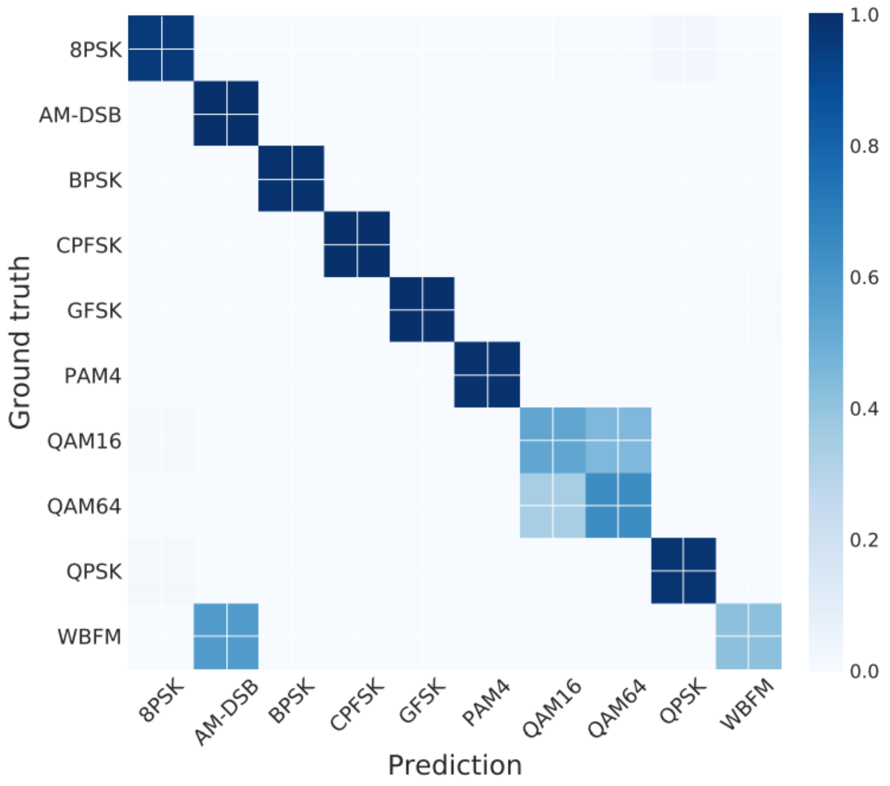
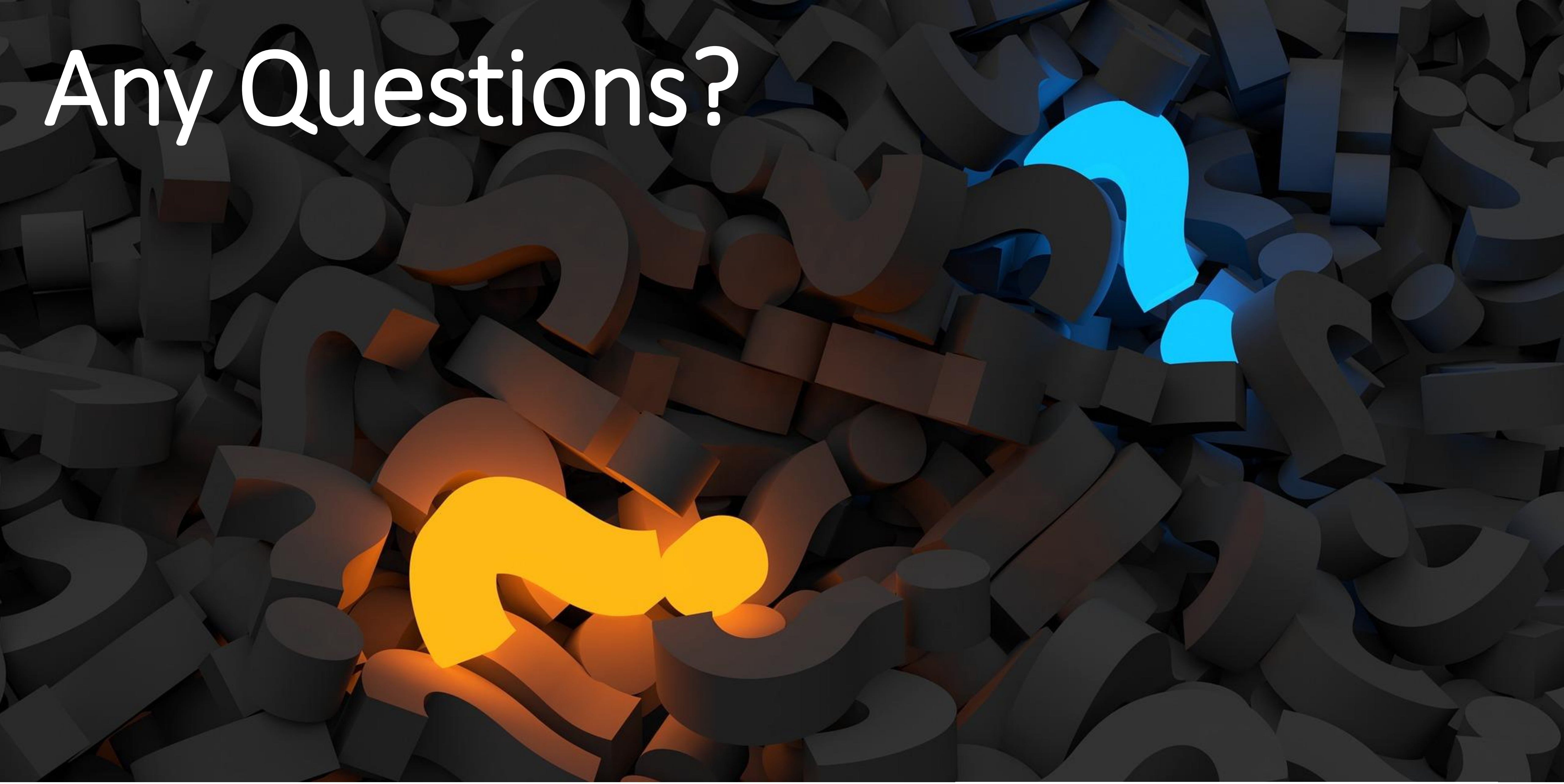


Figure 12: Confusion matrix of the CNN (SNR = 10 dB)

Research Challenges

- Data sets and challenges
- Data representation, loss functions, and training SNR
- Complex-valued neural networks
- ML-augmented signal processing
- System identification for end-to-end learning
- Learning from CSI and beyond

Any Questions?



HANBAT NATIONAL UNIVERSITY

frimpongemmanuelobeng@gmail.com