УДК 004.89, 519.725

ДЕКОДИРОВАНИЕ ЛИНЕЙНЫХ БЛОКОВЫХ КОДОВ МЕТОДАМИ ГЛУБОКОГО ОБУЧЕНИЯ.

А. Э. Жданов, к.т.н., alexandr_zhdanov@mail.ru

Представлен пример мягкого декодера линейного блокового кода BCH(63,36,11), позволяющий оптимизировать свои характеристики методами глубокого машинного обучения. Декодер реализуют алгоритм offset min-sum в целых числах с фиксированной разрядностью, что позволяет реализовать его в аппаратуре. От известных решений декодер отличается тем, что применена "flooding" версия алгоритма, а также выполнен обоснованный выбор начальных весов, исходя из структуры проверочной матрицы, такой, что характеристики декодирования представленного декодера с начальными весами совпадают с характеристиками алгоритма из [10] после глубокого обучения. Планируется провести глубокое обучения представленного декодера, согласно методологии из [10]. Ожидается, что будут достигнуты лучшие характеристики, сокращено время обучения, расширено множество линейных блоковых кодов допускающих подобное декодирование.

Ключевые слова : *машинное обучение, глубокое обучение, линейный блоковый код, offset min-sum, flooding decoding.*

DECODING LINEAR BLOCK CODES WITH DEEP LEARNING.

A. E. Zhdanov, PhD, alexandr_zhdanov@mail.ru

The soft decoder of a linear block code BCH(63,36,11) is presented, which allows optimizing its characteristics using deep machine learning methods. The decoder implements the offset min-sum algorithm in integers with a fixed bitwidth, which allows it to be implemented in hardware. The decoder differs from the known solutions in that the "flooding" version of the algorithm is applied, and a reasonable choice of initial weights is made, based on the structure of the parity check matrix, such that the decoding performanece of the presented decoder with initial weights coincide with the performanece of the algorithm from [] after deep learning. It is planned to carry out deep learning of the presented decoder, according to the methodology from []. It is expected that better performance will be achieved, training time will be reduced, and the set of linear block codes allowing such decoding will be expanded.

Keywords: *machine learning, deep learning, offset min-sum, flooding decoding*

Канальное кодирование обеспечивает надежную связь по ненадежным, зашумленным каналам: кодируя сообщения с избыточностью, можно декодировать сообщения таким образом, что ошибки, вносимые каналом, исправляются. Современные канальные коды обеспечивают очень низкую частоту ошибок при больших длинах блоков, но длинные блоки часто неприемлемы для приложений с малой задержкой. Несмотря на то, что существуют короткие блочные коды с отличными характеристиками исправления ошибок при оптимальном декодировании, разработка практичных алгоритмов декодирования низкой сложности, которые могут обеспечить результаты, близкие к оптимальным для коротких кодов, все еще остается открытой проблемой. Перспективным является подход к декодированию коротких блочных кодов, в котором декодер использует алгоритм машинного обучения. Нейронные декодеры были введены как обобщение классических алгоритмов декодирования распространения достоверности (BP), где решетчатый граф в алгоритме BP рассматривается как нейронная сеть, а веса в решетчатом графе оптимизируются путем обучения нейронной сети [15]. К сожалению, этот подход требует многих умножений, которые обычно являются дорогостоящими операциями.

В [10] предлагается более дружественный к оборудованию подход, в котором декодирование минимальной суммы смещения дополнено обучаемыми параметрами смещения, метод не использует умножения и имеет количество параметров, вдвое меньшее, чем у мультипликативного алгоритма. Это ускоряет обучение и обеспечивает реальный путь к аппаратным архитектурам.

В работах [11, 16] предлагают нейронный декодер для циклических кодов, используя их свойство циклической инвариантности

Важно отметить, что алгоритмы классы распространения достоверности первоначально разработаны для кодов с низкой плотностью проверок на четность, что означает, что проверочная матрица этих кодов должна удовлетворять ряду условий, например, минимальное количество циклов малой длины. В общем случае линейные блоковые не являются кодами с низкой плотностью проверок на четность и эффективность алгоритмов класса распространения достоверности под вопросом. Рассмотренные работы показывают, что выигрыш по сравнению с традиционными алгоритмами алгебраического декодирования может быть получен за счет введения дополнительных весов при итеративном декодировании на графе.

1. Алгоритмы класса распространения достоверности

Известны следующие варианты алгоритма итеративного декодирования с обменом сообщениями илиалгоритмы класса распространение достоверности:

- Min-sum алгоритм (аналог Max-Log MAP)
- Sum-product алгоритм (аналог Log - MAP)
- Sum-product в E-представлении
- Вероятностное декодирования (tanh() метод тангенса гиперболического).

Декодирование на графе впервые предложенное в [1], и исследованное в [2] [3] [4] [5] заключается в следующем. Декодирование производят итеративно, таким образом , что каждая итерация производит априорную информацию для следующей итерации. Блок схема декодирования путем обмена сообщениями показана на Рис. 1. Каждое ребро графа соответствуют двум числам, известным как сообщения: сообщение от переменного узла к проверочному и наоборот. Сообщения являются метриками мягких решений. Перед началом декодирования они принимают значение ноль.

Рис. 1 Декодирование путем обмена сообщениями

Операция вычисления исходящих сообщений от переменного узла к проверочному узлу включает в себя следующее: мягкое решение, соотнесенным с этим узлом, суммируют со всеми входящими в этот узел сообщениями, кроме сообщения от проверочного узла к которому готовится это исходящее сообщение (Рис. 2).

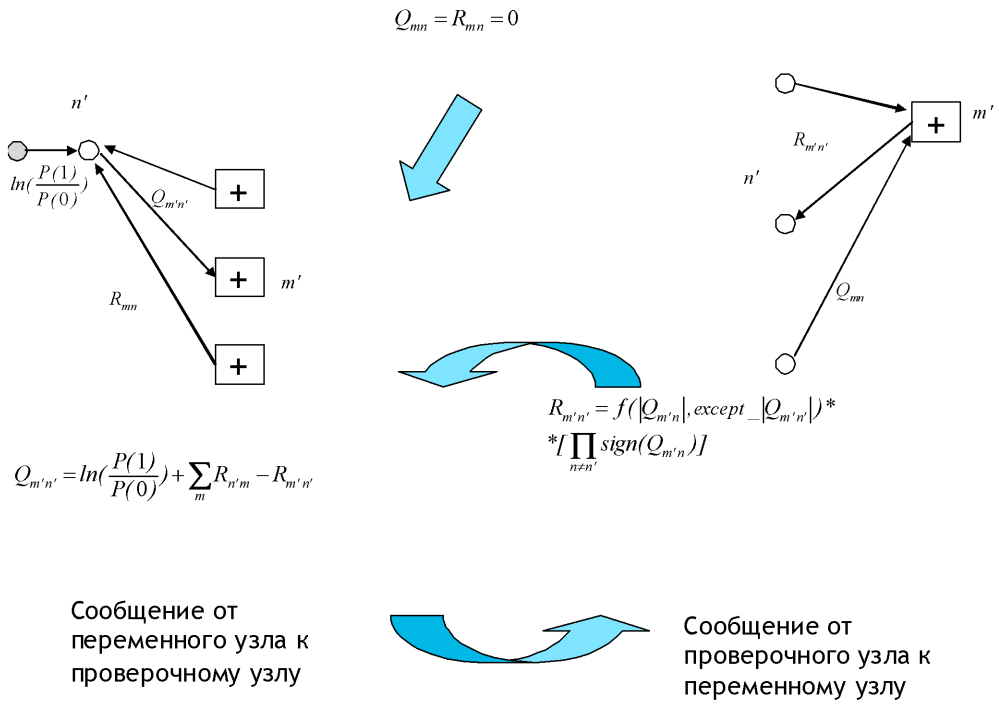

Рис. 2 Вычисление сообщений.

Операция вычисления исходящего сообщения проверочного узла состоит из двух частей:

- Определения знака сообщения
- Определения модуля сообщения.

Знак исходящего сообщения выбирают из соображений равенства нулю контрольной суммы. Каким образом определяют модуль исходящего сообщения, будет показано в следующем разделе.

Упрощенный способ формирования исходящего сообщения из проверочного узла (Offset Min-Sum)

Модуль исходящего сообщения вычисляют посредством функции f (Рис. 2).Эта функция решающая для логарифма отношения правдоподобия и обладает свойством коммутативности:

$$f(a,b,c,d) = f(a, f(b, f(c, f(d))))$$
$$f(d) = d$$

Рассмотрим следующие способы упрощенного вычисления функции f. [LeUngLeeWuPing 6]

Простейший способ известен как min-sum алгоритм. Абсолютная величина исходящего сообщения равна минимальному из входящих сообщений, участвующих в его формировании:

$$f(|Q_{m'n}|, except\_|Q_{m'n'}|) = min_{n \neq n'}(|Q_{m'n}|)$$

Другой метод совпадает с методом вычисления функции E в LOG MAP алгоритме. Функцию f определяют как:

$$f(a,b) = \min(a,b) + \beta,$$
$$\beta = \log(1+\exp(-|a-b|)) - \log(1+\exp(-|a+b|))$$

Таким образом β может быть аппроксимирован как:

$$\beta = \begin{Bmatrix} 1, |a-b| \le 1, |a+b| > 1 \\ -1, |a-b| > 1, |a+b| \le 1 \\ 0, else \end{Bmatrix}$$

Заметим, что знак дополнительного слагаемого β всегда обратный знаку исходящего сообщения, таким образом, амплитуда исходящего сообщения всегда меньше чем найденный минимум среди входящих сообщений, и может быть аппроксимирован как $x' = \max(x-\beta, 0)$ где x выход min-sum алгоритма. Более близкое приближение к методу tanh() выглядит следующим образом:

$$x' = \alpha \times \max(x-\beta, 0)$$

Такая аппроксимация известна как offset min-sum алгоритм [7, 8].

Планирование операций декодера: флудящие алгоритмы, вертикальный и горизонтальный шаффлинг.

Известен классический алгоритм декодирования LDPC: обмен сообщениями, который заключается в последовательном вычислении сообщений от переменных узлов к проверочным, и наоборот. То есть каждая итерация распадается на две полу-итерации. В отличие от него флудящий (“затопляющий”) алгоритм немедленно (как только это сообщение сформировано) осуществляет передачу сообщения на противоположный узел. Результатом данной операции является двукратное увеличение пропускной способности за счет выполнения большего числа итераций за то же самое время, однако, возрастает нагрузка на сетевые соединения, возрастает вероятность конфликтов при обращении к банкам памяти. Флудящие алгоритмы, в том случае, если узлы одной стороны графа Таннера реализуют параллельный алгоритм формирования сообщений, а узлы другой стороны последовательный алгоритм, называют алгоритмами шаффлинга (“тасование карт”). Если параллельный алгоритм реализуют переменные узлы, то говорят о вертикальном шаффлинге, если параллельный алгоритм реализуют проверочные узлы то говорят о горизонтальном шаффлинге. Тот узел, который осуществляет обработку данных в параллельном режиме, является мастером и инициирует обмен сообщениями с узлом, работающим в последовательном режиме. Здесь широко используют тот факт, что сообщение не участвует в формировании “встречного” сообщения, таким образом, каждый из узлов обменивающихся сообщениями получает самое “свежее” сообщение.

Напомним, что проверочные узлы это горизонтали матрицы H, а переменные узлы это вертикали матрицы. Рассмотрим в алгоритме горизонтального шаффлинга работу переменного узла – столбец матрицы H. Очевидно, что отличные от -1 элементы столбца представляют собой временную диаграмму обращений к этому узлу. Узел i_0 должен успевать формировать исходящее сообщение за время равное

$$t_{\min} = \min(P^{i_0,j_0} - P^{i_0,j_1}) \tag{1}$$

для любых j_0, j_1 таких, что $P^{i_0,j_0} > -1, P^{i_0,j_1} > -1$. Аналогичные условия можно поставить для горизонтального шаффлинга, естественно рассматривая вместо столбца строку.

<table>
<tr><td colspan="3" rowspan="2"></td><td colspan="3">Variable</td></tr>
<tr><td rowspan="2">Master</td><td colspan="2">Slave</td></tr>
<tr><td>Slow loop</td><td>Fast loop</td></tr>
<tr><td rowspan="3">Check</td><td colspan="2">Master</td><td>Flooding
(2 ways)</td><td>Flooding
(check way)</td><td>Shuffle
(Horizontal)</td></tr>
<tr><td rowspan="2">Slave</td><td>Slow loop</td><td>Flooding
(var. way)</td><td colspan="2" rowspan="2">edge
controled</td></tr>
<tr><td>Fast loop</td><td>Shuffle
(Vertical)</td></tr>
</table>

Рис. 3 Классификация алгоритмов шаффлинга [9].

Подобная архитектура представлена на Рис. 4, где проверочный узел дополнен двумя линиями задержки и двумя сумматорами, таким образом, что с переменного узла передают LLR $L_{m'n'} = ln(\frac{P(1)}{P(0)}) + \sum_m R_{n'm}$, а входящее сообщение $Q_{m'n'} = ln(\frac{P(1)}{P(0)}) + \sum_m R_{n'm} - R_{m'n'}$ получают непосредственно на проверочном узле, вычитая сохраненное в линии задержки исходящее сообщение $R_{m'n'}$. , а также из исходящего сообщения вычитают задержанное входящее.

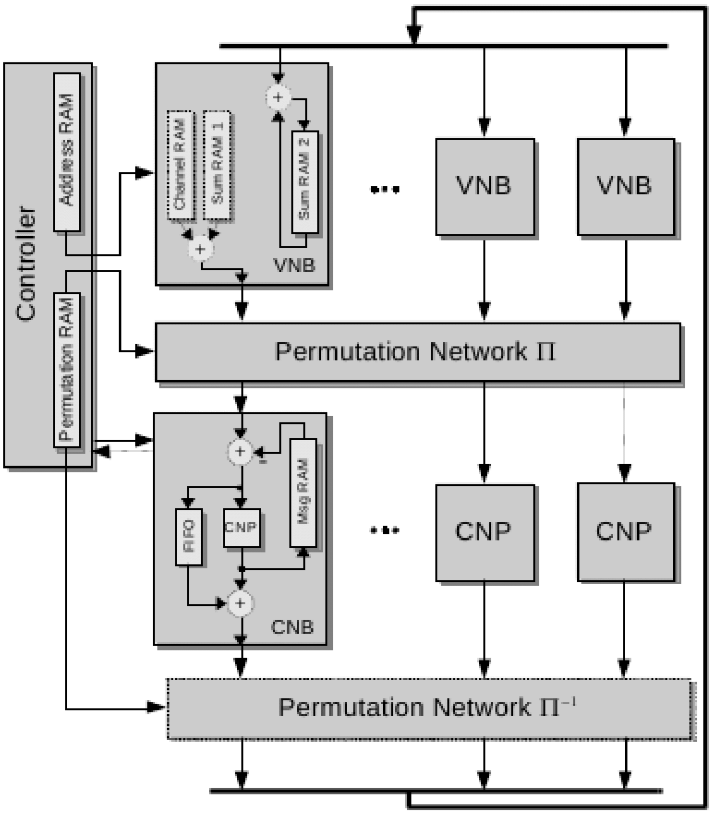

Рис. 4 Комбинированная архитектура [9].

2. Типичные конфигурации ошибок, вызванные дефектами графа Таннера.

Рассмотрим типичные возникающие конфигурации ошибок (Рис. 5). Наличие кодовых слов малого веса следует из существования циклов малой длины. Рассматривая цикл, как замкнутый путь проходящий по ребрам графа Таннера, можно заметить, что изменение всех бит в переменных узлах не вызовет изменения контрольных сумм в проверочных узлах, входящих в цикл. Так, например, в цикле длиной два (Рис. 5 *a*) контрольная сумма в проверочном узле вообще не зависит от значения переменного узла, поскольку они соединены двумя ребрами. В случае цикла-4 (Рис. 5 *b*) совместное изменение значений в двух переменных узлах не может быть детектировано контрольными узлами, входящими в цикл. В случае линейной комбинации циклов формируется «останавливающее множество» (Рис. 2 *c*), которое можно определить как подмножество проверочных вершин, для которого не существует переменных вершин, соединенных единственным ребром с каким-либо из проверочных вершин, входящих в это подмножество [15]. Множество переменных вершин, соединенных с только с «останавливающим» подмножеством, содержит наиболее вероятные комбинации ошибок. Действительно, ошибки в переменных узлах, вы-

званные, в том числе циклом, могут быть исправлены за счет других ребер связанных с данным узлом, их количество может быть оптимизационным критерием. При прочих равных условиях цикл, наружу которого смотрят больше ребер, является более безопасным с точки возникновения ошибок. Данный критерий известен как *ACE* [16], стр. 174. Все ошибочные комбинации, представленные на Рис. 5 кроме *e*) имеют *ACE*=0, поскольку не имеют ребер направленных наружу останавливающего множества. Функцией «соседства» будем называть количество проверочных (проверочных вершин со степенью 2) переменных узлов вовлеченных в останавливающее множество, как показано на (Рис. 5 *d*). Такое останавливающее множество всегда можно сформировать для информационных узлов с одинаковой минимальной степенью 3 , однако, если степень информационных узлов разная то ACE>0 (Рис. 5 *e*).

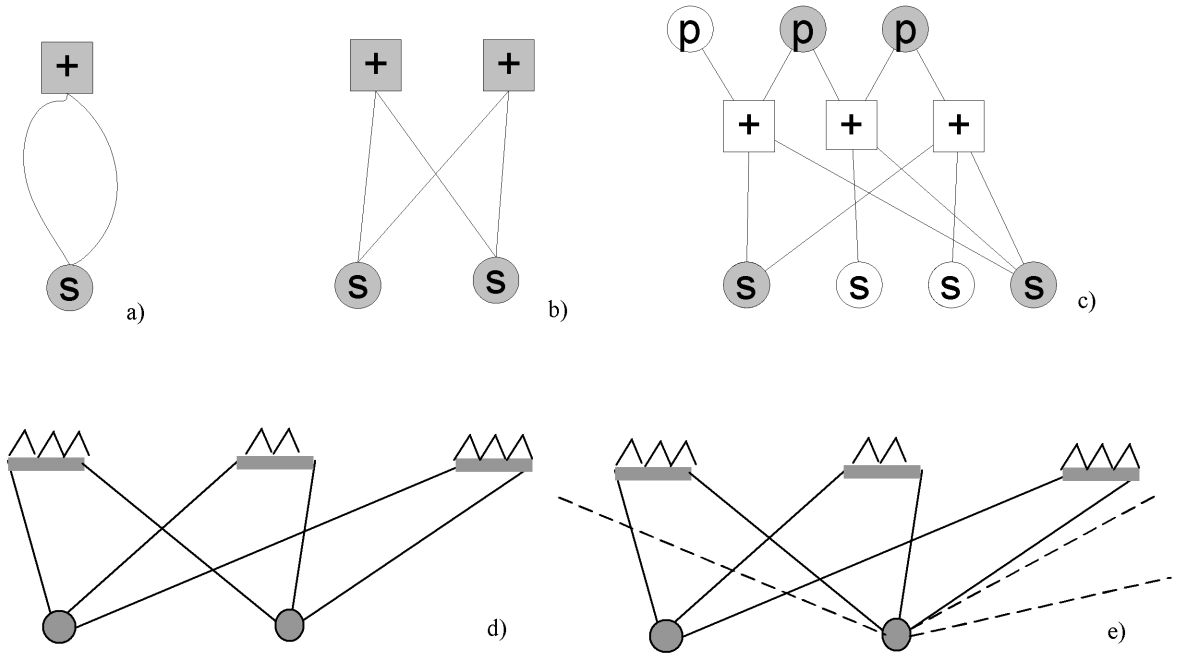

Рис. 5 Конфигурации ошибок

Предлагаемый алгоритм: мотивирующий пример

Подробней рассмотрим алгоритм из [10]. В нем формула $x' = \alpha \times \max\left(x - \beta, 0\right)$ заменятся формулой $x'_{m,n} = \alpha \times \max\left(x_{m,n} - \beta_{m,n}, 0\right)$, где $\beta_{m,n}$ является набором обучающих параметров, зависящим от номера переменного и проверочного узла. Параметры смещения могут быть обучены с помощью минипакетного стохастического градиентного спуска в автономной фазе. Потенциальная проблема обратного распространения возникает из-за того, что некоторые составляющие операции (ReLU(), min(), abs() и sign()) не дифференцируемы в определенных точках. Фреймворки глубокого обучения, такие как TensorFlow, преодолевают эту проблему, просто выбирая один из градиентов в точках недифференци-

руемости для обратного распространения через кусочно-дифференцируемые операции [17]. В [10] инициализируют смещения случайными значениями, взятыми из стандартного нормального распределения. Для обновления смещений после расчета градиентов используют оптимизатор Adam [14] со скоростью обучения 0,1.

Недостатком алгоритма из [10] является не использование алгоритмов флудящего типа, которые требуют меньшего числа итераций, а также случайное распределение начальных весов без учета структуры графа Таннера. В предлагаемом алгоритме использован алгоритм горизонтального шаффлинга с проверочным мастер-узлом. Начальная инициализация происходит путем обучения на графе, при этом если будут обнаружены дефекты графа Таннера, то будет назначен большой $\beta_{m,n}$, фактически выписан штраф. Инициализация случайным весом такого дефектного ребра приведет к тому, что дальнейшая инициализация будет заблокирована. Результату моделирования предлагаемого алгоритма и алгоритма из [10] приведены. Ожидается дальнейшее улучшение характеристик при глубоком обучении.

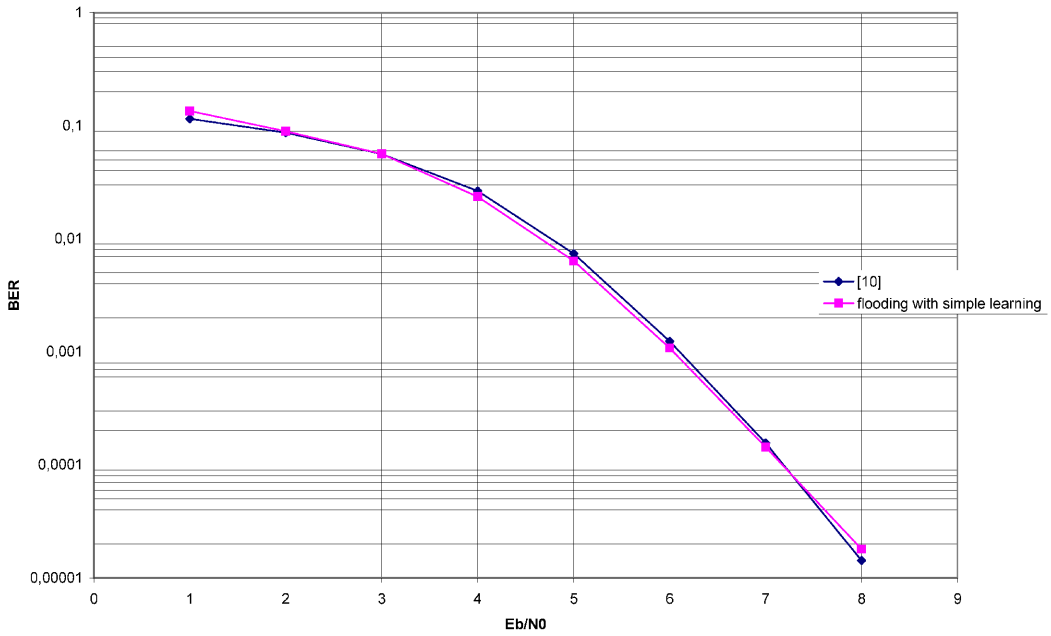

Рис. 6 Характеристики ofset min sum флудящего алгоритма с минимальным обучением для кода BCH (63,36,11)

Список литературы

1. R. M. Tanner, ``A recursive approach to low complexity codes," IEEE Trans. Inform. Theory, vol. IT-27, pp.533-547, Sept. 1981.
2. J. Pearl Probabilistic Reasoning in Intelligent System// San Mateo,CA, 1988
3. N. Wiberg, ``Codes and decoding on general graphs," Doctoral dissertation, 1996.
4. B. J. Frey, Graphical models for machine learning and digital communication, The MIT Press, Cambridge, Massachusetts, London, England, 1998.
5. G. D. Forney, ``Codes on graphs: normal realizations," IEEE Trans. Inform. Theory, vol. 47, pp. 520-548, Feb. 2001.
6. W. K. Leung, W. L. Lee A. Wu, L. Ping, An Efficient Implementation Technique of LDPC Decoder// Hong Kong Univ.
7. M. Jiang, Ch. Zhao,Li Zhang, E. Xu "Adaptive Offset Algorithm for Low-Density Parity Check Codes"// IEEE Communication Letters, Vol. 10, No 6, June 2006
8. J. Chen M. P. C. Fossorier "Near Optimum universal belief propagation based decoding of low-density parity check codes" // IEEE Transaction on Communication, vol. 53, No 8, pp 406-414, Mar. 2002
9. Frederic GUILLOUD "Generic Architecture for LDPC Codes Decoding"//PhD Paris 2004
10. Loren Lugosch and Warren J Gross. Neural offset min-sum decoding. In 2017 IEEE International Symposium on Information Theory (ISIT), pages 1361–1365. IEEE, 2017.
11. Nachmani E., Wolf L. Hyper-graph-network decoders for block codes //Advances in Neural Information Processing Systems. – 2019. – Т. 32.
12. Richardson T. J., Urbanke R. L. The capacity of low-density parity-check codes under message-passing decoding //IEEE Transactions on information theory. – 2001. – Т. 47. – №. 2. – С. 599-618.
13. Declercq D., Fossorier M., Biglieri E. Channel Coding: Theory, Algorithms, and Applications: Academic Press Library in Mobile and Wireless Communications. – Academic Press, 2014.
14. D. Kingma and J. Ba, "Adam: A method for stochastic optimization," International Conference on Learning Representations, 2015.
15. Eliya Nachmani, Yair Be'ery, and David Burshtein. Learning to decode linear codes using deep learning. In 2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton), pages 341–346. IEEE, 2016.

16. Chen X., Ye M. Cyclically equivariant neural decoders for cyclic codes //arXiv preprint arXiv:2105.05540. – 2021.
17. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., “TensorFlow: Large-scale machine learning on heterogeneous systems,” 2015, software available from tensorflow.org. [Online]. Available: http://tensorflow.org/