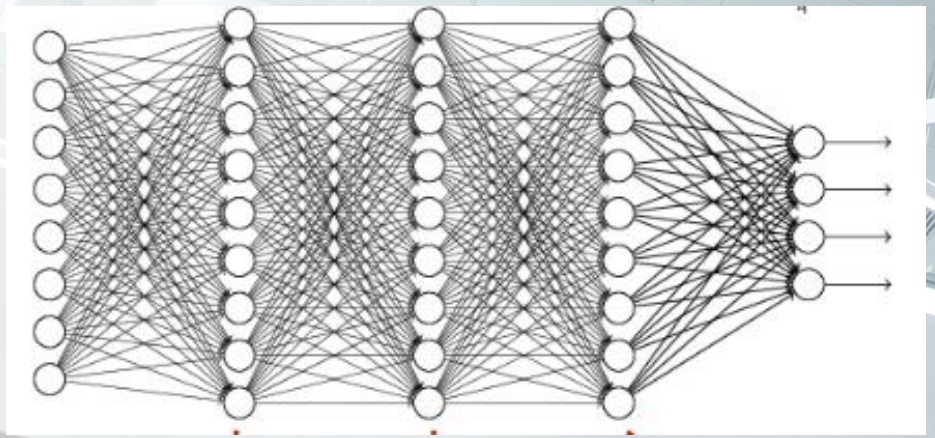
Western Digital.

18 NVMW

Training Iterative Decoders using Deep Learning Approaches

Eran Sharon, Ariel Navon

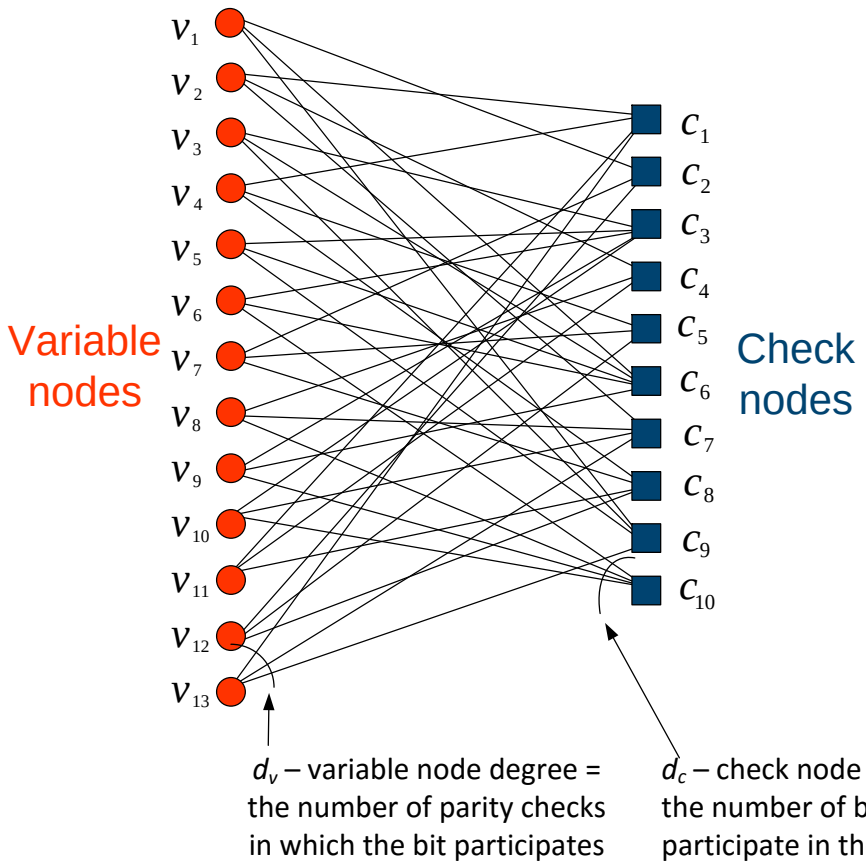
March 2018



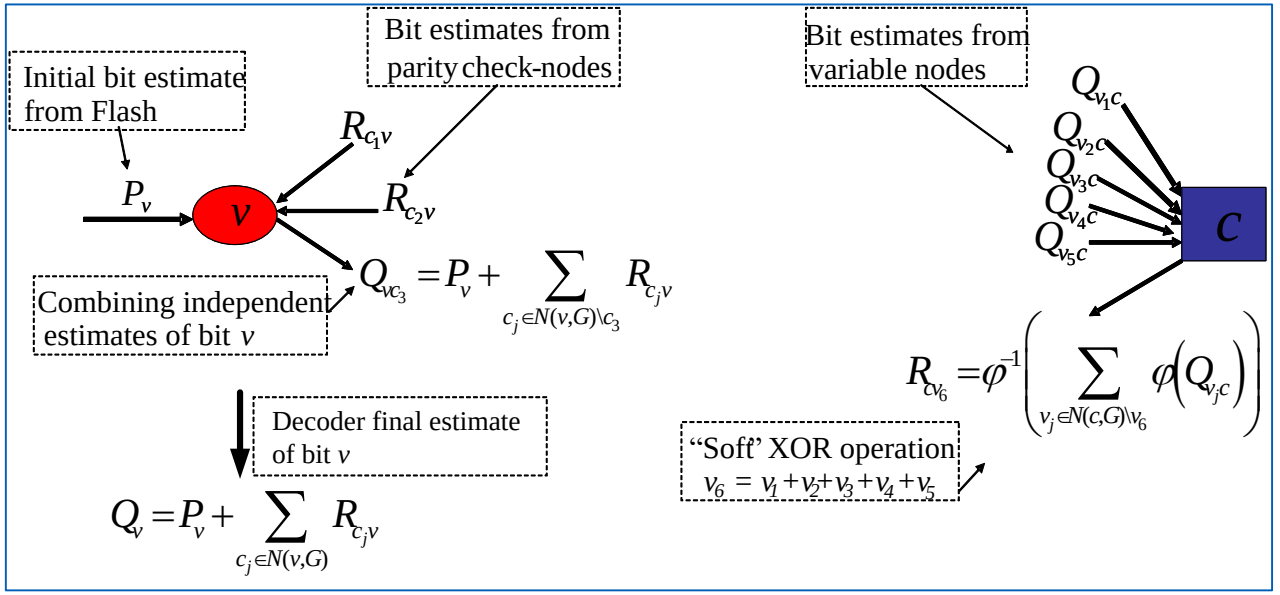
©2017 Western Digital Corporation or its affiliates. All rights reserved. Confidential.

LDPC codes and Iterative Decoders

Sparse bipartite graph

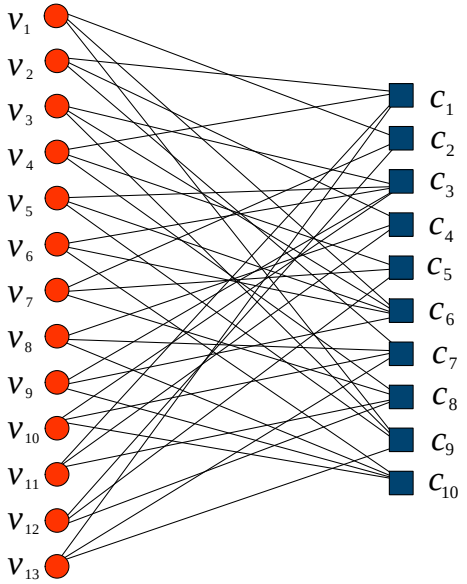


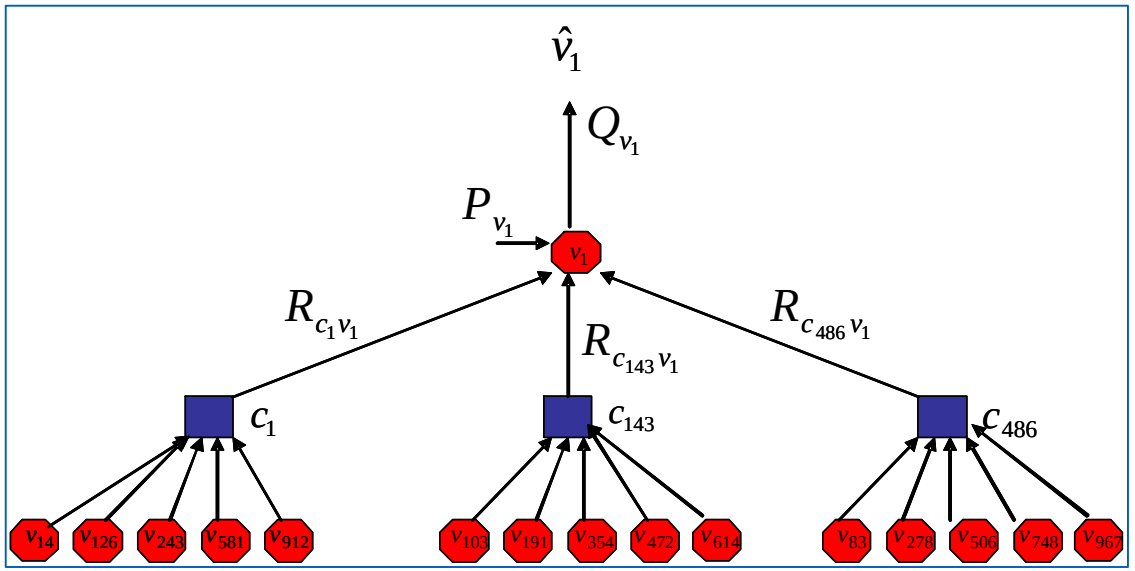
Belief Propagation (BP) Decoding:



Background – Sub optimality of iterative decoders

- **Long LDPC codes** can approach the **Shannon limit**
- **Long code** → **Latency** and **Complexity** penalties
- **Short codes** are required for certain applications, such as **SCM**
- Conventional **iterative coding** schemes **perform poorly for short codes**:
  - **BP** computation rules assume statistical independence of messages
    → **Assume cycle free computation tree** (holds asymptotically…)
  - Cannot avoid **many short cycles** in the graph of **short codes**
  - Impair the statistical independence assumption → **sub-optimal decoding**

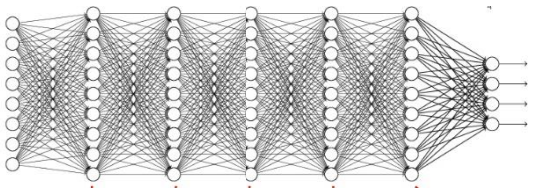




Training Iterative Decoders using Deep Learning

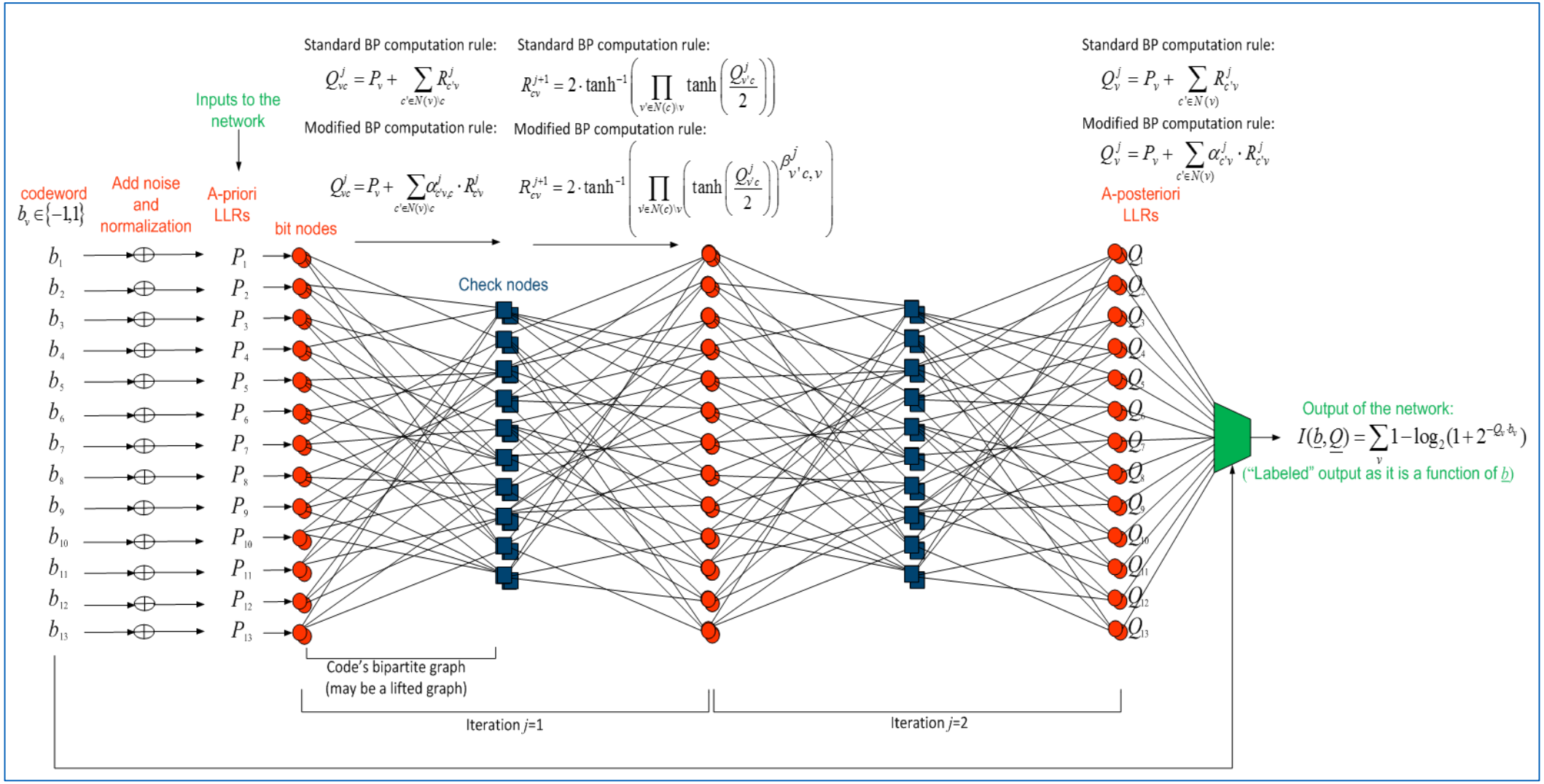
➔ Improved iterative decoder for short codes is proposed

- **Modifying the standard message-passing computation rules**, according to the specific statistical dependencies (induced by cycles in the code's graph) of certain code
- **Use Deep Learning methods for training an iterative decoder on a specific code -** similar ideas suggested in [1,2]
- **Iterative decoding fits very well to the Deep Learning framework**:
  - **Net Structure**: Iterative decoder computations along several iterations can be described in terms of a Deep-Neural-Net (DNN)
  - **"Big Data"**: Unlimited labeled data generated via an endless Monte-Carlo simulation
    - Noisy codewords serve as network inputs
    - Clean codewords serve as labeled network outputs



[1] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in 54'th Annual Allerton Conf. On Communication, Control and Computing, September 2016.
[2] T. Gruber, S. Cammerer, J. Hoydis, and S.t.Brink, "On deep learning based channel decoding," Conf. on Information Sciences and Systems, 2017.

Deep Learning Network for Improved Belief Propagation



- **Representation BP process as Neural Net**
- **Optimizing net weights is equivalent to learning the coefficients of improved BP rules**

Deep Learning Network for Improved Belief Propagation

Standard BP computation rules:

$$Q_{vc}^{j} = P_v + \sum_{c' \in N(v) \setminus c} R_{c'v}^{j}$$

$$R_{cv}^{j+1} = 2 \cdot \tanh^{-1}\left(\prod_{v' \in N(c) \setminus v} \tanh\left(\frac{Q_{v'c}^{j}}{2} \right) \right)$$

Modified BP computation rules:

$$Q_{vc}^{j} = P_v + \sum_{c' \in N(v) \setminus c} \alpha_{c'v,c}^{j} \cdot R_{c'v}^{j}$$

$$R_{cv}^{j+1} = 2 \cdot \tanh^{-1}\left(\prod_{v' \in N(c) \setminus v} \left(\tanh\left(\frac{Q_{v'c}^{j}}{2} \right) \right)^{\beta_{v'c,v}^{j}} \right)$$

- **Representation BP process as Neural Net**
- **Optimizing net weights is equivalent to learning the coefficients of improved BP rules**

Training Procedure for Improved Belief Propagation

Generate a batch of noisy codewords – for each one perform:

Forward pass –

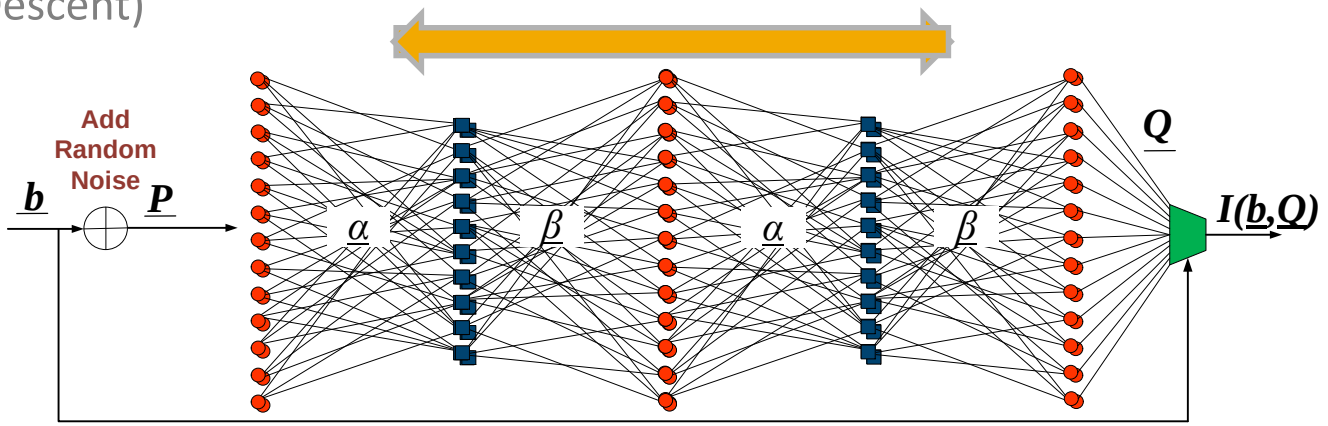
- **Message passing decoding**: converts a-priori LLR inputs ($\underline{P}$) to a-posteriori LLR outputs ($\underline{Q}$)
- Clean codeword serves as "ground-truth" ($\underline{b}$)
- Loss-Function: Mutual Information (MI) between $\underline{Q}$ and $\underline{b}$: $I\left(\underline{b},\underline{Q}\right) = \sum_v 1 - \log(1 + 2^{-Q_v \cdot b_v})$

Backward pass -

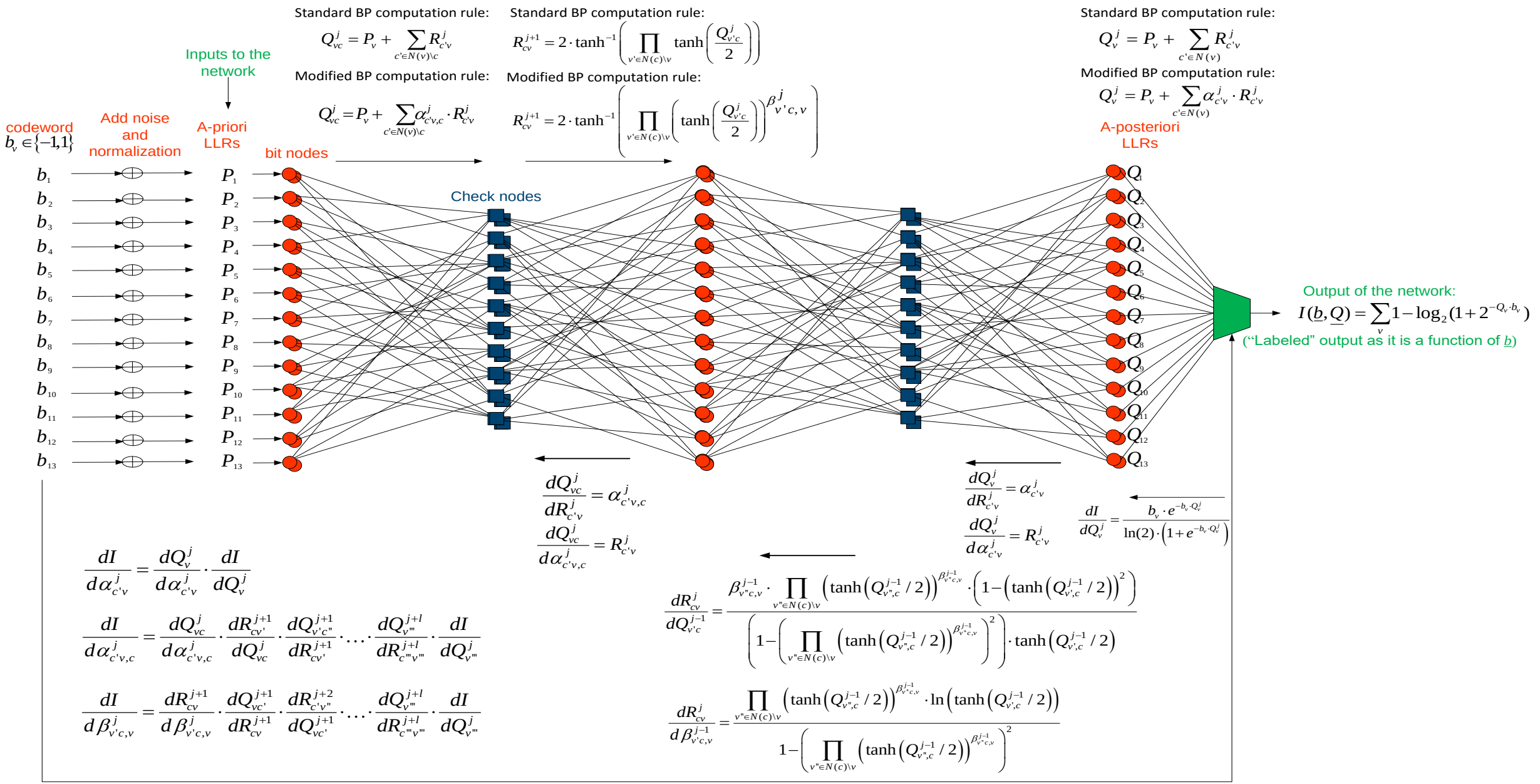
- **Backward message passing** of partial derivatives
- Simultaneously computes gradients for all the network parameters via back propagation
- Gradients indicate how changing each of the parameters affects the MI loss function

Update Network Parameters:

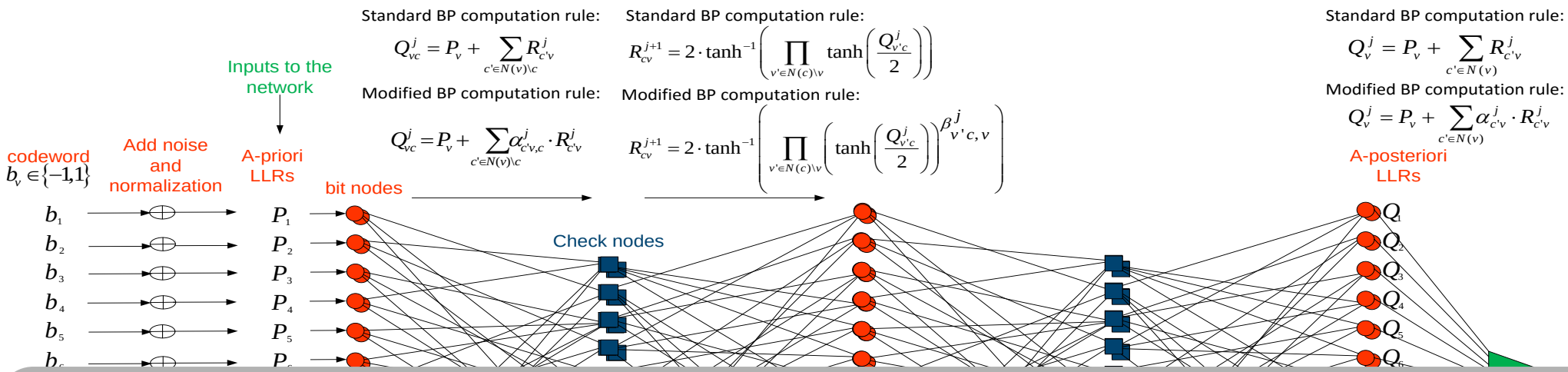
- Adjust α, β parameters of the computation rules according to averaged gradients over the batch to maximize $I(\underline{b},\underline{Q})$ (Stochastic Gradient Descent)



Training Procedure for Improved Belief Propagation



Training Procedure for Improved Belief Propagation



Non standard DL problem – Hard to use existing DL packages…

➔ Developed **tailored DL package** for iterative decoders from scratch (Math + Coding)

$$\frac{dQ_{vc}^{j}}{dR_{c'v}^{j}} = \alpha_{c'v,c}^{j} \qquad \frac{dQ_{vc}^{j}}{d\alpha_{c'v,c}^{j}} = R_{c'v}^{j}$$

$$\frac{dQ_{v}^{j}}{dR_{c'v}^{j}} = \alpha_{c'v}^{j} \qquad \frac{dQ_{v}^{j}}{d\alpha_{c'v}^{j}} = R_{c'v}^{j}$$

$$\frac{dI}{dQ_{v}^{j}} = \frac{b_v \cdot e^{-b_v \cdot Q_v^{j}}}{\ln(2) \cdot \left(1 + e^{-b_v \cdot Q_v^{j}}\right)}$$

$$\frac{dI}{d\alpha_{c'v}^{j}} = \frac{dQ_{v}^{j}}{d\alpha_{c'v}^{j}} \cdot \frac{dI}{dQ_{v}^{j}}$$

$$\frac{dI}{d\alpha_{c'v,c}^{j}} = \frac{dQ_{vc}^{j}}{d\alpha_{c'v,c}^{j}} \cdot \frac{dR_{cv'}^{j+1}}{dQ_{vc}^{j}} \cdot \frac{dQ_{v'c''}^{j+1}}{dR_{cv'}^{j+1}} \cdot \ldots \cdot \frac{dQ_{v'''}^{j+l}}{dR_{c'''v'''}^{j+l}} \cdot \frac{dI}{dQ_{v'''}^{j}}$$

$$\frac{dI}{d\beta_{v'c,v}^{j}} = \frac{dR_{cv}^{j+1}}{d\beta_{v'c,v}^{j}} \cdot \frac{dQ_{vc'}^{j+1}}{dR_{cv}^{j+1}} \cdot \frac{dR_{c'v''}^{j+2}}{dQ_{vc'}^{j+1}} \cdot \ldots \cdot \frac{dQ_{v'''}^{j+l}}{dR_{c'''v'''}^{j+l}} \cdot \frac{dI}{dQ_{v'''}^{j}}$$

$$\frac{dR_{cv}^{j}}{dQ_{v'c}^{j-1}} = \frac{\beta_{v''c,v}^{j-1} \cdot \prod_{v'' \in N(c) \setminus v} \left(\tanh\left(Q_{v'',c}^{j-1} / 2 \right) \right)^{\beta_{v''c,v}^{j-1}} \cdot \left(1 - \left(\tanh\left(Q_{v',c}^{j-1} / 2 \right) \right)^{2} \right)}{\left(1 - \left(\prod_{v'' \in N(c) \setminus v} \left(\tanh\left(Q_{v'',c}^{j-1} / 2 \right) \right)^{\beta_{v''c,v}^{j-1}} \right)^{2} \right) \cdot \tanh\left(Q_{v',c}^{j-1} / 2 \right)}$$

$$\frac{dR_{cv}^{j}}{d\beta_{v'c,v}^{j-1}} = \frac{\prod_{v'' \in N(c) \setminus v} \left(\tanh\left(Q_{v'',c}^{j-1} / 2 \right) \right)^{\beta_{v''c,v}^{j-1}} \cdot \ln\left(\tanh\left(Q_{v',c}^{j-1} / 2 \right) \right)}{1 - \left(\prod_{v'' \in N(c) \setminus v} \left(\tanh\left(Q_{v'',c}^{j-1} / 2 \right) \right)^{\beta_{v''c,v}^{j-1}} \right)^{2}}$$

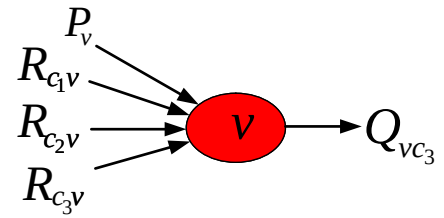
Parametric Computation Rule Models

- There may be various options for parametric computation rules.
- E.g. for the variable-to-check computation rule, we may use: $Q_{vc}^{j} = P_v + \sum_{c\prime \in N(v)\backslash c} \alpha_{c\prime v,c}^{j} \cdot R_{c\prime v}^{j}$

- Alternatively, we may use a more involved parametric computation rule:

$$Q_{vc} = P_v + \sum_{c\prime \in N(v)\backslash c} \left(\alpha_{c\prime v,c,1} + \alpha_{c\prime v,c,2} \cdot e^{-\alpha_{c\prime v,c,3} \cdot |\delta(\underline{R}_{cv}, c\prime)|}\right) \cdot R_{c\prime v}, \quad \text{where} \quad \underline{R}_{cv} = \{R_{c\prime v}, c' \in N(v)\backslash c\}$$

and $\delta(\underline{R}_{cv}, c') = R_{c\prime v} - \frac{1}{|N(v)\backslash c|} \sum_{c\prime\prime \in N(v)\backslash c} R_{c\prime\prime v}$

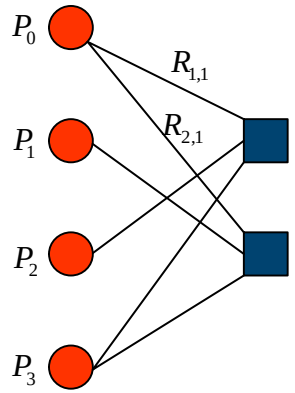


In this case, higher "damping" factor is applied to a message if it resembles its accompanying messages
Intuition: message resemblance indicate stronger statistical dependencies between the messages

- **What is a good parametric model?**
  - **A model that will allow the modified BP to approach MAP decoding**
- **How can we come up with such a model?**
  - **A parametric model may be determined by analyzing a "Toy" graph**
    - Simple enough for computing the expression for optimal MAP decoding
    - Model derived as the one which allows for the best fit between the Modified-BP and MAP

Toy Example – Case Study

- Choose a toy graph for which computing the MAP decoder is feasible
- Compare the MAP decoder to the Modified BP decoder in order to determine what parametric model can be used to approximate MAP



$$H=\begin{bmatrix}1&1&0&1\\1&0&1&1\end{bmatrix}$$

$$C^{\perp}=\begin{Bmatrix}0&0&0&0\\1&1&0&1\\1&0&1&1\\0&1&1&0\end{Bmatrix}$$

(codebook of the dual code- used for computing the MAP expression based on [1])

$$T_j=\tanh\left(\frac{P_j}{2}\right)$$

$$Q_{0,BP}=P_0+R_{1,1}+R_{2,1}=P_0+\log\frac{1+T_1\cdot T_3}{1-T_1\cdot T_3}+\log\frac{1+T_2\cdot T_3}{1-T_2\cdot T_3}=P_0+\log\frac{1+T_1\cdot T_3+T_2\cdot T_3+T_1\cdot T_2\cdot T_3^2}{1-T_1\cdot T_3-T_2\cdot T_3+T_1\cdot T_2\cdot T_3^2}$$

$$Q_{0,MAP}=P_0+\log\frac{1+T_1\cdot T_3+T_2\cdot T_3+T_1\cdot T_2}{1-T_1\cdot T_3-T_2\cdot T_3+T_1\cdot T_2}$$ (computed based on dual codebook [1])

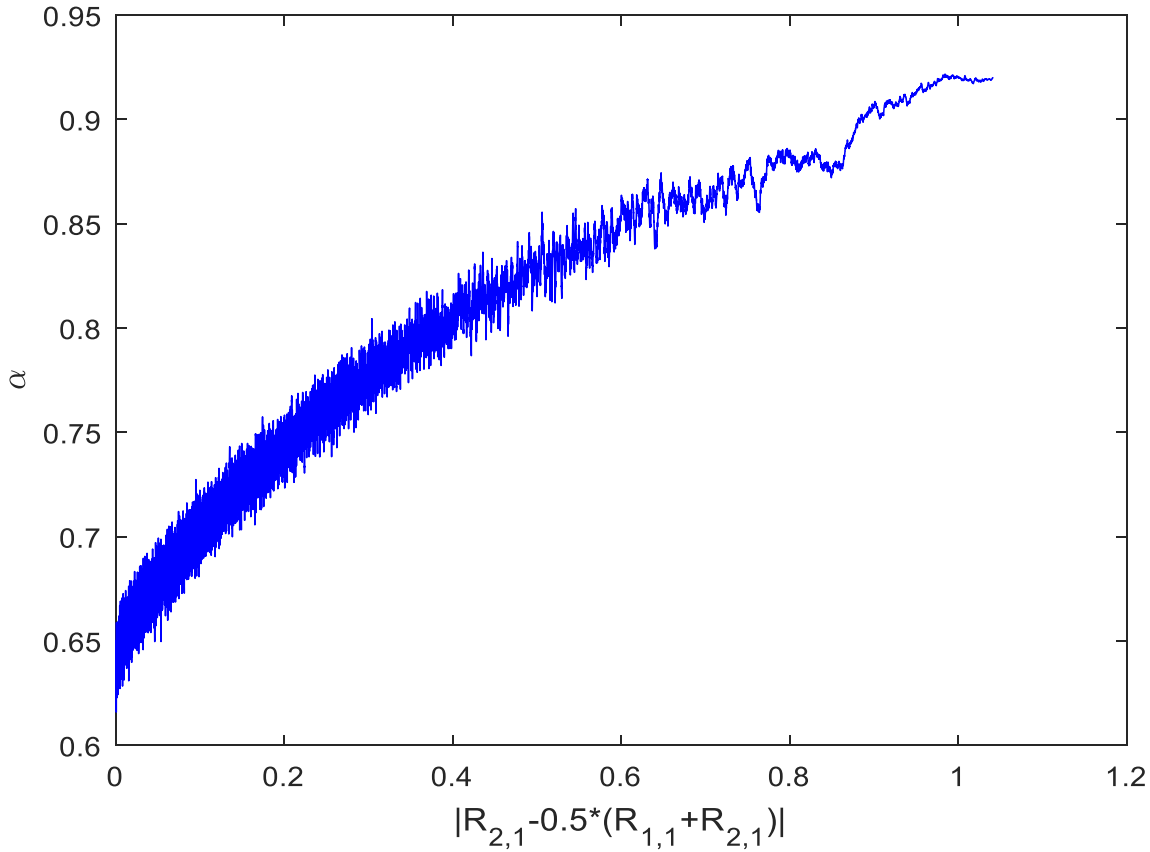
Assume: $Q_{0,MAP}=Q_{0,ModifiedBP}=P_0+R_{1,1}+\alpha\cdot R_{2,1}$

$$\Rightarrow\quad \alpha=\frac{\log\frac{1+T_1\cdot T_3+T_2\cdot T_3+T_1\cdot T_2}{1-T_1\cdot T_3-T_2\cdot T_3+T_1\cdot T_2}-\overbrace{\log\frac{1+T_1\cdot T_3}{1-T_1\cdot T_3}}^{R_{1,1}}}{\underbrace{\log\frac{1+T_2\cdot T_3}{1-T_2\cdot T_3}}_{R_{2,1}}}$$

[1] C.R.Hartmann and L.D.Rudolph, "An optimum symbol-by-symbol decoding rule for linear codes," *IEEE Trans. on Info. Theory*, vol. 22, pp.514-517, 1976.

Toy Example – Case Study

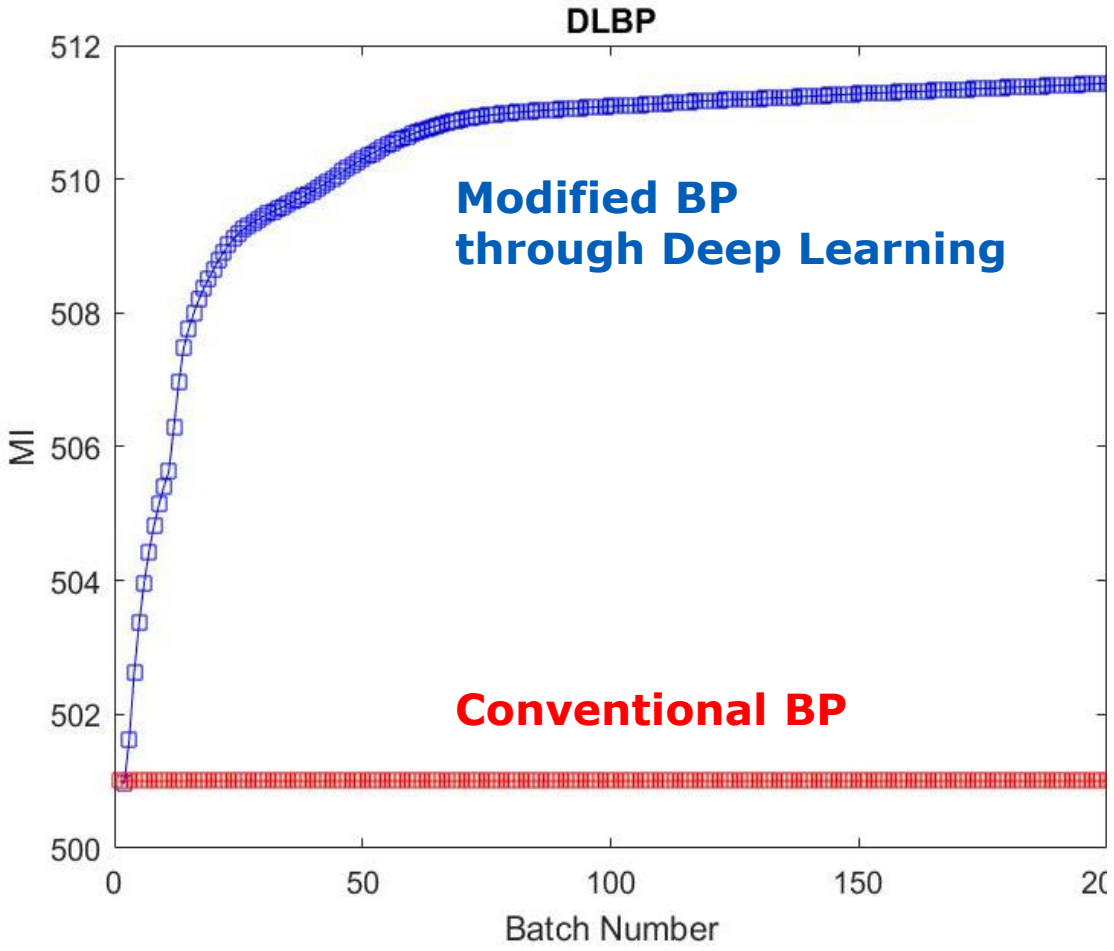
The following figure show α as a function of $|\delta| = |R_{2,1} - (R_{2,1} + R_{1,1})/2|$ for random selections of P_0, P_1, P_2 and P_3

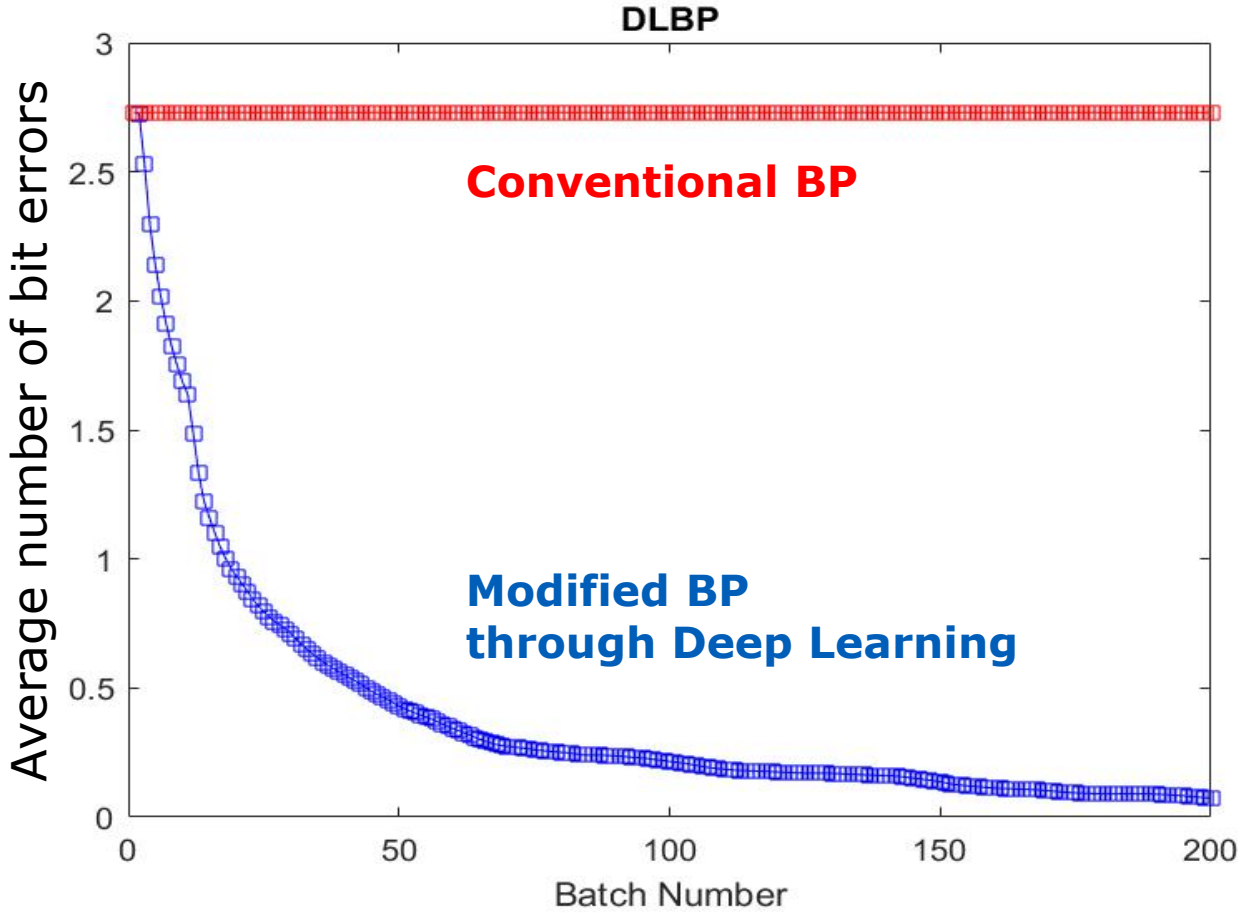


This indicates that a parametric model such as $\alpha(|\delta|) = \alpha_1 + \alpha_2 \cdot e^{-\alpha_3 \cdot |\delta|}$ may allow the modified BP to better approximate MAP decoding, by deriving the parameters of the model through the proposed deep learning approach.

Training results

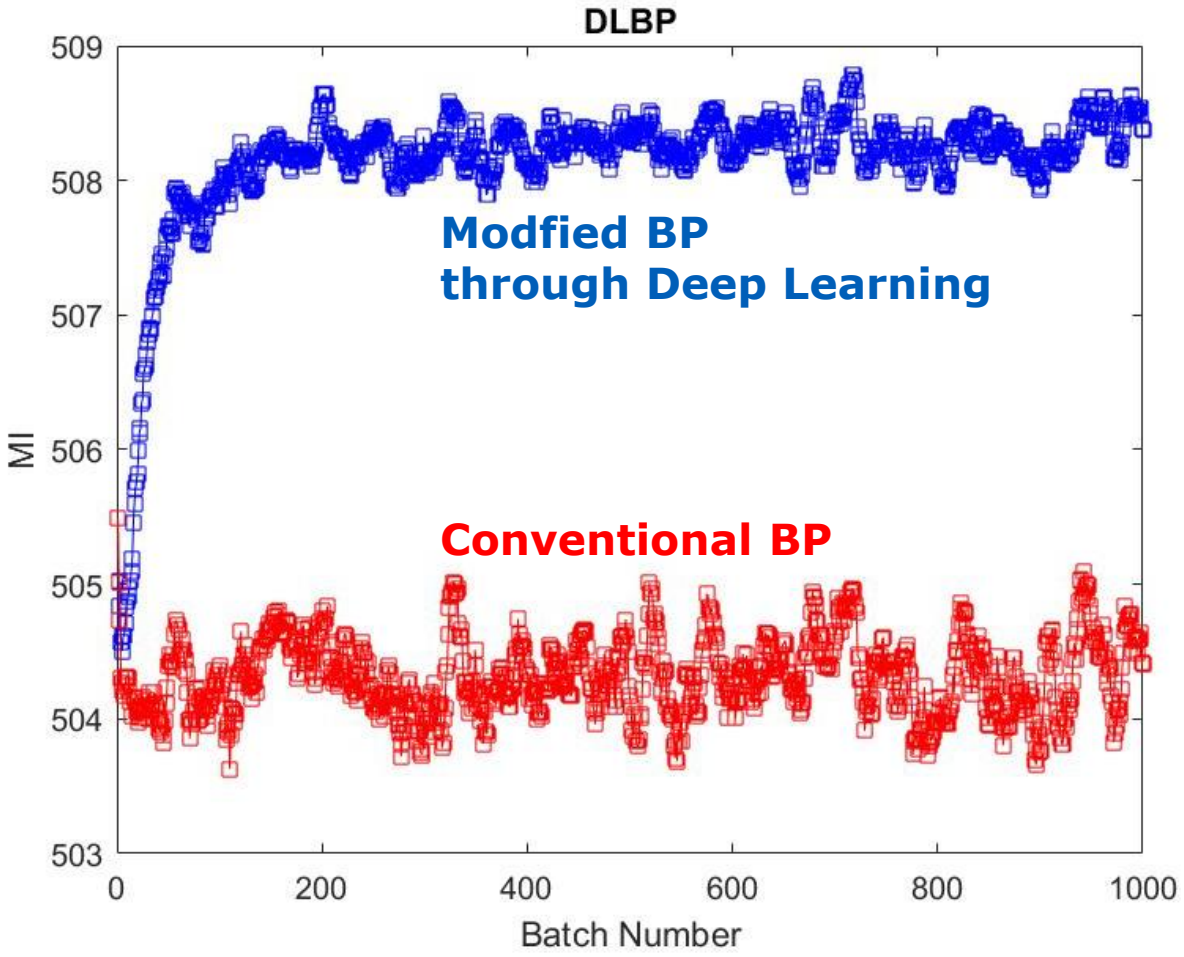
Training a length 512 code over the **same batch** over and over again – leads to "over fitting" - near perfect correction of the specific "noise" samples of the batch (but poor correction of other "noise" samples)

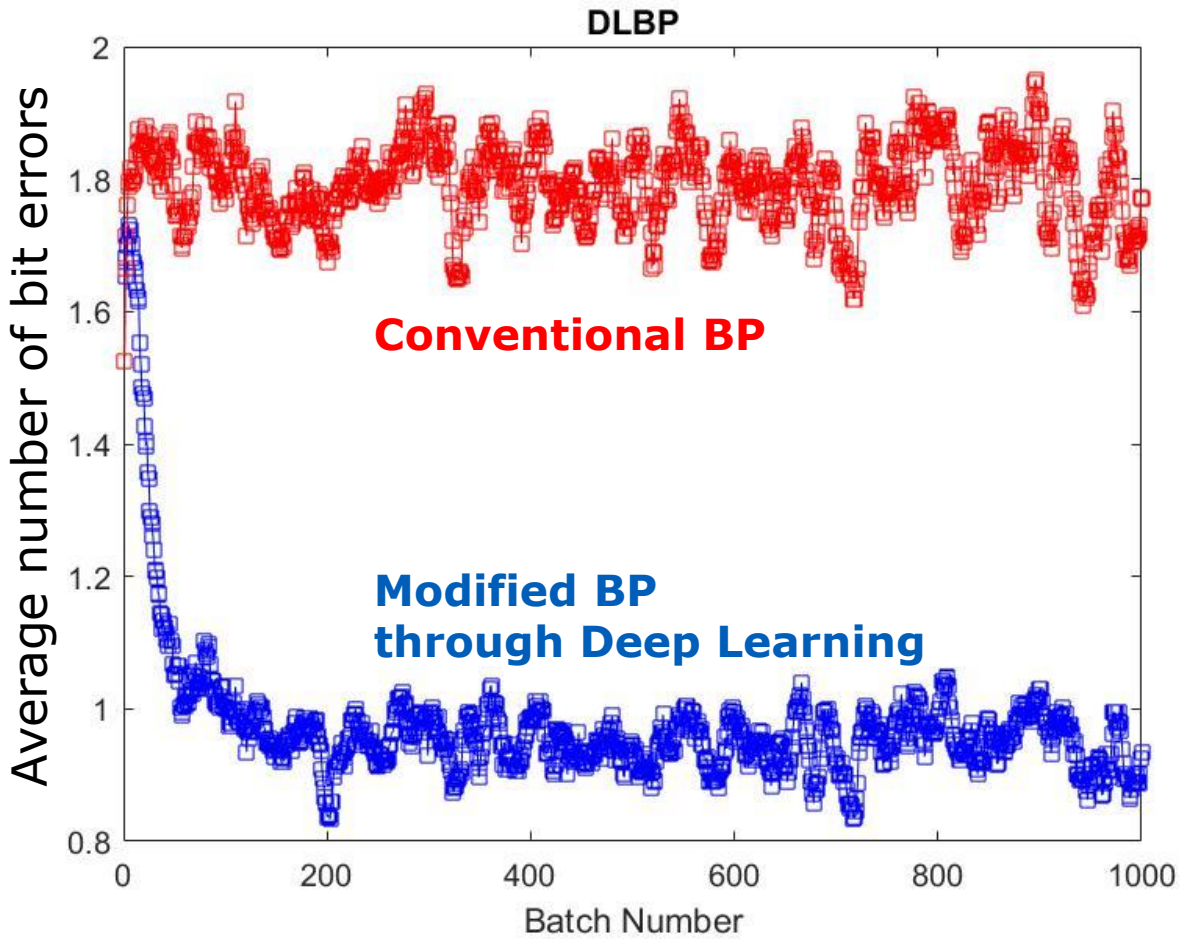




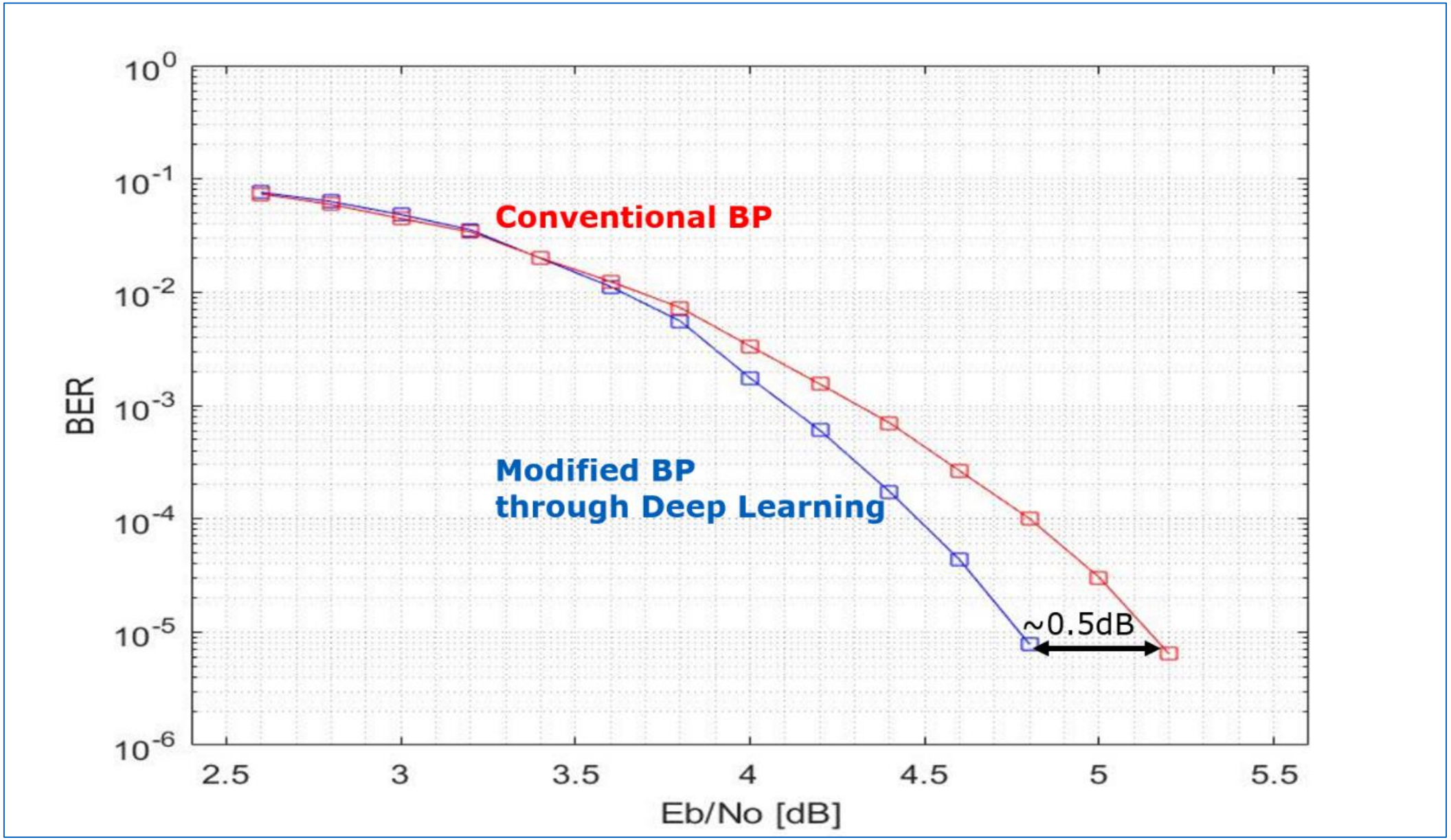
Training results

Training a length 512 code over **different** batches:



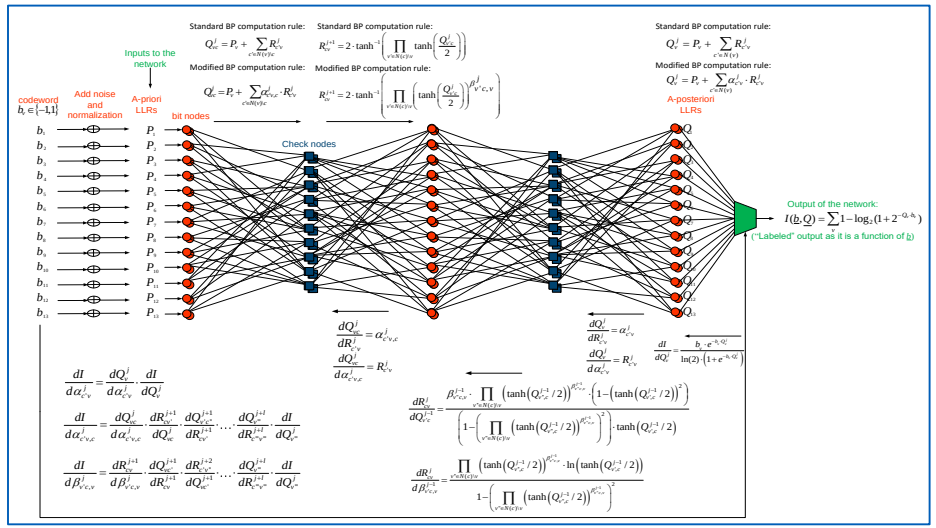


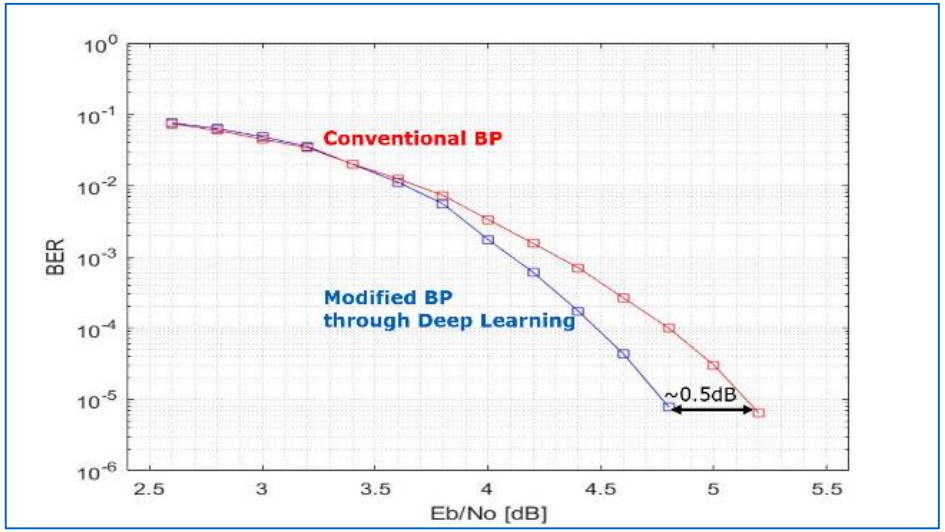
Simulation results



Summary - Training Iterative Decoders using DL

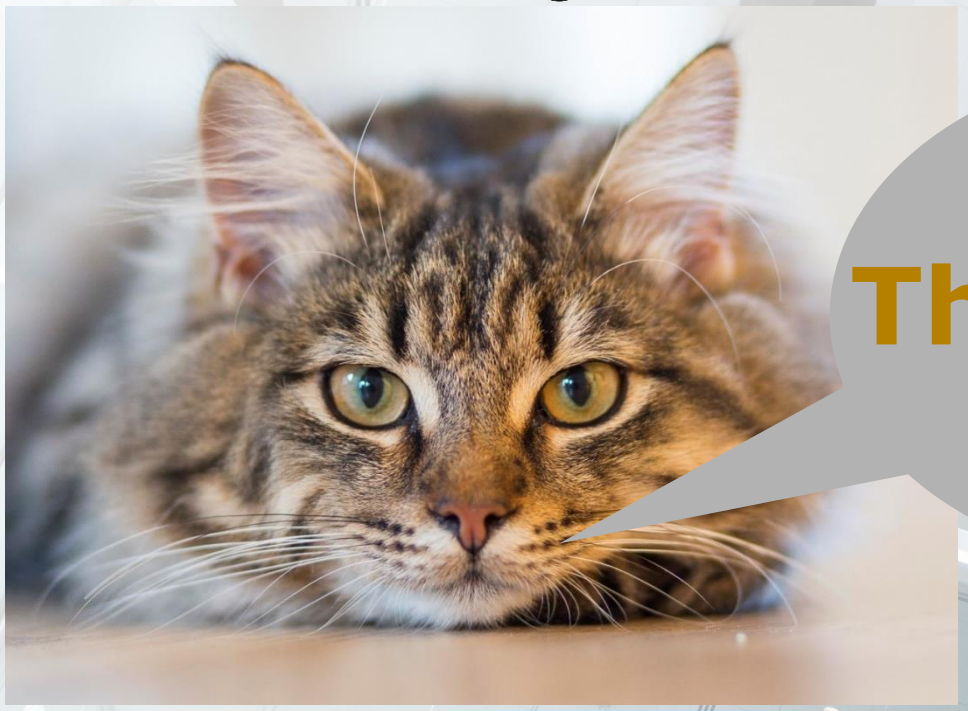
- Conventional iterative decoders perform poorly for short codes
- **Deep Learning** approach proposed for **training an iterative decoder**
  - learning complex statistical dependencies in the code's graph
    - Iterative decoder computations along several iterations is described as **DNN**
    - Unlimited labeled data generated via endless Monte-Carlo simulation – "**Big Data**"
- **Improved correction capability demonstrated for short codes**





- Future work includes decoder **On-Line training** – decoder improves performance over memory lifetime by adapting to the observed host traffic and/or NAND noises – complements the memory characteristic degradation during memory lifetime

Western Digital®



Thank you!

Questions?

Contact:

eran.sharon@wdc.com

©2017 Western Digital Corporation or its affiliates. All rights reserved. Confidential.

©2017 Western Digital Corporation or its affiliates. All rights reserved. Confidential.