# Linear Transformer Diffusion Model

Mario Hernandez

July 2024

**Abstract**

In this work we propose a Transformer Diffusion model for decoding on 5G NR LDPC codes. We achieve linear time complexity on par with state of the art 5G LDPC decoders. We propose adversarial training.

## 1 Introduction

Reliable digital communication is critical in the modern information age, demanding the design of robust codes that can withstand noisy transmission channels. Among the various coding schemes, low-density parity-check (LDPC) codes have emerged as a prominent solution, particularly in the context of 5G New Radio (NR). LDPC codes are favored for their high error-correcting performance and efficient decoding capabilities. However, the optimal decoding of LDPC codes remains a computationally challenging problem, often requiring sophisticated and resource-intensive algorithms.

Recent advancements in deep learning have introduced new possibilities for enhancing the decoding process. Transformer models, known for their effectiveness in natural language processing, have shown promise in various domains due to their ability to model long-range dependencies. However, their application to error correction codes, particularly LDPC codes, has been limited.

This paper proposes a model refinement for the context of 5G based on a proposed model in (ar5iv). By integrating diffusion processes with transformer architectures to create a Linear Transformer Diffusion Model tailored for decoding 5G LDPC codes. The diffusion process, inspired by the denoising diffusion probabilistic models (DDPM), introduces a new dimension to the decoding strategy, potentially improving both accuracy and computational efficiency.

## 2 Related Works

The decoding of error-correcting codes, particularly low-density parity-check (LDPC) codes, has seen significant advancements in recent years due to the integration of machine learning techniques. We review the key contributions in this area, focusing on neural network-based decoders, diffusion models, and their applications to LDPC codes.

### 2.1 Neural Network-Based Decoders

Neural network-based decoders have emerged as powerful tools for improving the performance of LDPC code decoding. A notable approach involves using neural networks to optimize belief propagation (BP) algorithms. The work by Bennatan et al. (2022) explores the use of neural networks to enhance BP decoding by learning optimal parameters for the Tanner graph, leading to improved decoding performance under various noise conditions (ar5iv) (ar5iv). Another significant contribution is the development of Neural Min-Sum (NNMS) decoders, which use neural networks to perform the min-sum algorithm more efficiently, thereby reducing the complexity and enhancing the error-correcting capabilities of LDPC codes (ar5iv).

### 2.2 Denoising Diffusion Probabilistic Models

Denoising Diffusion Probabilistic Models (DDPM) have been applied to various domains for generative tasks and have shown promise in the field of error correction. The work by Ho et al. (2020) demonstrated the effectiveness of diffusion models in generating high-quality samples by reversing a diffusion process. This concept has been adapted to error correction codes, where the diffusion process models the corruption of codewords during transmission, and the reverse process aims to recover the original codewords. The integration of diffusion models with neural network-based decoders has shown potential in handling complex noise patterns and improving decoding accuracy (ar5iv) (ar5iv).

### 2.3 Applications to LDPC Codes

Several studies have explored the application of these advanced decoding techniques to LDPC codes, particularly in the context of 5G communication systems. The paper by Shental et al. (2021) introduces a framework for using DDPMs to decode LDPC codes, demonstrating significant improvements in decoding performance compared to traditional algorithms. This approach leverages the noise modeling capabilities of diffusion processes to enhance the robustness of LDPC code decoding in noisy environments (ar5iv). Additionally, the work by Nachmani et al. (2018) on learning to decode protograph LDPC codes with neural networks has provided a foundation for integrating these models with structured codes used in 5G NR, further optimizing the decoding process for real-world applications (ar5iv).

## 3 Background

### 3.1 Communication Channels

In digital communication systems, data is transmitted over channels that introduce noise and errors. A common model is the Additive White Gaussian Noise

(AWGN) channel, where the received signal $y$ is expressed as:

$$y = x + z \tag{1}$$

where $x$ is the transmitted signal and $z$ is Gaussian noise with zero mean and variance $\sigma^2$.

## 3.2 Pipeline from Input Bits to Decoded Bits

The process involves several steps, from encoding the input message to decoding it at the receiver end. The main components of the pipeline are illustrated in Figure **??**.

### 3.2.1 Encoding

The input message $m \in \{0,1\}^k$ is encoded using a generator matrix $G$ into a codeword $x \in \{0,1\}^n$, satisfying $Hx = 0$ where $H$ is the parity-check matrix of size $(n-k) \times n$.

### 3.2.2 Modulation

The codeword $x$ is modulated for transmission. In this example, Log-Likelihood Ratios (LLRs) are used for modulation:

$$x_s = 2x - 1 \tag{2}$$

where $x_s \in \{\pm 1\}^n$.

### 3.2.3 Transmission over AWGN Channel

The modulated signal $x_s$ is transmitted over an AWGN channel:

$$y = x_s + z \tag{3}$$

where $y \in \mathbb{R}^n$ is the received signal and $z \sim \mathcal{N}(0, \sigma^2 I)$ is the Gaussian noise.

### 3.2.4 Decoding

The goal of the decoder $f : \mathbb{R}^n \to \mathbb{R}^n$ is to provide a soft approximation $\hat{x}$ of the transmitted codeword:

$$\hat{x} = f(y) \tag{4}$$

## 3.3 Transformers

The Transformer model, introduced by Vaswani et al. (2017), is a novel neural network architecture that relies entirely on self-attention mechanisms to draw global dependencies between input and output. The self-attention mechanism is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{5}$$

where $Q$, $K$, and $V$ are the query, key, and value matrices, respectively, and $d_k$ is the dimensionality of the keys.

## 3.4 Transformer Architecture for Decoding

We employ a Transformer architecture as part of a model-free approach for decoding. Preprocessing steps transform the channel output $y$ into a vector $\tilde{y}$ of dimensionality $2n - k$:

$$\tilde{y} = h(y) = [|y|, s(y)] \tag{6}$$

where $s(y) = Hy_b$ and $y_b$ is the binary code syndrome obtained by multiplying $y$ with the parity-check matrix $H$.

## 3.5 Reverse Diffusion Process

The decoding involves an iterative reverse diffusion process to refine the codeword estimate. Each step $t$ is controlled by a variance schedule $\beta_t$:

$$y = x_t - \sqrt{\beta_t} z, \quad z \sim \mathcal{N}(0, I) \tag{7}$$

### 3.5.1 Lie-Trotter Splitting

This method splits the reverse diffusion into condition and diffusion subproblems:

$$r_{t-\frac{1}{2}} = r_t - \eta \nabla_{r_t} \log p(c|r_t) \tag{8}$$

$$r_{t-1} = r_{t-\frac{1}{2}} + \mathcal{N}(0, \sigma_t I) \tag{9}$$

### 3.5.2 Strang Splitting

A higher-order method that improves accuracy:

$$r_{t-\frac{1}{2}} = r_t - \frac{\eta}{2} \nabla_{r_t} \log p(c|r_t) \tag{10}$$

$$r_{t-1} = r_{t-\frac{1}{2}} + \mathcal{N}(0, \sigma_t I) \tag{11}$$

$$r_{t-1} = r_{t-1} - \frac{\eta}{2} \nabla_{r_{t-1}} \log p(c|r_{t-1}) \tag{12}$$

## 3.6 Bipartite Graph Structure of the Parity Check Matrix

Error correction codes such as Low-Density Parity-Check (LDPC) codes are often represented using bipartite graphs, known as Tanner graphs. A Tanner graph is a bipartite graph that consists of two sets of nodes: variable nodes (VNs) and check nodes (CNs). The edges connect VNs to CNs based on the parity-check matrix $H$. For a codeword $c$ and parity-check matrix $H$, the relation is given by:

$$Hc^T = 0 \tag{13}$$

where $H$ is an $(n - k) \times n$ matrix over $\mathbb{GF}(2)$, representing the parity-check constraints.

## 3.7 Denoising Diffusion Probabilistic Models (DDPM)

DDPMs are a class of generative models that iteratively denoise a variable corrupted by Gaussian noise. The forward process adds noise to the data $x_0$ according to a variance schedule $\beta_t$:

$$q(x_t|x_{t-1}) = \mathcal{N}(x_t; \sqrt{1-\beta_t}x_{t-1}, \beta_t I) \tag{14}$$

The reverse process aims to recover $x_0$ by iteratively removing the noise, modeled as:

$$p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, t), \sigma_\theta(x_t, t)) \tag{15}$$

## 3.8 Generative Adversarial Networks (GANs)

GANs, introduced by Goodfellow et al. (2014), consist of two neural networks, the generator $G$ and the discriminator $D$, which compete in a zero-sum game. The objective of the GAN is defined as:

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1 - D(G(z)))] \tag{16}$$

where $p_{\text{data}}(x)$ is the data distribution and $p_z(z)$ is the prior distribution of the latent variable $z$. The generator aims to produce data samples that are indistinguishable from real data, while the discriminator attempts to differentiate between real and generated samples.

## 3.9 Generative Adversarial Networks (GANs) for Communication Channels

Generative Adversarial Networks (GANs) are a class of machine learning frameworks where two models, a generator (G) and a discriminator (D), are trained simultaneously with competing objectives. In the context of communication channels, GANs can be employed to model and mitigate channel noise effectively.

### 3.9.1 Generator (G)

The generator aims to create realistic channel noise, which, when added to the transmitted codeword, mimics the actual noise experienced in the communication channel. Formally, the generator takes the transmitted codeword $c$ and optionally structured noise $z$ to produce generated noise $z_G$:

$$z_G = G(c, z) \tag{17}$$

### 3.9.2 Discriminator (D)

The discriminator attempts to distinguish between the actual noisy codeword and the generator's output. It takes the received signal $r$, which is the transmitted codeword $c$ plus noise, and tries to estimate the transmitted codeword $\hat{c}$ and the noise $\hat{z}$:

$$\hat{c}, \hat{z} = D(r) \tag{18}$$

#### 3.9.3 Adversarial Training

During training, the generator and discriminator are updated iteratively. The generator aims to produce noise that fools the discriminator, while the discriminator improves its ability to correctly identify and decode the received signal. The loss functions for the generator and discriminator are typically based on binary cross-entropy:

$$\text{Loss}_D = -\mathbb{E}[\log D(r)] - \mathbb{E}[\log(1 - D(G(c, z)))] \tag{19}$$

$$\text{Loss}_G = \mathbb{E}[\log(1 - D(G(c, z)))] \tag{20}$$

#### 3.9.4 Bit Error Rate (BER) and Frame Error Rate (FER)

The performance of the GAN-based communication system is evaluated using the Bit Error Rate (BER) and Frame Error Rate (FER). BER is the ratio of erroneous bits to the total transmitted bits, while FER is the ratio of erroneous frames to the total transmitted frames.

$$\text{BER} = \frac{\text{Number of Erroneous Bits}}{\text{Total Number of Transmitted Bits}} \tag{21}$$

$$\text{FER} = \frac{\text{Number of Erroneous Frames}}{\text{Total Number of Transmitted Frames}} \tag{22}$$

This GAN-based approach leverages the adversarial nature of GANs to create robust models for communication channels, improving the accuracy of transmitted data decoding in noisy environments.

# 4 5G NR LDPC Codes

## 4.1 5G New Radio (NR)

5G New Radio (NR) is the global standard for a unified, more capable 5G wireless air interface. It is designed to support enhanced mobile broadband, ultra-reliable low-latency communication, and massive machine-type communication.

## 4.2 LDPC Codes in 5G NR

Low-Density Parity-Check (LDPC) codes are a key component of the 5G NR standard, chosen for their excellent error-correcting performance and efficiency. LDPC codes in 5G NR are defined by the 3rd Generation Partnership Project (3GPP) in the TS 38.212 standard.


Figure 1: Illustration of the GAN approach for modeling communication channels. The generator creates noise to simulate channel conditions, while the discriminator attempts to decode the received signal.

#### 4.2.1 Structure of LDPC Codes

The parity-check matrix $H$ of an LDPC code is sparse and typically represented as a Tanner graph consisting of variable nodes (VNs) and check nodes (CNs). The parity-check matrix $H$ for 5G NR LDPC codes has specific design features to support various block lengths and code rates efficiently.

#### 4.2.2 Encoding Process

The encoding process maps an input bit sequence $\mathbf{m} \in \{0,1\}^k$ to a codeword $\mathbf{c} \in \{0,1\}^n$ using a generator matrix $G$ such that:

$$\mathbf{c} = \mathbf{m}G \tag{23}$$

The codeword $\mathbf{c}$ satisfies the equation $H\mathbf{c}^T = 0$.

#### 4.2.3 Transmission

The codeword is modulated and transmitted over the AWGN channel:

$$\mathbf{y} = \mathbf{x}_s + \mathbf{z} \tag{24}$$

where $\mathbf{x}_s$ is the modulated signal and $\mathbf{z} \sim \mathcal{N}(0, \sigma^2 I)$ is the Gaussian noise.

#### 4.2.4 Decoding Process

The decoding process involves iterative algorithms, such as belief propagation or message passing, to estimate the transmitted codeword $\mathbf{c}$ from the received signal $\mathbf{y}$. The objective is to find the codeword $\hat{\mathbf{c}}$ that minimizes the syndrome:

$$\text{syndrome}(\mathbf{y}) = H\hat{\mathbf{c}}^T \mod 2 \tag{25}$$

### 4.3 Compliance with 5G NR Standard

5G NR LDPC codes are compliant with the 3GPP TS 38.212 standard, which specifies:

- Support for multiple block lengths and code rates.
- Structured parity-check matrices designed for efficient encoding and decoding.
- Compatibility with high-throughput and low-latency requirements.

## 5 Model Architecture

The architecture of the Transformer Diffusion model is designed to leverage the strengths of both transformer networks and denoising diffusion probabilistic models (DDPM) for the task of decoding low-density parity-check (LDPC) codes in 5G communication systems. The model consists of several key components, which are described in detail below.

### 5.1 Model Initialization

The Transformer Diffusion model is initialized with several parameters:

- **Model Type**: Specifies whether the model is generative or discriminative.
- **Number of Steps ($n_{\text{steps}}$)**: Defines the number of diffusion steps.
- **Parity-Check Matrix ($H$)**: A sparse matrix representing the LDPC code.
- **Embedding Dimension ($d_{\text{model}}$)**: The dimension of the embeddings.

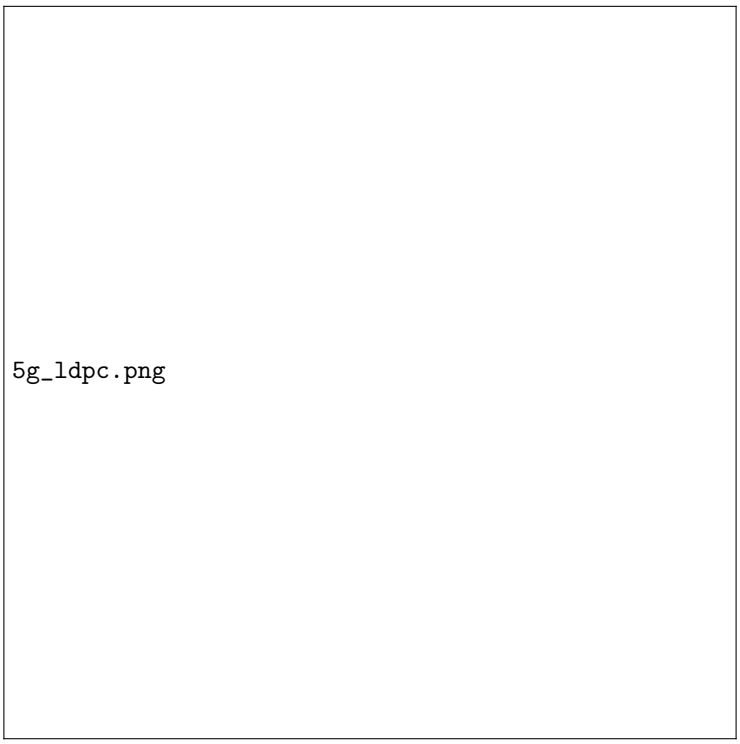


Figure 2: Illustration of the 5G NR LDPC coding and decoding process.

- **Number of Heads ($h$)**: The number of attention heads in the transformer.
- **Number of Transformer Layers ($t_{\mathbf{layers}}$)**: The number of layers in the transformer decoder.
- **Beta Schedule ($\beta_t$)**: Defines the noise schedule for the diffusion process.

## 5.2 Mask Creation

A mask is created based on the parity-check matrix $H$ to facilitate the attention mechanism in the transformer:

$$\text{mask}_{i,j} = \begin{cases} 1 & \text{if } H[i,j] = 1 \\ -\infty & \text{otherwise} \end{cases} \tag{26}$$

### 5.3 Embedding Layers

The source embedding layer (src_embed) and time embedding layer (time_embed) are initialized as trainable variables:

$$\text{src\_embed} \in \mathbb{R}^{1\times(n+m)\times d_{\text{model}}} \tag{27}$$

$$\text{time\_embed} \in \mathbb{R}^{n_{\text{steps}}\times d_{\text{model}}} \tag{28}$$

### 5.4 Transformer Decoder

The transformer decoder consists of $t_{\text{layers}}$ layers, each with multi-head attention and feed-forward neural networks. The attention mechanism is defined as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{29}$$

where $Q$, $K$, and $V$ are the query, key, and value matrices, respectively, and $d_k$ is the dimensionality of the keys.

### 5.5 Forward Diffusion Process

The forward diffusion process adds Gaussian noise to the codeword over $n_{\text{steps}}$ steps. At each step $t$, the noisy codeword $c_t$ is given by:

$$c_t = \sqrt{1-\beta_t}\cdot c_{t-1} + \sqrt{\beta_t}\cdot z \tag{30}$$

where $z \sim \mathcal{N}(0, I)$ is Gaussian noise and $\beta_t$ is the noise variance at step $t$.

### 5.6 Reverse Diffusion Process

The reverse diffusion process aims to recover the original codeword by iteratively denoising $c_t$ using the transformer decoder. The noise estimate $\hat{z}$ is predicted by the model:

$$\hat{z} = \text{Transformer}(c_t, t) \tag{31}$$

The updated codeword at step $t-1$ is given by:

$$c_{t-1} = \frac{1}{\sqrt{1-\beta_t}}\left(c_t - \beta_t\cdot\hat{z}\right) \tag{32}$$

### 5.7 Training Objective

The training objective is to minimize the discrepancy between the true noise $z$ and the predicted noise $\hat{z}$. This is typically done using a mean squared error (MSE) loss:

$$\mathcal{L}_{\text{MSE}} = \mathbb{E}\left[\|z-\hat{z}\|^2\right] \tag{33}$$

### 5.8 Line Search for Optimal Step Size

To refine the estimated noise, a line search is performed to find the optimal step size $\lambda$ that minimizes the syndrome:

$$\lambda = \arg\min_{\lambda} \sum_{i=1}^{m} (H[i] \cdot (c_t - \lambda \cdot \hat{z}) \mod 2) \tag{34}$$

### 5.9 Final Decoding Step

The final decoding step involves iterating through the reverse diffusion process until the syndrome is zero, indicating a valid codeword:

$$\text{syndrome}(c_t) = H \cdot c_t \mod 2 \tag{35}$$

The process stops when $\text{syndrome}(c_t) = 0$.

## 6 Optimizing

This section details the implementation and optimization of the Linear Transformer and the application of numerical splitting methods to enhance the efficiency and accuracy of the model.

### 6.1 Linear Transformer Attention Mechanism

The Linear Transformer model was designed to handle long sequences with a complexity of $O(n)$. The attention mechanism in the Linear Transformer is implemented as follows:

- The input tensor $x$ is first projected into query, key, and value tensors using learned weight matrices.

$$\text{query}, \text{key}, \text{value} = W_q x, W_k x, W_v x \tag{36}$$

- The key and value tensors are projected into a lower-dimensional space using projection matrices.

$$\text{key} = \text{key} \cdot W_k^{\text{proj}}, \quad \text{value} = \text{value} \cdot W_v^{\text{proj}} \tag{37}$$

- The query, key, and value tensors are reshaped to facilitate multi-head attention.

$$\text{query} = \text{reshape}(\text{query}), \quad \text{key} = \text{reshape}(\text{key}), \quad \text{value} = \text{reshape}(\text{value}) \tag{38}$$

- The attention scores are computed using the scaled dot-product of the query and key tensors.

$$\text{scores} = \frac{\text{query} \cdot \text{key}^T}{\sqrt{d_k}} \tag{39}$$

- A mask is applied to the scores to handle padding and other constraints.

$$\text{scores} += \text{mask} \cdot -\infty \tag{40}$$

- The attention weights are computed using the softmax function, and the output is obtained by applying these weights to the value tensor.

$$\text{attention} = \text{softmax}(\text{scores}) \cdot \text{value} \tag{41}$$

This efficient implementation of the attention mechanism allows the model to scale linearly with the sequence length, significantly improving its performance on long sequences.

## 6.2 Numerical Splitting Method for Enhanced Diffusion

To further enhance the performance, we incorporated a numerical splitting method, specifically the Strang splitting, into the reverse diffusion process. This method separates the update steps into condition and diffusion subproblems, which are handled independently for improved numerical stability and accuracy.

### 6.2.1 Lie-Trotter Splitting (LTSP)

The Lie-Trotter splitting method divides the reverse diffusion process into two steps:

1. **Condition Subproblem**: Update the state with respect to the condition term.

$$r_{t-\frac{1}{2}} = r_t - \eta \nabla_{r_t} \log p(c|r_t) \tag{42}$$

2. **Diffusion Subproblem**: Update the state by adding Gaussian noise.

$$r_{t-1} = r_{t-\frac{1}{2}} + \mathcal{N}(0, \sigma_t I) \tag{43}$$

### 6.2.2 Strang Splitting (STSP)

The Strang splitting method provides a higher-order splitting approach:

1. **Half-Step Condition Subproblem**:

$$r_{t-\frac{1}{2}} = r_t - \frac{\eta}{2} \nabla_{r_t} \log p(c|r_t) \tag{44}$$

2. **Full-Step Diffusion Subproblem**:

$$r_{t-1} = r_{t-\frac{1}{2}} + \mathcal{N}(0, \sigma_t I) \tag{45}$$

3. **Second Half-Step Condition Subproblem**:

$$r_{t-1} = r_{t-1} - \frac{\eta}{2} \nabla_{r_{t-1}} \log p(c|r_{t-1}) \tag{46}$$

By incorporating these splitting methods, we achieve a balance between computational efficiency and numerical accuracy, resulting in improved performance for the Transformer Diffusion model.

## 7 Experiments

## 8 Results

## 9 Discussion

## 10 Conclusion

## References

[1] A. Vaswani et al., "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[2] R. Child et al., "Generating long sequences with sparse transformers," in *arXiv preprint arXiv:1904.10509*, 2019.