1. RNN decoding of linearblock codes

https://arxiv.org/abs/1702.07560

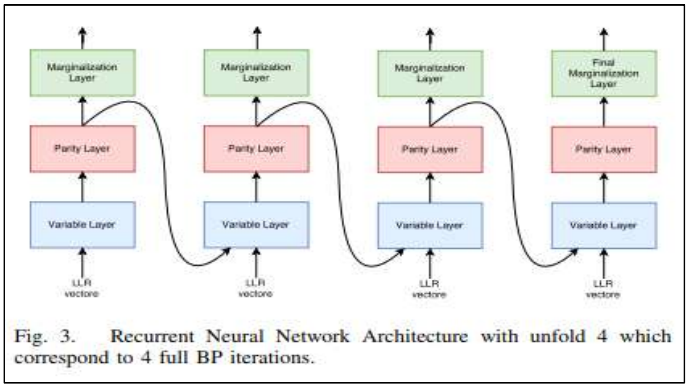

Fig. 3. Recurrent Neural Network Architecture with unfold 4 which correspond to 4 full BP iterations.

In this paper we introduce a recurrent neural network architecture for decoding linear block codes. Our method shows comparable bit error rate results compared to the feed-forward neural network with significantly less parameters.

(idea : RNN -> LSTM)

2. Deep learning methods for improved decoding of linear codes

https://arxiv.org/pdf/1706.07043.pdf

We also introduce a recurrent neural decoder architecture based on the method of successive relaxation. Another technique which can be used to improve the performance of belief propagation is the method of successive relaxation (or simply "relaxation"), as described in [23].

In relaxation, an exponentially weighted moving average is applied to combine the message sent in iteration $t-1$ with the raw message computed in iteration t to yield a filtered message, m'_t:

$$m'_t = \gamma m'_{t-1} + (1-\gamma)m_t \quad (19)$$

where γ is the relaxation factor. As $\gamma \to 0$, the decoder becomes less relaxed, and as $\gamma \to 1$, the decoder becomes more relaxed. When $\gamma = 0$, the decoder reverts to being a normal, non-relaxed decoder.

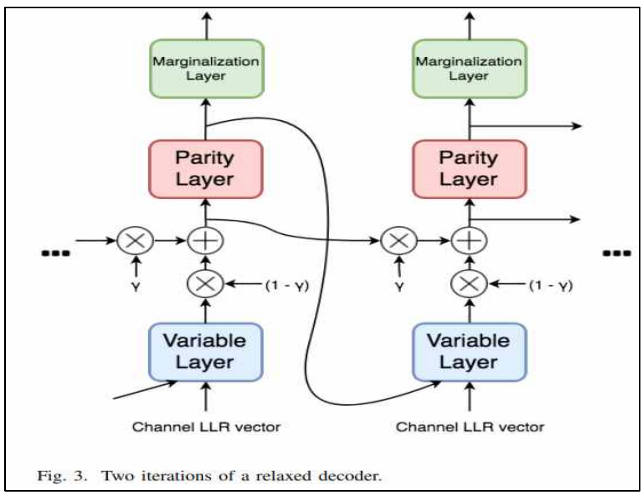

Fig. 3. Two iterations of a relaxed decoder.

(1) SGD로 γ값을 최적화한다고 하였지만, 그 값에 따른 성능 차이를 언급하고 있지 않다. 물론 brute force simulation을 통해서 보단 SGD로 최적화 후 BER 결과값을 출력했다고 하지만 그에 따른 성능 차이 비교.

2) 이것을 LSTM이나 CNN에 적용해보기)

3. Pruning Neural Belief Propagation Decoders

https://arxiv.org/pdf/2001.07464.pdf

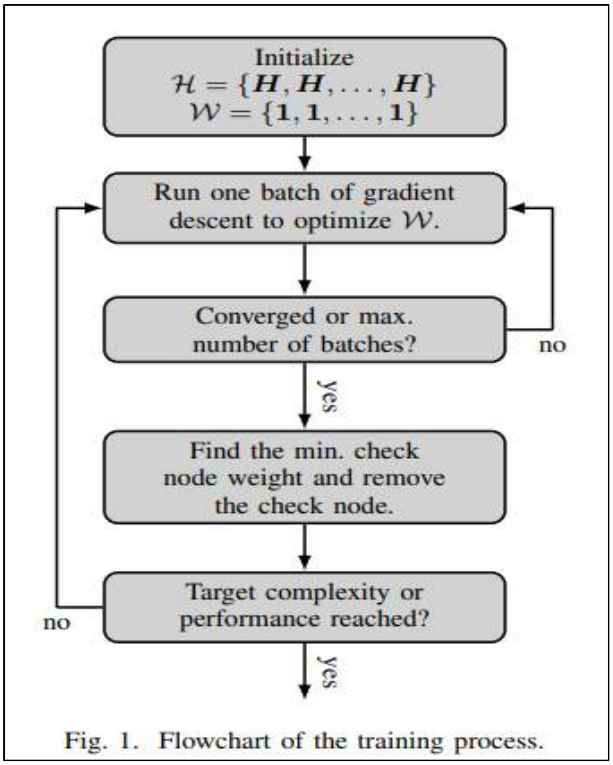

Fig. 1. Flowchart of the training process.

Considering the weights in the Tanner graph, the magnitude of the weights gives an indication of the importance of the edge in the decoding process. A magnitude close to zero indicate that the edge has low importance. By tying the weights for each CN, i.e., enforcing that the weights of all incoming edges to a single CN are equal, the weights can be interpreted as an indication of the importance of the CN in the decoding process.

CNs with connected low-weight edges do not play an important role in the decoding process and can be removed. We use this magnitude-based pruning approach to reduce the complexity of the decoder by removing CNs from the Tanner graph.

(1. 가장 작은 weight를 가진 CNS 값을 모두 0으로 만드는 것 혹은 dependent한 row값을 0으로 설정하여, 제거하는 방법 -> weight 모두 제거 시 성능 감소가 있으니, weight를 training으로 모두 제거말고 일부 제거하여, 성능향상과 복잡성 감소 이득

2. 보통 train 횟수와 test의 횟수를 동일하게 진행하는데, train 과정에서 얻은 최적의 결과값을 goal로 설정 후, 그에 90%, 95% 등 적정 수준 도달 시 test epoch stop.

3. Although recurrent neural network (RNN) [11] can process long code, some performance degradation appears since the neural network is unable to make the best use of the continuous inputs.
long code에서 좋은 성능을 보일 수 있는 RNN이 지속적으로 input을 neural network에 입력할 수 없어, 성능 하락이 있으니, 만일 A라는 짧은 코드가 있다면, 이것을 protograph ldpc를 만들 때, copy & permutation을 하는 것처럼 입력 값을 적정 횟수만큼 cop & permutation으로 continuous input이 되도록 설정.)

[11] G. Kechriotis, E. Zervas, and E. S. Manolakos, "Using recurrent neural networks for adaptive communication channel equalization," IEEE transactions on Neural Networks, vol. 5, no. 2, pp. 267-278, 1994.