BottleNet++: An End-to-End Approach for Feature Compression in Device-Edge Co-Inference Systems

J. Shao and J. Zhang, "BottleNet++: An End-to-End Approach for Feature Compression in Device-Edge Co-Inference Systems," *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, 2020, pp. 1-6. (cited by 50)

Advisor: Dr. Chih-Yu Wang

Presenter: Shao-Heng Chen

Date: April 06, 2022

Outline

1. Overview
2. BottleNet++ architecture
3. Network components and training strategy
4. Evaluations
5. Conclusions and future works

Overview

1. Challenges for co-inference?

   – Network splitting, and feature compression

2. Why BottleNet++?

   – Feature compression and transmission vs. Feature coding

   – Fault-tolerance property and JSCC [1] vs. Source coding

3. How to model the wireless channel?

   – As a non-trainable layer [2]

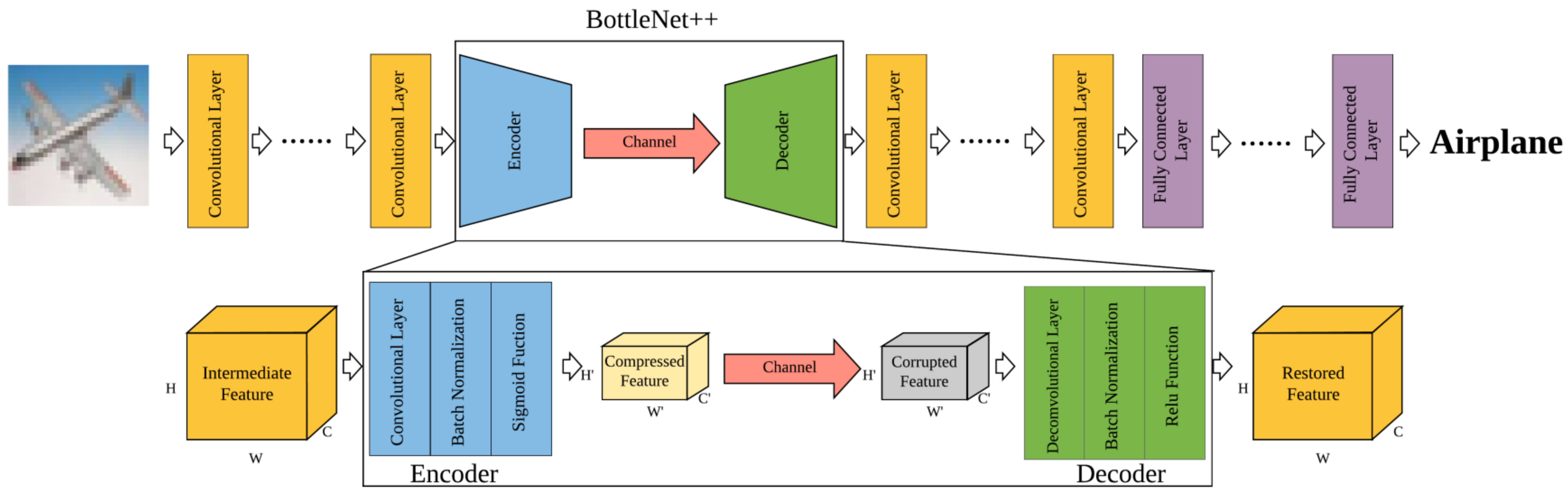
[1] K. Choi, K. Tatwawadi, A. Grover, T. Weissman and S. Ermon, "Neural joint source-channel coding", *International Conference on Machine Learning*, pp. 1182-1192, 2019. (cited by 44)

[2] E. Bourtsoulatze, D. B. Kurka and D. Gündüz, "Deep joint sourcechannel coding for wireless image transmission", *IEEE Transactions on Cognitive Communications and Networking*, 2019. (cited by 150)

BottleNet++ Architecture

– End-to-end architecture

– Encoder + Channel model + Decoder



Encoder

– Feature Compression, Joint Source-Channel Coding (JSCC)

– Compress C channels to C’ channels

(1) A conv. layer

(2) A batch normalization layer

(3) A Sigmoid activation layer

$$feature\ tensor\ (channel, width, height)\ size = (C, W, H)$$

$$kernel\ size = 2 \times 2$$

$$C\ \ filters\ with\ stride\ size\ \left(\left\lfloor\frac{W}{W'}\right\rfloor, \left\lfloor\frac{H}{H'}\right\rfloor\right) = (2, 2)$$

Channel Model

(1)AWGN channel [2]

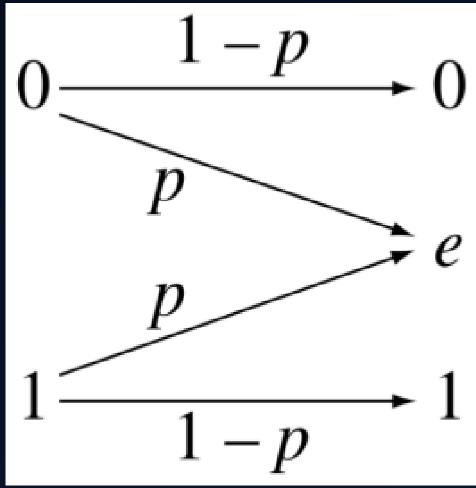
$- f(x) = x + n, n \sim N(0, \sigma^2)$

(2)Binary Erasure Channel [1]

– Bit Erasure Rate p model the deep fades and burst errors

– Network converges when p is not too large

$- quantizer\ \tilde{X} = round\left(X \cdot (2^n - 1)\right) / (2^n - 1)$



[1] K. Choi, K. Tatwawadi, A. Grover, T. Weissman and S. Ermon, "Neural joint source-channel coding", *International Conference on Machine Learning*, pp. 1182-1192, 2019. (cited by 44)

[2] E. Bourtsoulatze, D. B. Kurka and D. Gündüz, "Deep joint sourcechannel coding for wireless image transmission", *IEEE Transactions on Cognitive Communications and Networking*, 2019. (cited by 150)

Decoder

– Restore the bitstream to feature tensor

– Recover C’ channels to C channels

(1) A deconv. layer

(2) A batch normalization layer

(3) A ReLU activation layer

$$feature\ tensor\ (channel, width, height)\ size = (C', W', H')$$

$$kernel\ size = 2 \times 2$$

$$C\ filters\ with\ stride\ size\ \left(\left\lfloor \frac{W}{W'} \right\rfloor, \left\lfloor \frac{H}{H'} \right\rfloor\right) = (2, 2)$$

Training Strategy

– Direct training may cause slow convergence

(1) Train the DNN

(2) Train the encoder/decoder

(3) Fine-tuning whole network

– Model the wireless channel as a non-trainable layer [3]

– Only need to retrain the JSCC module for different channel conditions

[3] A. E. Eshratifar, A. Esmaili and M. Pedram, "BottleNet: A Deep Learning Architecture for Intelligent Mobile Cloud Computing Services," 2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED), 2019, pp. 1-6 (cited by 58)

Experimental settings

– Image classification on CIFAR-100, 100 class of 60000 32 x 32 color images

– Main branch: VGG16, ResNet50

– AWGN channel with $PSNR(dB) = 10log_{10}\left(\frac{1}{\sigma^2}\right)(dB) = 20\ dB$

– BEC with $Bit\ Erasure\ Rate\ (BER) = p = 0.01$

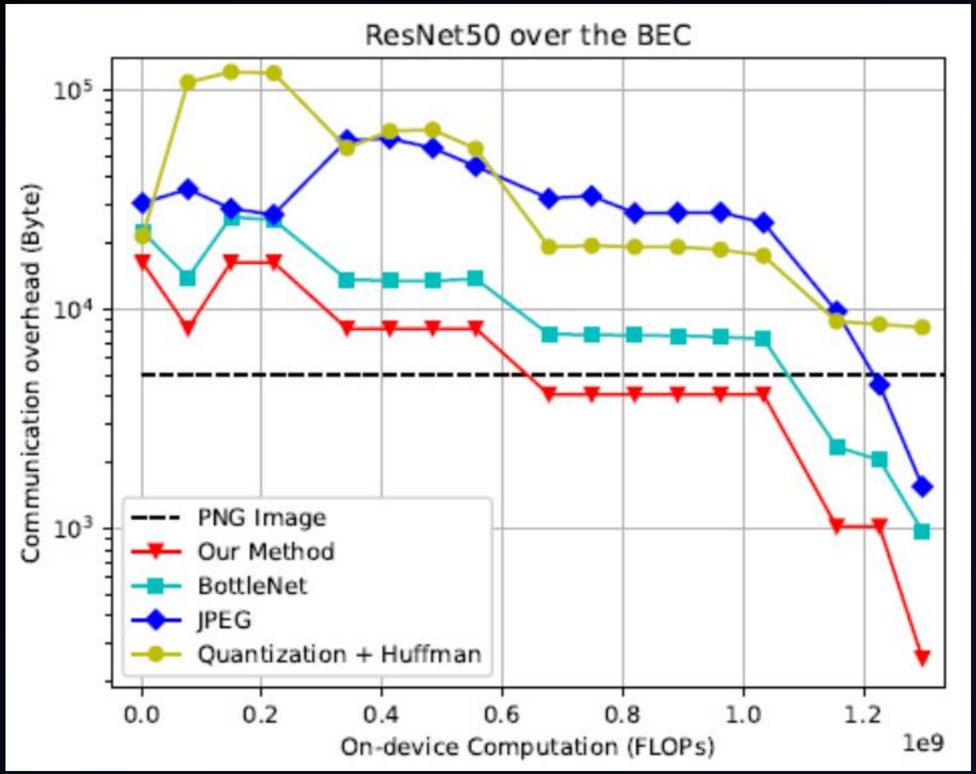
– Note.

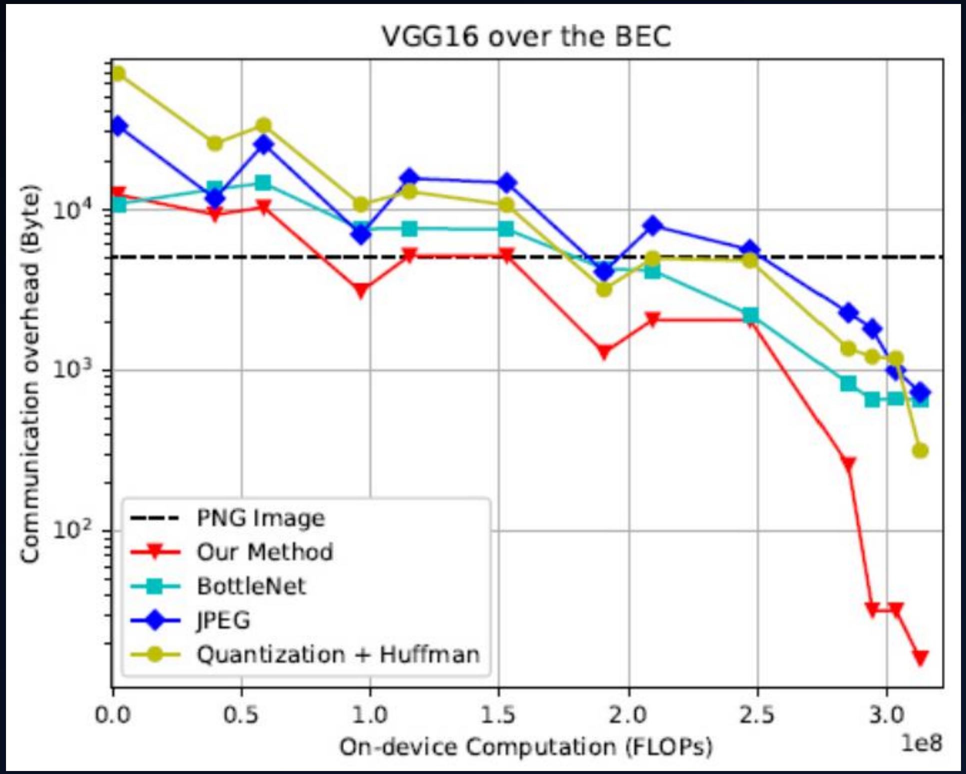
(1) Not all layers can be use as a splitting point

(2) FC layer, Conv. layer can be regarded as one

Experimental results

– On-device Computational vs. Communication Overhead in BEC





$Fig. 2. On - device\ Computation\ vs. Communication\ Overhead\ in\ BEC\ with\ (a)\ ResNet50\ and\ (b)\ VGG16$

Experimental results

– Minimum On-device Comp. with Comm. Overhead less than raw data

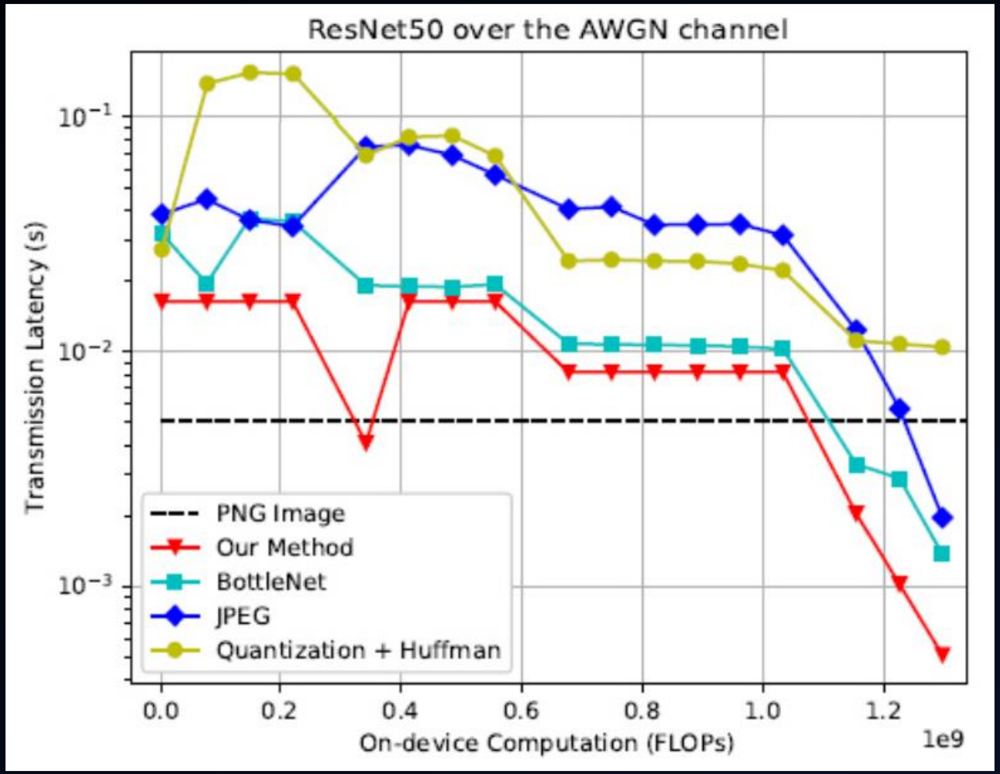
On-device Computation (FLOPs)	BottleNet++ (ours)	BottleNet	JPEG	Quan.+Huffman
ResNet50 (BEC)	$\mathbf{6.8 \times 10^8}$	1.1×10^9	1.2×10^9	1.3×10^9
ResNet50 (AWGN)	$\mathbf{3.4 \times 10^8}$	1.1×10^9	1.3×10^9	1.3×10^9
VGG16 (BEC)	$\mathbf{9.6 \times 10^7}$	1.9×10^8	1.9×10^8	1.9×10^8
VGG16 (AWGN)	$\mathbf{9.6 \times 10^7}$	1.9×10^8	1.9×10^8	1.9×10^8

– 256× bit compression ratio, 32KB Float → 128 Bytes Integer

– 64× bandwidth reduction, 2048-symbol → 32-symbol

Experimental results

– On-device Computational vs. Transmission Latency in AWGN channel



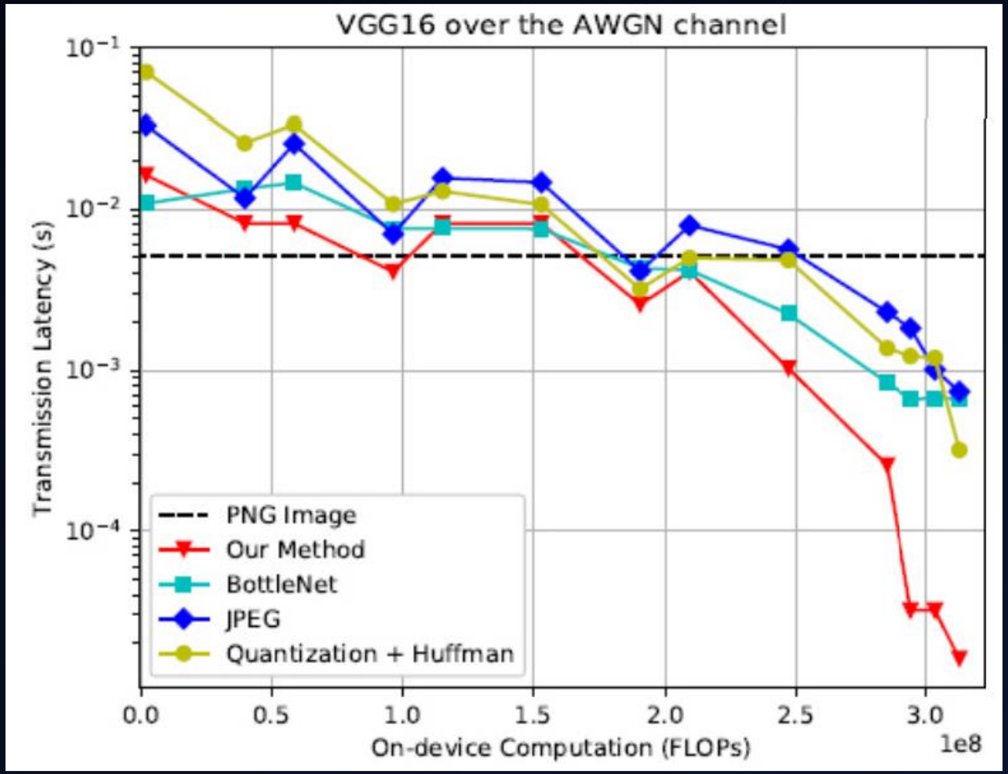


Fig. 2. On – device Computation vs. Transmission Latency in BEC with (c) ResNet50 and (d) VGG16

Experimental results

– Accuracy degradation for testing generalization ability

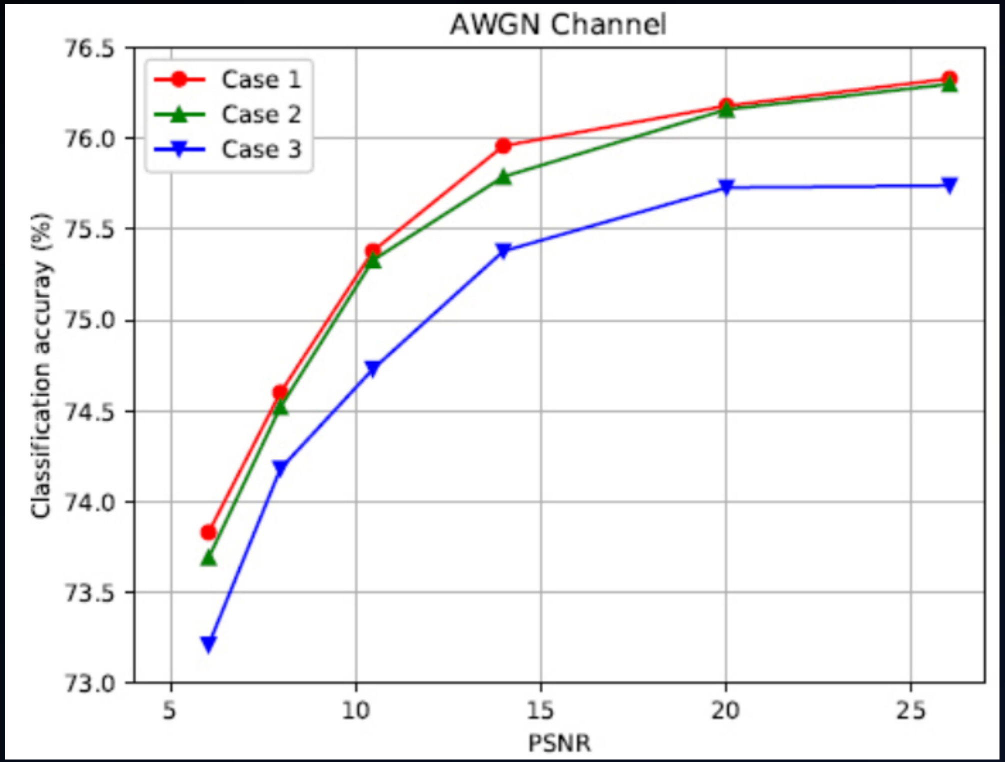


Fig. 3. Accuracy degradation over the (a) AWGN channel

– CSI (BER, PSNR) known or unknown

	training	testing
Case 1	O	O
Case 2	O	X
Case 3	X	X

Experimental results

– Accuracy degradation for testing generalization ability

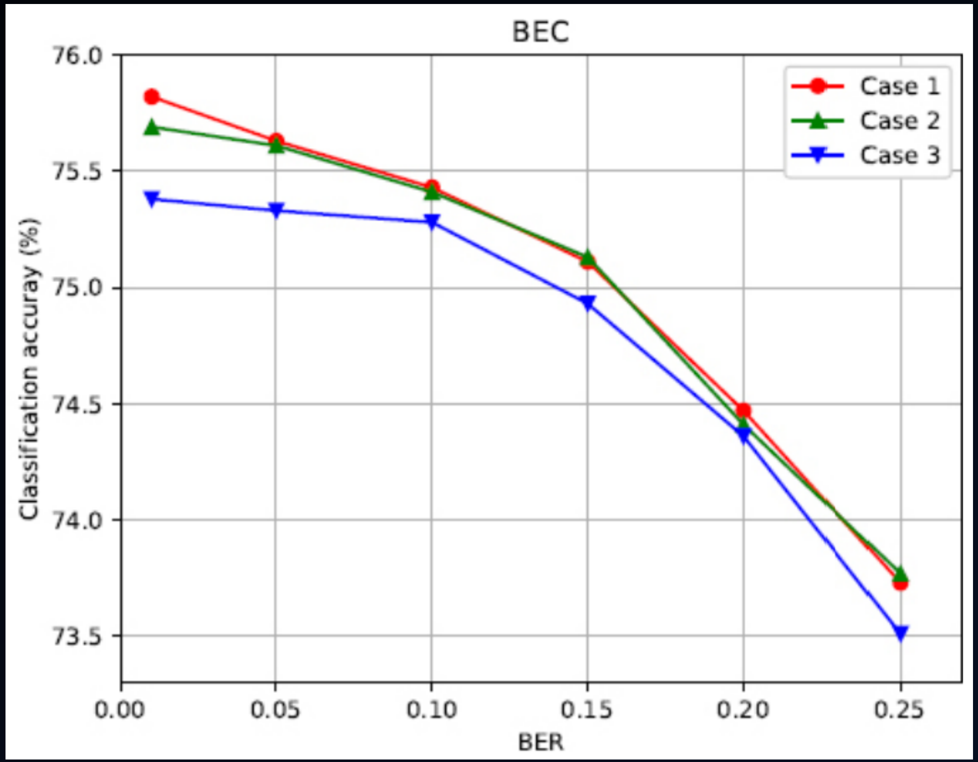


Fig. 3. Accuracy degradation over the (b) BEC

– CSI (BER, PSNR) known or unknown

	training	testing
Case 1	O	O
Case 2	O	X
Case 3	X	X

Conclusions and Future Works

– Useful insights

(1) Model the wireless channel as a non-trainable seems reasonable

(2) Not all layers can be use as a splitting point

– Ideas

(1) Maybe first test on DL_CFAR, before jump into YOLO_CFAR

(2) Need to figure out the reasonable setting for CFAR case