Нейросетевое декодирование на основе графа Таннера

Газизуллин Реналь Кварацхелия Георгий Шаглаев Илья

Университет Иннополис

15 декабря 2025 г.

Содержание

1. Введение
2. Neural Belief Propogation
3. Результаты

Тема проекта

Проект 3. **Применение нейронных сетей для декодирования линейных кодов. Подход на основе графа Таннера**

Задачи:

- Реализовать алгоритм из представленной статьи
- Проверить его работу на МПП-кодах из стандарта 5G
- Сравнить производительность с алгоритмом Sum-Product

Мотивация

В современном мире важна скорость и простота, при этом и терять в качестве нельзя.
Имеющийся алгоритм Back Propogation (BP) хорошо работает с LDPC кодами, но для этого требует большие длины блоков. Для коротких и умеренных HDPC кодов обычный BP даёт плохие результаты, по сравнению с оптимальным Maximum Likelihood (ML).
Для решения проблемы с высокой вычислительной сложностью и проблемами в коротких кодах у BP, предлагается использовать нейронные сети и обучение.

Граф Таннера

Определение

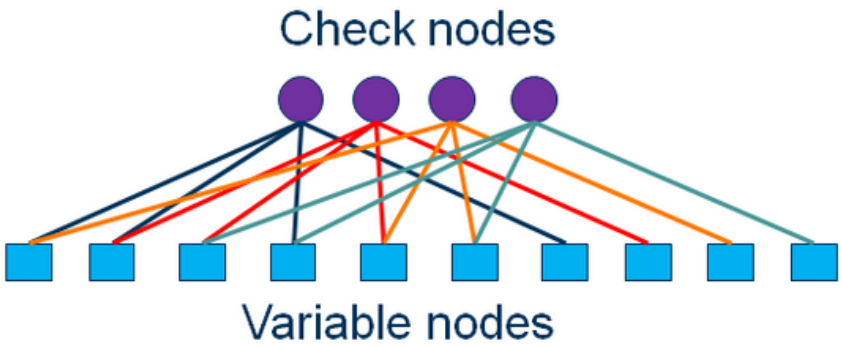
Граф Таннера — это двудольный граф, который описывает структуру линейного блочного кода через его проверочную матрицу H.
Граф Таннера $G = (V \cup C, E)$ состоит из:

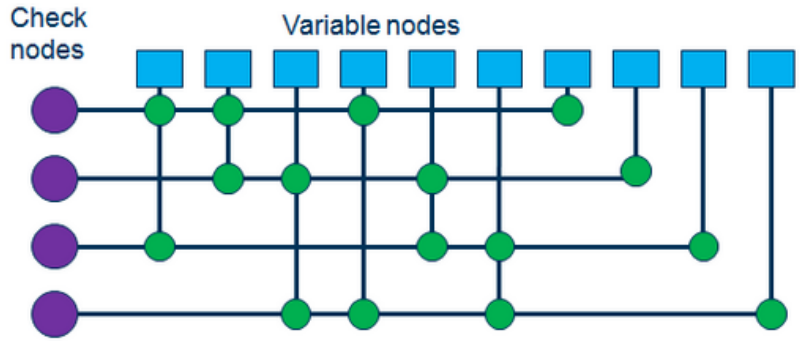
- **Узлов переменных**: $V = \{v_1, \ldots, v_n\}$, каждому узлу соответствует кодовый бит x_j.
- **Узлов проверок**: $C = \{c_1, \ldots, c_m\}$, каждому узлу соответствуем проверочное уравнение:

$$\sum_{j=1}^{n} H_{ij} x_j = 0 \pmod 2$$

- **Рёбер**: между v_i и c_j существует ребро тогда и только тогда, когда $H_{ij} = 1$.

$$\mathbf{H}_{M\&N} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$$





Визуализация примера графа Таннера.

Belief Propagation: Trellis-представление

- Рассматривается линейный код с проверочной матрицей H и соответствующим **графом Таннера**.
- Алгоритм belief propagation (BP) является **итеративным алгоритмом обмена сообщениями** между переменными (variable nodes) и проверками чётности (check nodes).
- BP можно представить как **trellis**, полученный разворачиванием алгоритма по итерациям $t = 1, \ldots, T$.
- На каждой итерации выполняются два шага:
  - сообщения $m_{v \to c}^{(t)}$ от переменных к проверкам,
  - сообщения $m_{c \to v}^{(t)}$ от проверок к переменным.
- Обновление сообщений:

$$m_{v \to c}^{(t)} = \ell_v + \sum_{c' \in \mathcal{N}(v) \setminus c} m_{c' \to v}^{(t-1)}$$

$$m_{c \to v}^{(t)} = 2 \tanh^{-1} \prod_{v' \in \mathcal{N}(c) \setminus v} \tanh\left(\frac{m_{v' \to c}^{(t)}}{2} \right)$$

- После T итераций вычисляется апостериорный LLR:

$$L_v = \ell_v + \sum_{c \in \mathcal{N}(v)} m_{c \to v}^{(T)}$$

Ограничения классического Belief Propagation

- Алгоритм belief propagation является **оптимальным только на деревьях**.
- Реальные линейные коды (например, LDPC) имеют **короткие циклы в графе Таннера**.
- Наличие циклов приводит к:
  - корреляции сообщений,
  - повторному распространению одной и той же информации.
- В trellis-представлении это проявляется как **самовлияние сообщений** между итерациями.
- Правила обновления BP:
  - фиксированы,
  - одинаковы для всех рёбер и итераций,
  - не адаптируются к структуре кода.
- Следствие: **заметный разрыв с ML-декодированием** для кодов малой и средней длины.

Идея Neural Belief Propagation

- Belief propagation представлен как **trellis вычислительного графа** из T итераций.
- Классический BP использует **фиксированные правила обновления сообщений**.
- Основная идея: **ввести обучаемые веса** на рёбрах trellis, сохранив структуру BP.
- Обновление сообщений с весами:

$$m_{v\to c}^{(t)} = w_{vc}^{(t)} \ell_v + \sum_{c' \in \mathcal{N}(v)\setminus c} w_{c'v}^{(t)} m_{c'\to v}^{(t-1)}$$

- Веса позволяют:
  - компенсировать эффекты коротких циклов,
  - адаптировать алгоритм к конкретному коду,
  - приблизиться к ML-декодированию.
- Получается **нейронный декодер**, структурно эквивалентный BP.

Два варианта весов в Neural Belief Propagation

- В trellis-представлении Neural BP возможны два способа задания весов.
- **Unrolled Neural BP**:
  - разные веса на каждой итерации t,
  - архитектура глубокой нейросети,
  - высокая гибкость, но много параметров.
- **RNN Neural BP (shared weights)**:
  - одинаковые веса для всех итераций,
  - рекуррентная архитектура,
  - меньше параметров и лучшее обобщение.
- Экспериментально показано, что RNN-вариант достигает сравнимой производительности при существенно меньшей сложности.

Обучение Neural Belief Propagation

- Вход сети:
  - логарифмические отношения правдоподобия (LLR)

$$\ell_v = \log \frac{P(y_v|x_v = +1)}{P(y_v|x_v = -1)}$$

  - канал AWGN.
- Выход сети:
  LLR L_v после T итераций BP.
- Используется Cross-Entropy Loss.
- Обучение выполняется методом градиентного спуска.
- После обучения веса фиксируются и используются для декодирования.

Нулевое кодовое слово

- Рассматриваются **линейные блочные коды**.
- Для линейного кода:
  - нулевое слово 0 всегда является кодовым,
  - все кодовые слова эквивалентны с точки зрения ошибок.
- Используется симметричный канал:
  - BPSK-модуляция,
  - аддитивный белый гауссов шум (AWGN).
- Симметрия канала означает:

$$P(y|x) = P(-y| - x)$$

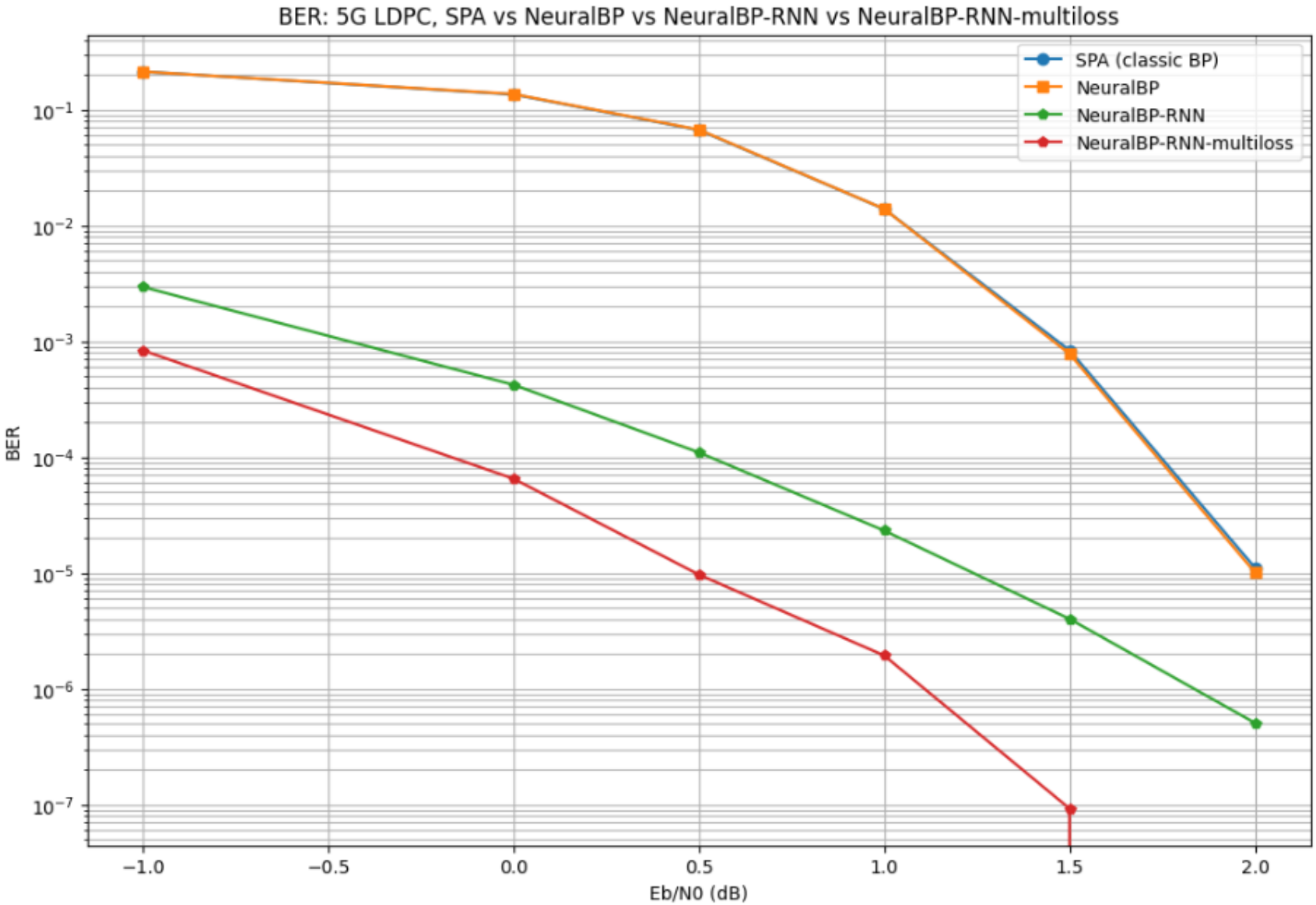
- Следствие:
  - статистика ошибок не зависит от переданного кодового слова,
  - обучение на 0 эквивалентно обучению на всём коде.
- Это позволяет существенно:
  - упростить обучение,
  - сократить объём требуемых данных

Эксперимент

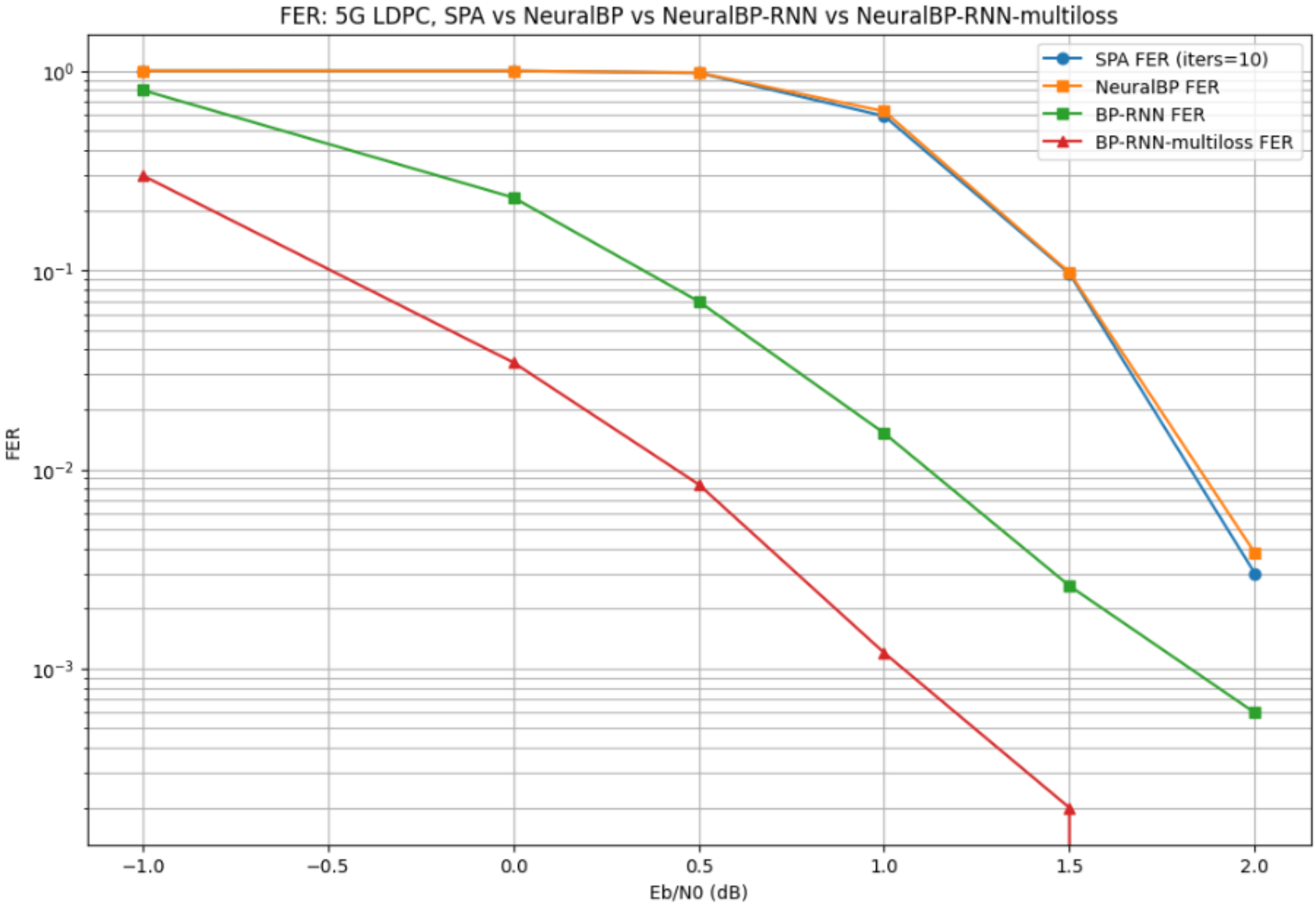
- Взяли протоматрицу из репозитория.
- Преобразовали её в матрицу $H \in \{0,1\}^{1472 \times 2176}$.
- Обучили Neural BP, Neural BP с RNN.
- Посчитали на каждом из алгоритмов ошибки: BER(bit error rate) и FER(frame error rate).
- Построили графики

Исходный код нашего проекта можно найти в репозитории github.

Результаты


BER: 5G LDPC, SPA vs NeuralBP vs NeuralBP-RNN vs NeuralBP-RNN-multiloss
SPA (classic BP)
NeuralBP
NeuralBP-RNN
NeuralBP-RNN-multiloss
BER
10^{-1}
10^{-2}
10^{-3}
10^{-4}
10^{-5}
10^{-6}
10^{-7}
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Eb/N0 (dB)

Результаты


FER: 5G LDPC, SPA vs NeuralBP vs NeuralBP-RNN vs NeuralBP-RNN-multiloss
SPA FER (iters=10)
NeuralBP FER
BP-RNN FER
BP-RNN-multiloss FER
FER
10^{0}
10^{-1}
10^{-2}
10^{-3}
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
Eb/N0 (dB)

Спасибо за внимание!