Error Correction Code Transformer
Reproducibility Challenge – proposal
20212245 김희서

1. summary of the paper you chose.
해당 연구는 기존의 방법과 달리 기계 번역와 같은 자연어 처리에서 주로 사용되던 transformer architecture 기반의 model free decoder를 임의의 블록길이를 가진 선형 코드(LDPC, BCH 등)의 soft decoding에 처음으로 제안하였고, adapted masked self-attention module을 통해 algebraic code와 bits 간의 상호작용을 모델에 적용한 것이 큰 연구의 방향입니다.

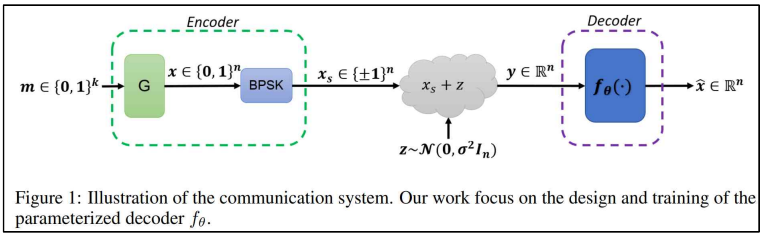


Figure 1: Illustration of the communication system. Our work focus on the design and training of the parameterized decoder f_θ.

해당 그림은 전반적인 통신 시스템에서의 인코딩, 디코딩 과정을 나타낸 것입니다.
입력 메시지 $m \in (0,1)^k$는 generator matrix G에 의해서 codeword $x \in (0,1)^n$으로 인코딩이 되고, BPSK과 AWGN 채널을 통과하면 $y = x_s + z$인 채널 출력이 되는 데, 이때 x_s는 x의 BPSK modulation이고, z는 random noise입니다. $f : R^n \rightarrow R^n$인 디코더는 인코딩 된 x값을 최대한 잘 추정하는 것이 목표($\hat{x} = f(y)$)입니다.

저자는 디코딩을 위한 model-free approach의 하나의 부분으로 transformer architecture 적용을 제안하였고, 전처리 과정 $\tilde{y} = h(y) = [|y|, s(y)]$에서는 채널 출력인 y의 절댓값과 신드롬($Hy_b \in (0,1)^{n-k}$)값을 결합함으로써, $2n-k$ 차원의 벡터를 만들었습니다. 후처리 과정($\hat{x} = yf(h(y))$)에서는 clean-up 단계 거치면서, 예측된 codeword 값을 구합니다. 해당 논문에서는 θ에 의해 매개변수화 되기에 $\hat{x} = y \times f_\theta(h(y))$로 표시됩니다.

transformer의 경우, 우선 입력 sequence를 각 element에 대한 positional 임베딩을 결합한 고차원공간으로 embedded 되고, 그 값은 다수의 normalized self-attention과 feed-forward blocks를 통해 전파됩니다. 그리고 self-attention의 경우, 우선 입력되는 벡터들로부터 3개의 벡터인 key, value, query를 만든 후, query와 key의 내적으로 query와 key와의 연관성을 내적을 통해 구한 다음에, key 벡터 사이즈의 제곱근으로 나눠서 안정적인 gradient을 얻습니다.

그리고 그 값을 softmax 점수를 구하고, key에 대한 의미적인 결과 값인 'value'과 곱한여, 중요하지 않은 값을 제거합니다. 마지막으로 그 값을 모두 더하여 self-attention의 출력값을 얻습니다. 추가적으로 만일 h개의 self-attention function이 입력에 적용되면 multi-head self-attention(MH-SA)이 됩니다. 지금까지 기존의 transformer의 방법이라면, 이것을 '**error correction code**'를 위한 transformer로 적용하면, 우선 채널 출력의 n차원의 y을 high dimensional embedding $(\phi_i)_{i=1}^{2n-k}$을 합니다.

$(\phi_i) = \begin{cases} |y_i| W_i, & \text{if } i \le n \\ (1 - 2(s(y))_{i-n+1}) W_i, & otherwise \end{cases}$ 로 표현되는데, $(W_j \in R^d)_{j=1}^{2n-k}$은 bit position에 따라 정의되는 one-hot encoding으로 정의되고, I의 값에 따라 다음의 식으로 표현되는 이유는 $\tilde{y} = h(y) = [|y|, s(y)]$의 값이 채널 출력의 절댓값(channel output reliability)과 신드롬값을 통해 구성되기 때문이고, $(1 - 2(s(y))_{i-n_1})$으로 표현되는 이유는 BPSK 과정 때문입니다. 즉 해당 과정은 position을 나타내는 아래첨자를 통해 bit-wise embedding이 되는 것을 알 수 있고, I의 값에 따라 scaled 되는 것을 알 수 있기에, 'scaled bit-wise embedding'임을 알 수 있습니다. 이 과정을 저자는 '**positional reliability encoding**'이라고 언급하였습니다.

그리고 error correction code를 위한 'self-attention'을 적용하였는데, 함수와 알고리즘은 다음과 같이 표현된다.

$g(H) : (0,1)^{(n-k) \times n} \rightarrow -\infty, 0^{2n-k \times 2n-k}$, $A_H(Q,K,V) = Softmax(\frac{QK^T + g(H)}{\sqrt{d}})V.$

Algorithm 1: Mask construction Pseudo Code

```
function g(H)
    l,n = H.shape
    k = n-l
    mask = eye(2n-k)
    for i in range(0,n-k) do
        idx = where(H[i]==1)
        for j in idx do
            mask[n+i,j] = mask[j,n+i] = 1
            for k in idx do
                mask[j,k] = mask[k,j] = 1
    return −∞(¬ mask)
```

그리고 저자는 adapted-mask self attention mechanism을 제안하였는데, 해당 방법은 모든 비트 쌍 간의 관계에 대한 정보를 포함하는 대칭 마스크를 제안하였고, 행에서 1의 위치를 unmask 합니다. 즉, fig2의 c에서 check node c_0와 variable node X_0가 연결되어 parity check matrix의 값에 1이 배치되어야 하고, 그 해당 되는 값이 unmask가 된 것을 확인할 수 있는데, 이것은 디코딩 과정에서 신드롬값처럼 비트들이 서로 연결되어 있고, 서로 영향을 줄 수 있기에, unmask를 했습니다.

그리고 fig2의 b를 보면, 테너 그래프에 의해 가능한 관계보다 큰 구조 때문에, 디코딩에서 더 flexibility라는 이점을 가집니다. 즉, (b)의 경우, mask의 크기는 $7 \times 7 = (n-k) \times (n-k)$이지만, 제안된 masking은 10×10의 matrix이기에, parity check equation을 넘는 영역에서 서로 영향을 비트들이 줄 수 있기

때문이고, mask보다 큰, 즉 n+i의 신드롬 비들은 algorithm 1의 8번째 줄에서 'mask[n+i,j]=mask[j,n+i]=1'로 표현되는데, syndrome bit에 해당하는 1의 값들이 parity check equation을 정의하기 때문입니다. 즉, syndrome의 값은 parity check equation의 column의 linear combination으로 표현할 수 있기 때문입니다.

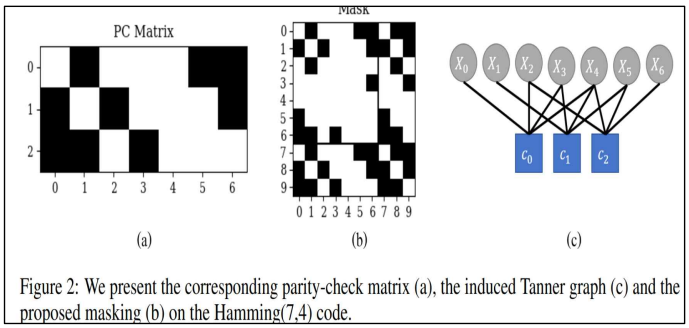


Figure 2: We present the corresponding parity-check matrix (a), the induced Tanner graph (c) and the proposed masking (b) on the Hamming(7,4) code.

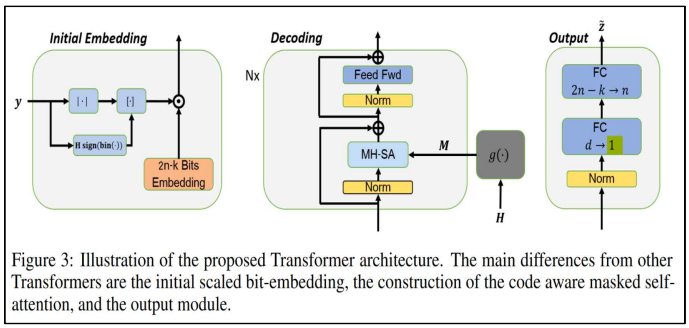


Figure 3: Illustration of the proposed Transformer architecture. The main differences from other Transformers are the initial scaled bit-embedding, the construction of the code aware masked self-attention, and the output module.

다음의 fig3은 전반적인 transformer 아키텍처로, 앞서 언급한 방법을 활용하여 진행됩니다. initial embedding에서는 |y|인 channel output의 절댓값인 reliability와 신드롬값을 결합한 $2n-k$ 차원의 벡터와 high 'd' dimensional embedding을 합니다. 디코더에서는 normalization layer에 의해 인터리빙 된 multi-head self-attention(MH-SA)과 feed-forward(Feed Fwd) layer로 구성됩니다.

그리고 output에서는 1번째 fully connected layer에서 element-wise embedding에서 one-dimensional $2n-k$ vector로 줄이고, 2번째 layer에서는 'soft decoded noise($\tilde{z}$)'를 나타내는 n 차원 벡터로 줄입니다. 그리고 loss function의 경우, multiplicative noise $\tilde{z}$를 학습하는 것이 목표입니다. 이때, $\tilde{z}_s$은 soft multiplicative noise로 ($y = x_s\tilde{z}_s$), $\tilde{z}_s = \tilde{z}_s x_s^2 = yx_s$를 얻을 수 있습니다.

즉 x_s는 x의 BPSk modulation인 {-1, +1}이기에, 제곱하면 1이 되어, $\tilde{z}_s = yx_s$이 되고, 이를 통해 loss function 값을 구하면, $L = -\sum_{i=1}^{n} \tilde{z}_i \log(f_\theta(y)) + (1-\tilde{z}_i)\log(1-f_\theta(y))$이 됩니다.

시뮬레이션에서는 저자가 제안한 방법의 우수성을 나타내기 위해서, Polar, BHC, LDPC의 다양한 코드의 조건에서 진행하였고, 복잡성과 masking의 유무, self-attention mechanism의 영향에 따른 성능 차이를 보여주었습니다.

2. what you chose that specific paper?

- suspicious bit?

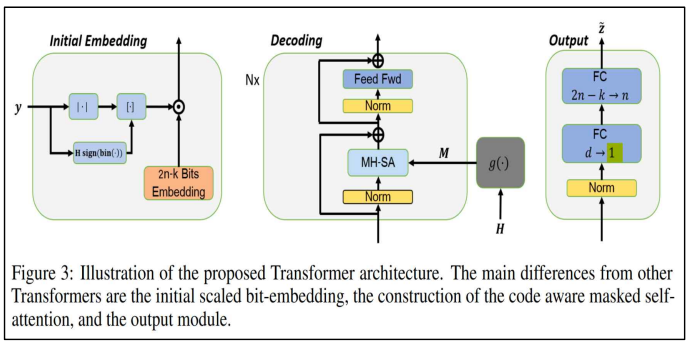
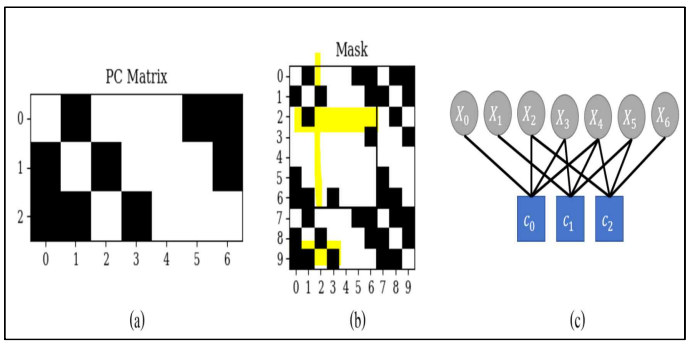


Figure 3: Illustration of the proposed Transformer architecture. The main differences from other Transformers are the initial scaled bit-embedding, the construction of the code aware masked self-attention, and the output module.



1) dimension : 해당 논문의 경우, NeurIPS 2022에 올라온 논문으로 아카이브에 올라온 논문 기준으론 아키텍처의 구조를 나타내는 figure와 제안된 masking에서의 오류가 있어 이를 확인하려고 합니다.

즉, 1번째 layer는 element-wise embedding을 one-dimensional $2n-k$ 벡터로 줄이는 것인데, 표현 방법에서의 '1'과 2번째 layer에서의 $2n-k$는 동일한 값이여야 하는데, 표현을 다르게 한 것이 실수인지, 다른 것을 의미하는 것인지, 아니면 one-dimensional, $2n-k$을 각각 표현하기 위해서 기재한 것인지 파악할 것입니다.

2) performance : Figure 4.(c)를 보면, RNN를 사용한 이유가 transformer를 사용하였기에, 성능을 비교했다고 생각했고, 저자는 논문에서 절반의 layer 수인 10을 사용하고, 모델 embedding dimension을 128로 감소하면서, 파라미터의 수도 2M로 급감하였다. 즉, 복잡도가 많이 감소하였는데, 성능도 개선되었습니다.

보통 연구의 경우, 복잡도와 성능의 tradeoff 관계 때문에, 복잡도를 개선하면서, 이전 연구의 성능을 최대한 비슷하게 따라가거나 성능 개선만을 집중하는데, 모두 다 개선 된 점에 대해서 성능을 확인해봐야 한다고 생각하였습니다.

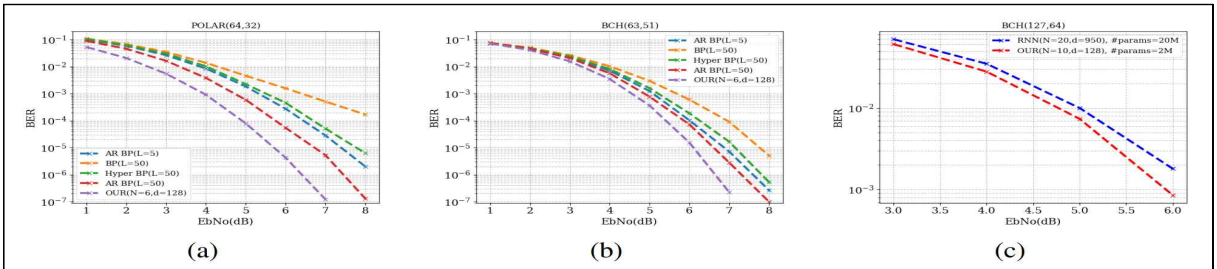


Figure 4: BER for various values of SNR for (a) Polar(64,32) and (b) BCH(63,51) codes. Comparison with the best model of Bennatan et al. [2] on BCH(127,64) code (c).

- importance?

제가 속한 연구실의 연구 주제는 채널코딩 분야이고, 이때 오류 정정 코드는 노이즈 있는 채널에서 데이터의 신뢰성 있는 전송을 보장하는, 통신 물리 계층의 중요한 요소입니다.

최근 여러 'Eliya Nachmani'와 같은 선행 연구자가 2017년부터 채널코딩 분야에 딥러닝을 적용하면서 기존의 디코딩보다 좋은 성과를 보여주었습니다. 하지만, 해당 방법의 경우, check node와 variable node를 연결하는 edge에 서로 다른 딥러닝을 통해 최적화된 weight를 할당함으로써 성능을 개선시킨 방법이였지만, 복잡도가 증가하는 단점이 있었습니다.

해당 논문에서는 그런 문제를 처음으로 elements 사이의 복잡한 interaction을 표현하는 능력 때문에 기계 번역 등에서 사용되는 특징을 가진 Transformers를 통해, 임의의 길이를 가진 linear 코드에서의 soft decoding을 구현하였습니다.

해당 방법은 기존의 방법보다 우수한 성능을 보여주었고, 처음으로 transformer가 적용된 만큼 향후 관심 갖고 연구할 만한 주제가 될 것이라고 생각합니다.

즉, 기존의 딥러닝를 적용한 채널코딩 연구의 경우, RNN, CNN와 같은 기존의 기법을 채널코딩에 알맞게 변형하여, 적용하는 방식으로 진행되거나, normalized min-sum, offset min-sum에서의 correction factor 값을 최적화하거나, edge에 할당된 weight의 값을 기준으로 pruning을 하는 연구가 주로 연구가 되었는데, 새로운 transformer라는 기계번역 등에서 활용된 방법을 적용했다는 것이 관심을 가지게 된 이유입니다.

3. experiment plan - what experiment are you doing?

1) 우선 공식적인 코드가 있고, suspicious의 이유로 성능 평가를 다시 해볼 것입니다.

2) 딥러닝을 적용한 채널코딩의 경우에서는 information bit의 길이가 증가함에 따라, 계산이 복잡해지고, 그런 이유로 짧은 길이에서 주로 연구가 되는데, 해당 기법 또한 그러한지 확인하여, 한계를 파악하려고 합니다.

3) 다양한 attention이 있기에, 이를 적용해보고자 합니다.

4) 해당 논문의 경우, 시뮬레이션을 다양한 길이와 코드에 적용하여, 성능의 우수성을 제안한다고 하였고, LDPC, Polar Codes, BCH Code에 적용한다고 하였는데, BER 성능에서는 Polar code와 BCH code만을 적용하였습니다.

이것은 HDPC인 BCH와 달리 LDPC는 low density parity check code에서는 성능 개선이 되지 않기에, 이를 기재하지 않은 것 인지 여부를 파악하고자 합니다.

4. GPU assets - how much additional resource?

NVIDIA Geforce GTX 750 Ti

5. codebase - official code, or built from scratch?

https://github.com/yonilc/ecct : official code.