2019 ICASSP

Low-complexity Recurrent Neural Network-based Polar Decoder with Weight Quantization Mechanism

Speaker: Chieh-Fang (Jeff) Teng

Email: jeff@access.ee.ntu.edu.tw

Advisor: Prof. An-Yeu (Andy) Wu

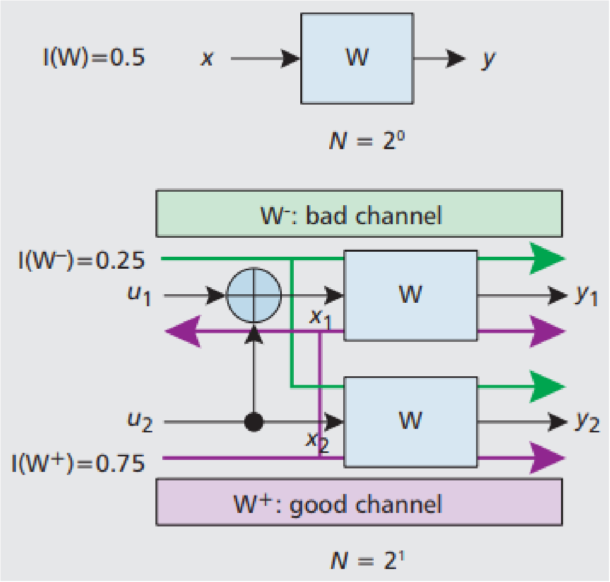
Date: 2019/05/14

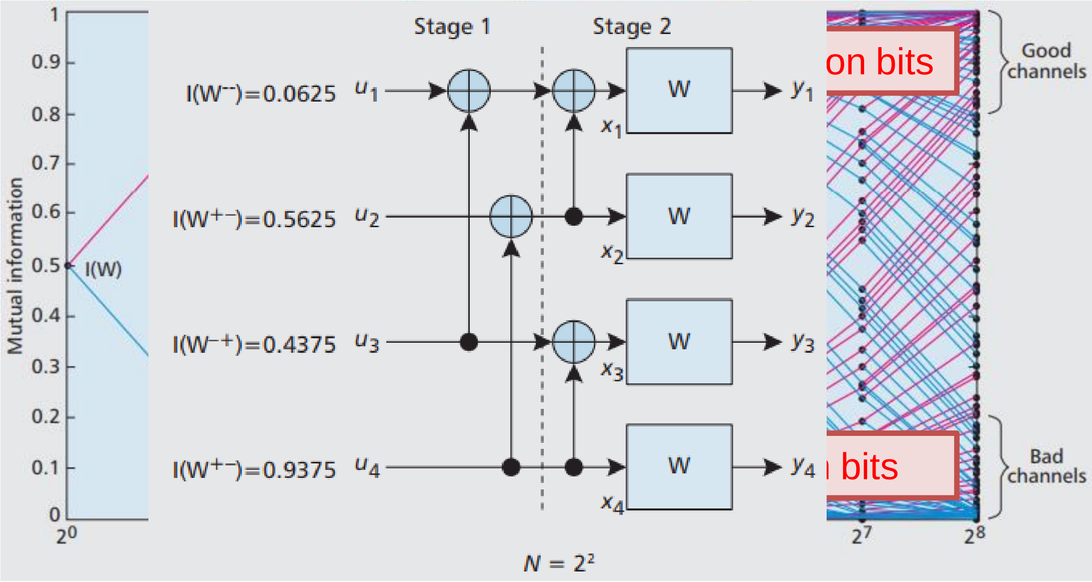
Outline

- Polar code
  - Encoding
  - Decoding: belief propagation
- Neural network polar decoder
- Motivation and proposed approach
  - Recurrent architecture
  - Codebook-based weight quantization
- Simulation results and analysis
- Conclusion

Polar Code [1-2]

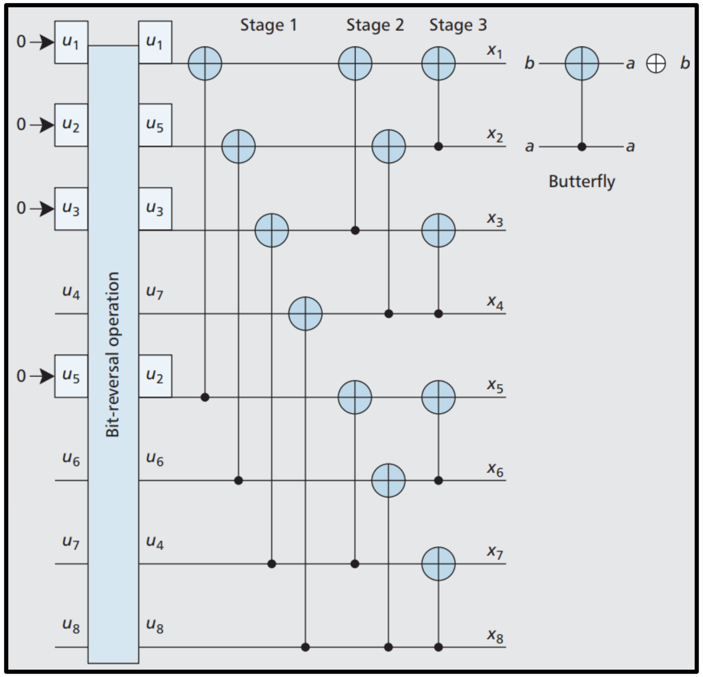
- Proposed by Arikan in 2009 with provable achievement of Shannon capacity given binary input discrete memoryless channel (B-DMC)
- Channel polarization
  - Matthew effect
  - With recursive implementation, good channels get better and the bad ones get worse





Polar Code: Encoding

- Code length: $N = 2^n, n = 1, 2, \ldots$
- Information length: K
- Code rate: $R = K/N$
- Frozen bits: $N - K$ fixed value of zeros known both by encoder and decoder



- $\boldsymbol{x}^N = \boldsymbol{u}^N \boldsymbol{G}_N = \boldsymbol{u}^N \boldsymbol{F}_2^{\otimes n} \boldsymbol{B}_N$
  - Codeword: $\boldsymbol{x}^N$
  - Binary source block: $\boldsymbol{u}^N = (u_1, u_2, \ldots, u_N)$
  - Generator matrix: $\boldsymbol{G}_N = \boldsymbol{F}_2^{\otimes n} \boldsymbol{B}_N$
  - $\boldsymbol{F}_2^{\otimes n}$: n-th Kronecker power of $\boldsymbol{F}_2 = \begin{bmatrix} 1 & 0 \\ 1 & 1 \end{bmatrix}$
  - $\boldsymbol{B}_N$: bit-reversal permutation matrix

Polar Code [1-9]

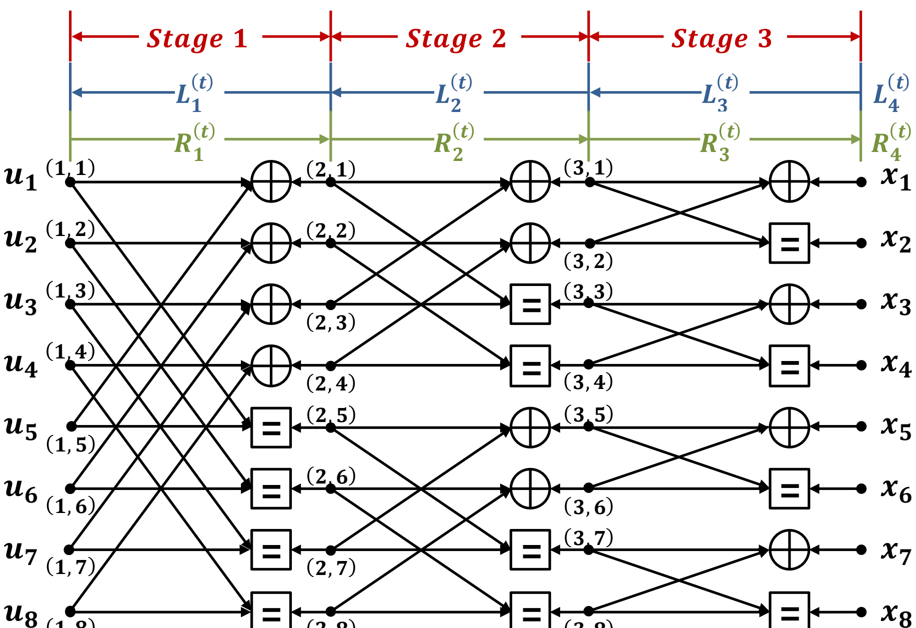
- Architecture flexibility
- Multi-code rate support
- Low cost of implementation
- Meet 5G communication protocol and adopted by 3GPP in 2016 for short codes used in control channel
- Other applications: error correction code in flash memory
- Decoding algorithm: successive cancelation and belief propagation [3-9]

	Successive Cancelation (SC)	Belief Propagation (BP)
Performance	High	Low
Complexity	Low	High
Latency	High	Low
Throughput	Low	High

Polar Code: Belief Propagation [8]

- Message passing algorithm for decoding
- Iterative processing over the factor graph of (N, K) polar code
  - Left-to-right message: $R_{i,j}^{(t)}$, j-th node at the i–th stage
  - Right-to-left message: $L_{i,j}^{(t)}$, j–th node at the i–th stage

$$\hat{u}_j^N = \begin{cases} 0, if\, L_{1,j}^T \geq 0 \\ 1, if\, L_{1,j}^T < 0 \end{cases}$$



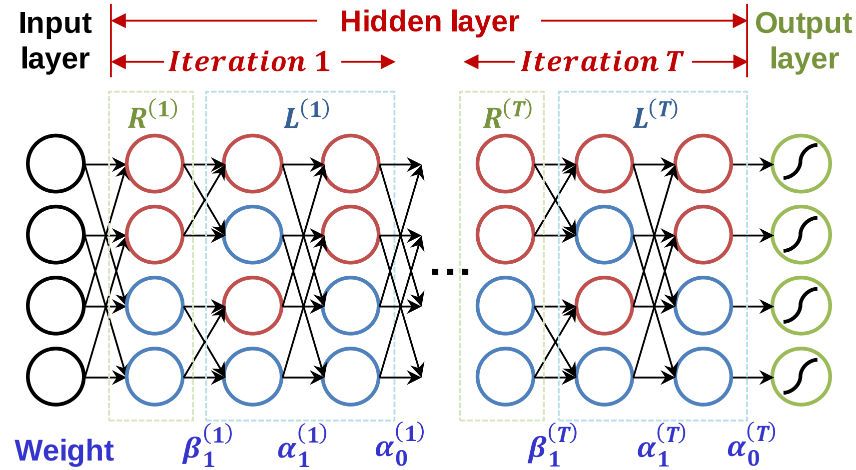
Unified scaled min-sum with $\alpha = 0.9375$

$$\begin{cases} L_{i,j}^{(t)} = \alpha g\left(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ L_{i,j+N/2^i}^{(t)} = \alpha g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + L_{i+1,j+N/2^i}^{(t-1)} \\ R_{i+1,j}^{(t)} = \alpha g\left(R_{i,j}^{(t)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ R_{i+1,j+N/2^i}^{(t)} = \alpha g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + R_{i,j+N/2^i}^{(t)} \end{cases}$$

$$g(x, y) \approx \text{sign}(x)\text{sign}(y)\min(|x|, |y|)$$

Multiple Scaled Belief Propagation [10]

- Neural network-based BP: take advantage of the structure of belief propagation decoding
- Outperform conventional algorithm within fewer iterations



Unified scaled min-sum with $\alpha = 0.9375$

$$\begin{cases} L_{i,j}^{(t)} = \alpha g\left(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ L_{i,j+N/2^i}^{(t)} = \alpha g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + L_{i+1,j+N/2^i}^{(t-1)} \\ R_{i+1,j}^{(t)} = \alpha g\left(R_{i,j}^{(t)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ R_{i+1,j+N/2^i}^{(t)} = \alpha g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + R_{i,j+N/2^i}^{(t)} \end{cases}$$

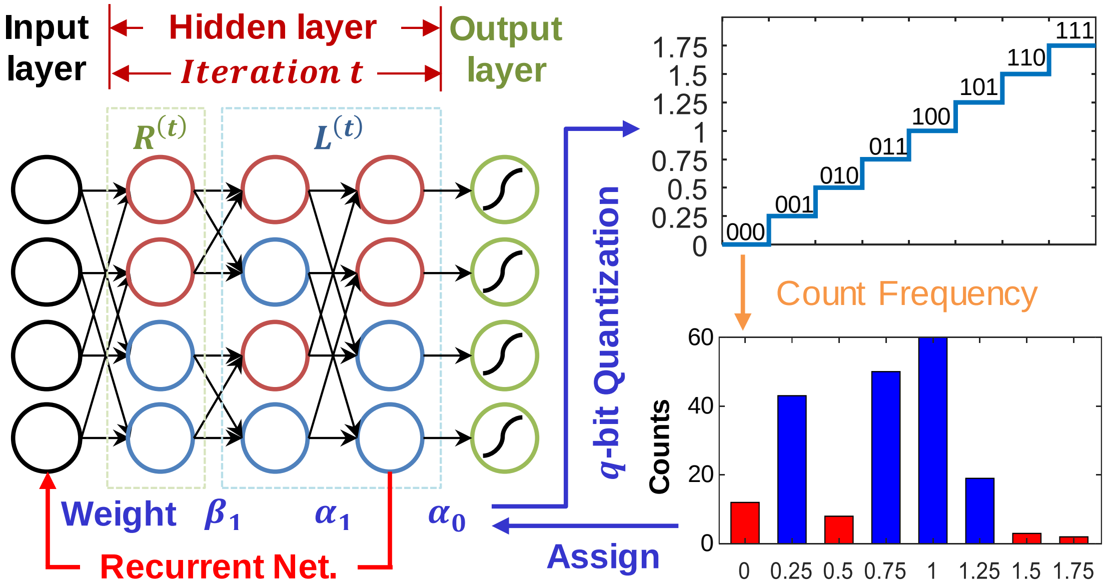
Multiple scaled min-sum

$$\begin{cases} L_{i,j}^{(t)} = \alpha_{i,j}^{(t)} g\left(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ L_{i,j+N/2^i}^{(t)} = \alpha_{i,j+N/2^i}^{(t)} g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + L_{i+1,j+N/2^i}^{(t-1)} \\ R_{i+1,j}^{(t)} = \beta_{i+1,j}^{(t)} g\left(R_{i,j}^{(t)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ R_{i+1,j+N/2^i}^{(t)} = \beta_{i+1,j+N/2^i}^{(t)} g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + R_{i,j+N/2^i}^{(t)} \end{cases}$$

General case when α & $\beta = 0.9375$ ➔ no worse performance

Proposed Recurrent Architecture with Codebook-based Weight Quantization

- Multiple scaled min-sum induces additional memory overhead for weight storage ➔ hinder the deployment of neural network decoder
- Massive multiplication on edges results in additional complexity



- Recurrent architecture ➔ dramatically reduces memory overhead
- Codebook-based weight quantization ➔ alleviates complexity

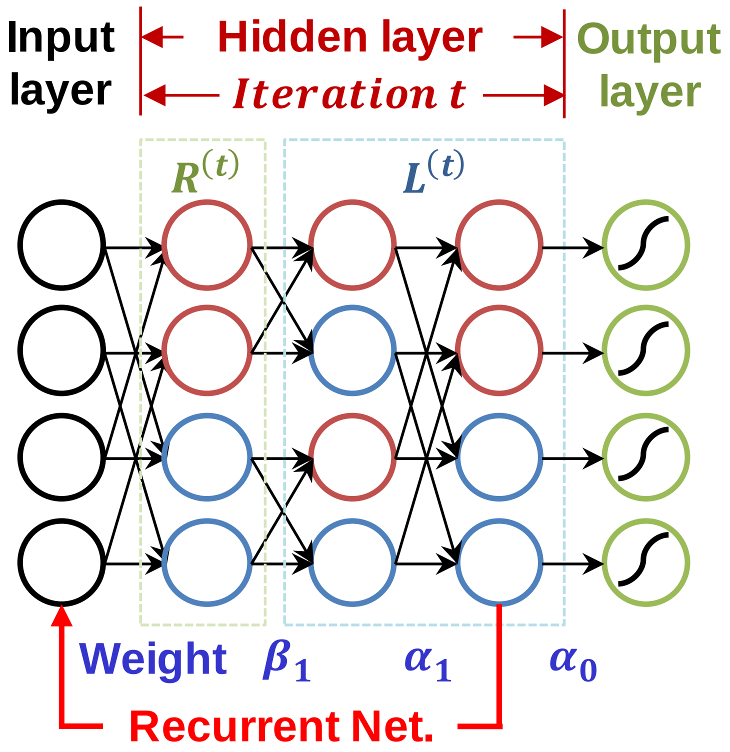
Recurrent Architecture [11]

- Force the network to reuse shared weights among different iteration
- Recurrent architecture leads to a different optimization problem
  - Dramatically reduce memory overhead with a little performance degradation

$$\begin{cases} L_{i,j}^{(t)} = \alpha_{i,j}^{(t)} g\left(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ L_{i,j+N/2^i}^{(t)} = \alpha_{i,j+N/2^i}^{(t)} g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + L_{i+1,j+N/2^i}^{(t-1)} \\ R_{i+1,j}^{(t)} = \beta_{i+1,j}^{(t)} g\left(R_{i,j}^{(t)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ R_{i+1,j+N/2^i}^{(t)} = \beta_{i+1,j+N/2^i}^{(t)} g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + R_{i,j+N/2^i}^{(t)} \end{cases}$$

$$\Downarrow$$

$$\begin{cases} L_{i,j}^{(t)} = \alpha_{i,j} g\left(L_{i+1,j}^{(t-1)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ L_{i,j+N/2^i}^{(t)} = \alpha_{i,j+N/2^i} g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + L_{i+1,j+N/2^i}^{(t-1)} \\ R_{i+1,j}^{(t)} = \beta_{i+1,j} g\left(R_{i,j}^{(t)}, L_{i+1,j+N/2^i}^{(t-1)} + R_{i,j+N/2^i}^{(t)}\right) \\ R_{i+1,j+N/2^i}^{(t)} = \beta_{i+1,j+N/2^i} g\left(R_{i,j}^{(t)}, L_{i+1,j}^{(t-1)}\right) + R_{i,j+N/2^i}^{(t)} \end{cases}$$



Codebook-based Weight Quantization [12-13]

- Weights are quantized after each epoch during the training process
- Double quantization: reduce both the required number of weights and the precision for each weights
- Scaling parameters are close to 1 ➔ q-bit quantization with step = $2^{-(q-1)}$
- Design c-bit codebook by counting the frequency ➔reduce q-bit to c-bit

0.78	0.56	0.95	1.02
0.45	0.35	0.94	1.66
1.22	0.81	1.47	1.16
0.12	0.99	1.28	0.72

Original Weights (32bit)

(a) q-bit Quantization $(q = 3)$

0.75	0.50	1.00	1.00
0.50	0.25	1.00	1.75
1.25	0.75	1.50	1.25
0.00	1.00	1.25	0.75

(b) Count Frequency

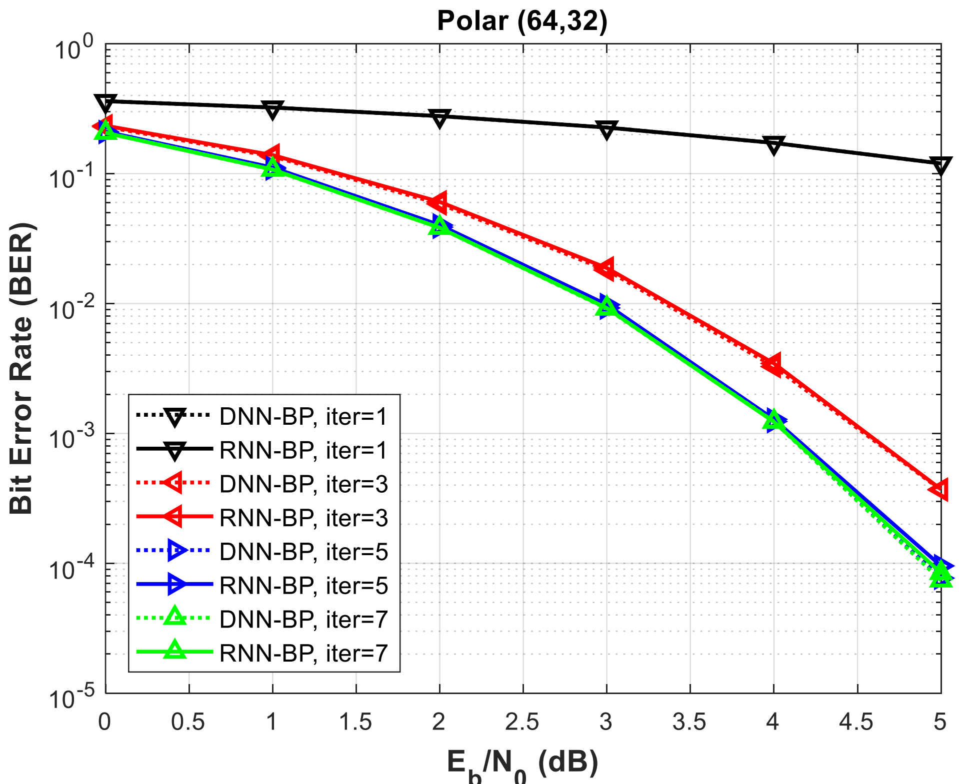
Value$_{(2)}$	Value$_{(10)}$	Count Freq.	Codebook Index
000	0.00	1	
001	0.25	1	
010	0.50	**2**	00
011	0.75	**3**	01
100	1.00	**4**	10
101	1.25	**3**	11
110	1.50	1	
111	1.75	1	

(c) c-bit Codebook-based Quantization $(c = 2)$

01	00	10	10
00	00	10	11
11	01	11	11
00	10	11	01

Simulation Results: Performance of DNN-BP and RNN-BP

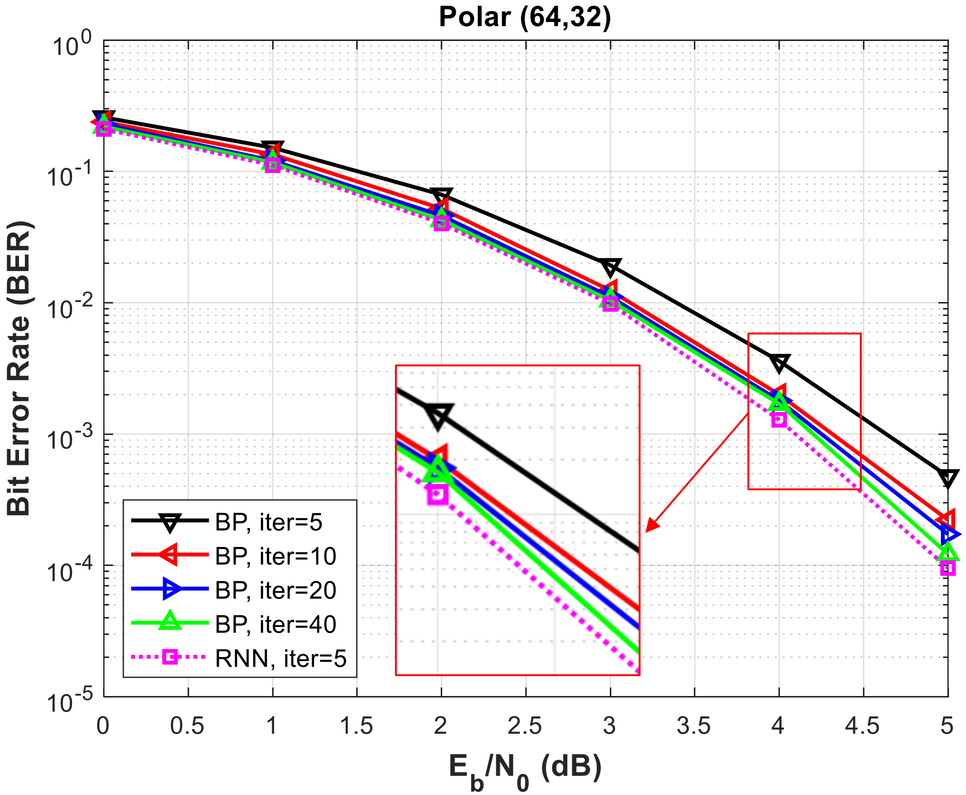
Parameter	Setups
Encoding	Polar (64,32)
SNR	0 ~ 5
Training codewords/SNR	40000
Testing codewords/SNR	100800
Mini-batch size	2400



- Five iteration is enough for convergence
- RNN-BP has almost the same performance as DNN-BP and reduces memory overhead by 80%

Simulation Results: Performance of BP and RNN-BP

Parameter	Setups
Encoding	Polar (64,32)
SNR	0 ~ 5
Training codewords/SNR	40000
Testing codewords/SNR	100800
Mini-batch size	2400

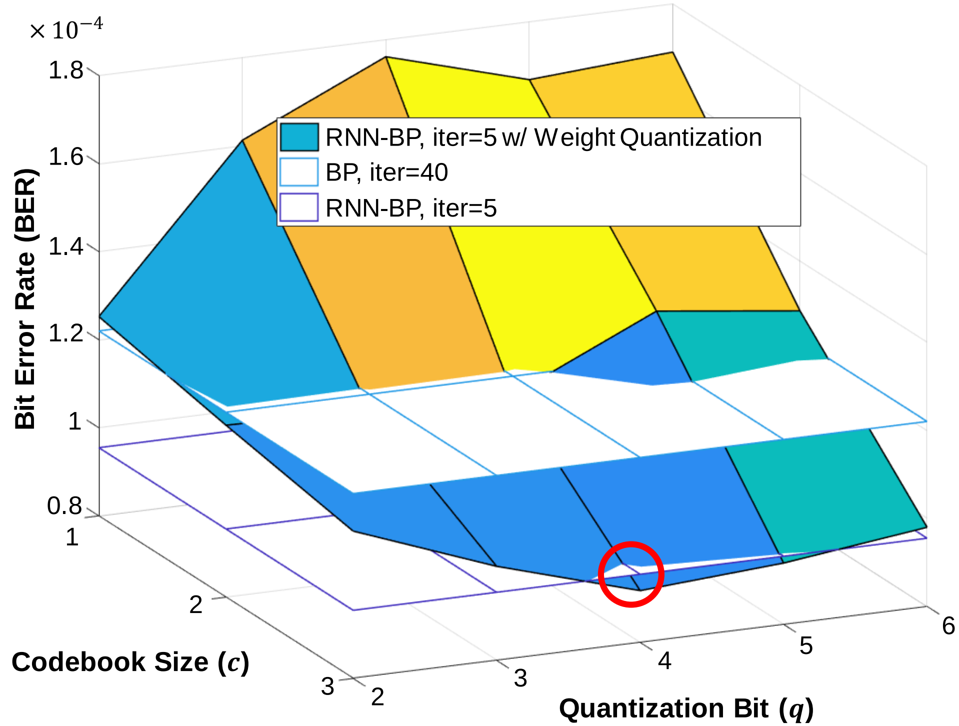


- RNN-BP with 5 iteration outperforms conventional BP with 40 iteration

➔ Reduce latency and complexity with higher throughput

Simulation Results: Performance of Weight Quantization

Parameter	Setups
Encoding	Polar (64,32)
SNR	5
Quantization bit (q)	2, 3, 4, 5, 6
Codebook size (c)	1, 2, 3



- When $c \leq 2$, longer bit length may result in local minimum
- When $c > 2$, longer bit length has lower BER
- Codebook size higher than 1 can outperform conventional BP

Complexity Analysis

	Addition	Multiplication	Memory (bit)
Conventional BP [7]	$2TN\log N$ $\sim 30{,}720$	0	0
DNN-BP [10]	$2TN\log N$ $\sim 3{,}840$	$2TN\log N$ $\sim 3{,}840$	$64TN\log N$ $\sim 122{,}880$
Proposed RNN-BP with codebook-based weight quantization	$2qTN\log N$ $\sim 15{,}360$	0	$2cN\log N$ $\sim 2{,}304$

*Iterations T for BP, DNN-BP, and RNN-BP are set to 40, 5, and 5, respectively. $N = 64$, $q = 4$, $c = 3$

- DNN-BP dramatically reduces the addition operations at the expense of significant memory overhead
- Proposed approach reduces memory overhead by 98% and replaces multiplication with shift and addition without visible performance loss

Conclusion

- Proposed recurrent architecture can learn the shareable parameters with effective reduction of memory overhead by 80%
- Proposed codebook-based weight quantization can further reduce memory overhead by 90% and alleviate hardware complexity
- Our proposed design is low complexity, low latency and high throughput; while being feasible for realizing neural network decoders in communication systems

Reference (1/2)

[1] E. Arikan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” *IEEE Trans. Inf. Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[2] K. Niu, K. Chen, J. Lin, Q. T. Zhang, “Polar Codes: Primary Concepts and Practical Decoding Algorithms,” *IEEE Communication Magazine*, vol. 52, no. 7, pp. 192-203, Jul. 2014.

[3] “Final report of 3GPP TSG RAN WG1 #87 v1.0.0,” Reno, USA, Nov. 2016.

[4] A. Alamdar-Yazdi and F. R. Kschischang, “A simplified successive cancellation decoder for polar codes,” *IEEE Commun. Lett*., vol. 15, no. 12, pp. 1378–1380, Dec. 2011.

[5] C. Leroux, I. Tal, A. Vardy, and W. J. Gross, “Hardware architectures for successive cancellation decoding of polar codes,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, pp. 1665–1668, May 2011.

[6] E. Arikan, “Polar codes: A pipelined implementation,” in *Proc. 4th Int. Symp. on Broad. Commun. ISBC 2010*, pp. 11-14, July 2010.

[7] A. Pamuk, “An FPGA implementation architecture for decoding of polar codes,” in *International Symposium on Wireless Communication Systems (ISWCS)*, Nov 2011.

[8] B. Yuan and K. K. Parhi, “Architecture optimizations for BP polar decoders,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, May 2013.

[9] B. Yuan and K.K. Parhi, “Early stopping criteria for energy-efficient low-latency belief-propagation polar code decoders,” *IEEE Trans. Signal Process.*, vol.62, no.24, pp.6496–6506, Dec.15, 2014.

Reference (2/2)

[10] W. Xu, Z. Wu, Y.-L. Ueng, X. You, C. Zhang, “Improved polar decoder based on deep learning,” *Proc. IEEE International Workshop on Signal Processing Systems (SiPS)*, pp. 1-6, Oct. 2017.

[11] E. Nachmani, E. Marciano, L. Lugosch, W. Gross, D. Burshtein and Y. Be'ery, “Deep Learning Methods for Improved Decoding of Linear Codes,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 119-131, 2018.

[12] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, Y. Bengio, “Quantized neural networks: Training neural networks with low precision weights and activations,” *CoRR*, 2016.

[13] S. Han, H. Mao, and W. J. Dally, “Deep compression: Compressing deep neural network with pruning, trained quantization and Huffman coding,” *International Conference on Learning Representations*, 2016.

The end

Thank you for your listening