



# Segurança em Redes 6G

## Um Estudo de Caso com Deep Learning na Camada Física

Fernando, Emanuel, Pedro W., Gustavo, Pedro J.

Introdução ao Projeto

# PLS-DNN: Segurança da Camada Física com Deep Learning

Este projeto explora o uso de Redes Neurais Profundas (DNNs) para garantir a segurança na camada física de comunicações sem fio, com foco em redes 6G.



Fundamentos Técnicos

# Framework PLS-DNN em PyTorch

Implementação robusta com PyTorch para simulações de segurança em cenários de comunicação desafiadores.



## Deep Learning

O coração do nosso modelo de segurança.



## PyTorch

Framework flexível para desenvolvimento.



## Camada Física

Onde a segurança se manifesta.



Made with GAMMA

# Configurações Globais: O Ambiente de Simulação

1

**MESSAGE\_LEN**

1000 bits/pacote

2

**BATCH\_SIZE**

64 pacotes/lote

3

**NUM\_EPOCHS**

500 (Treinamento aprofundado)

4

**LEARNING\_RATE**

0.002 (Otimizado para convergência)

5

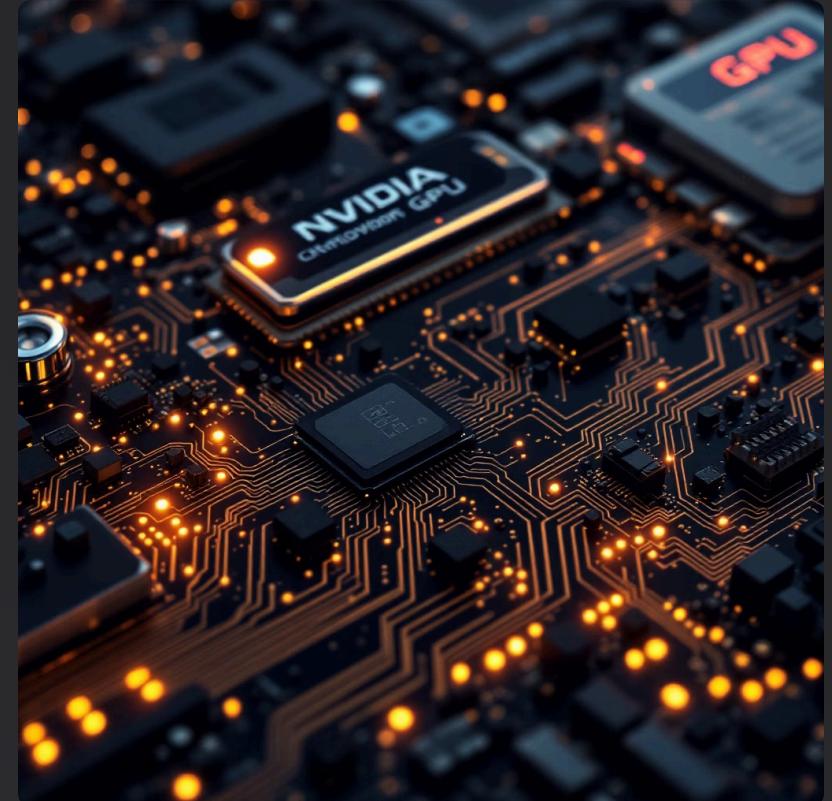
**SNR\_TRAIN\_DB**

10 dB (Cenário desafiador)

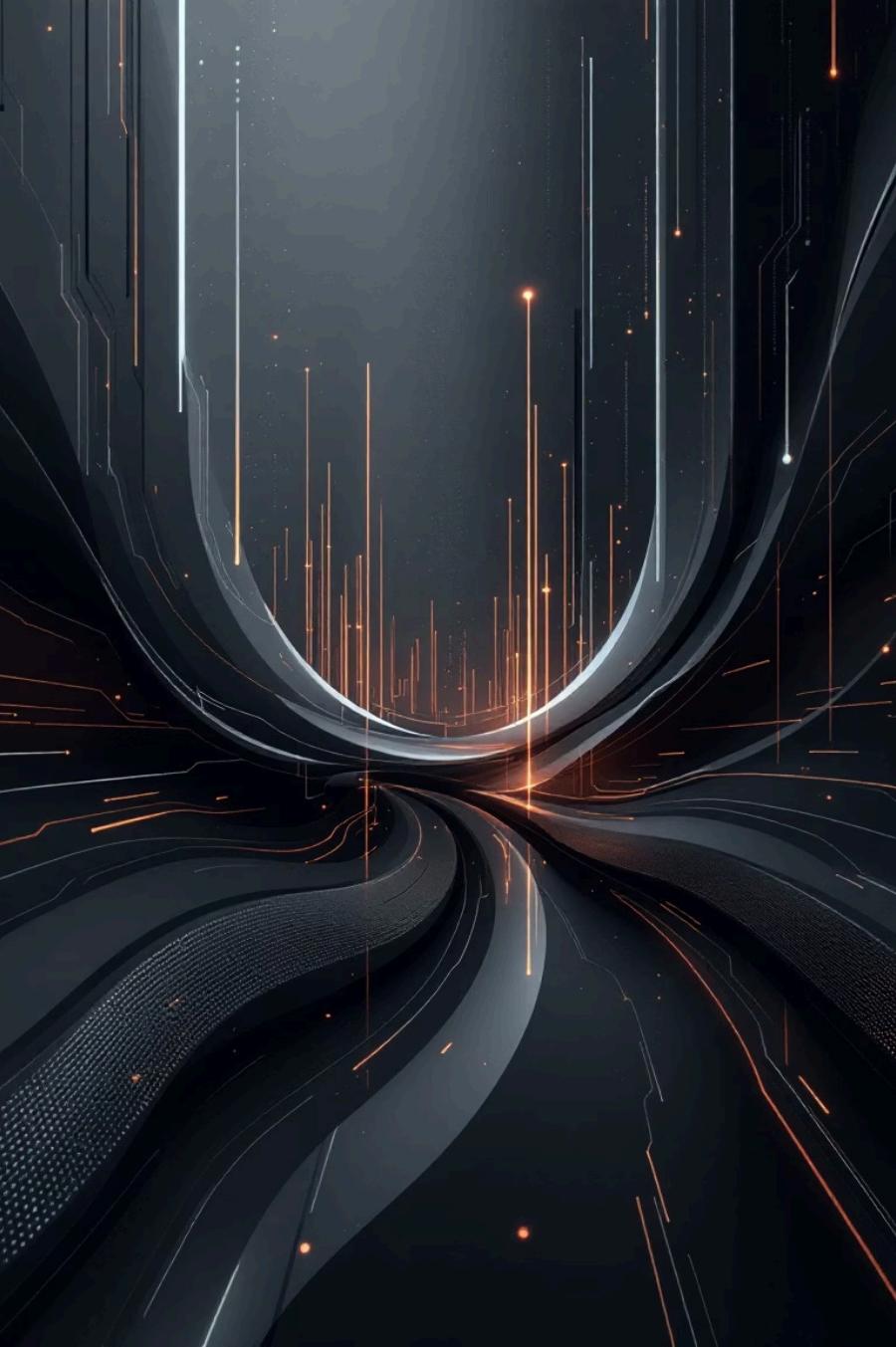
# Otimização de Hardware para Desempenho

A escolha do hardware é crucial para a eficiência do treinamento de modelos de Deep Learning. Detectamos automaticamente a presença de uma GPU NVIDIA para acelerar o processamento, garantindo simulações mais rápidas e complexas.

- **GPU NVIDIA:** Desempenho superior, ideal para cálculos paralelos de redes neurais.
- **CPU:** Opção alternativa quando a GPU não está disponível, com desempenho limitado para grandes modelos.

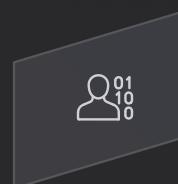


if torch.cuda.is\_available(): DEVICE = torch.device("cuda") else: DEVICE = torch.device("cpu")



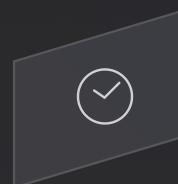
# Componentes do Sistema de Comunicação

A classe `CommunicationSystem` é responsável por simular as etapas fundamentais da transmissão de dados, desde a geração dos bits até a aplicação do ruído no canal.



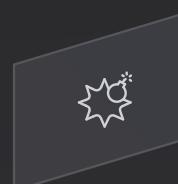
## Geração de Bits

Bits aleatórios (0 ou 1) simulando Alice.



## Modulação BPSK

Transforma bits em sinais (-1 ou 1).



## Canal AWGN

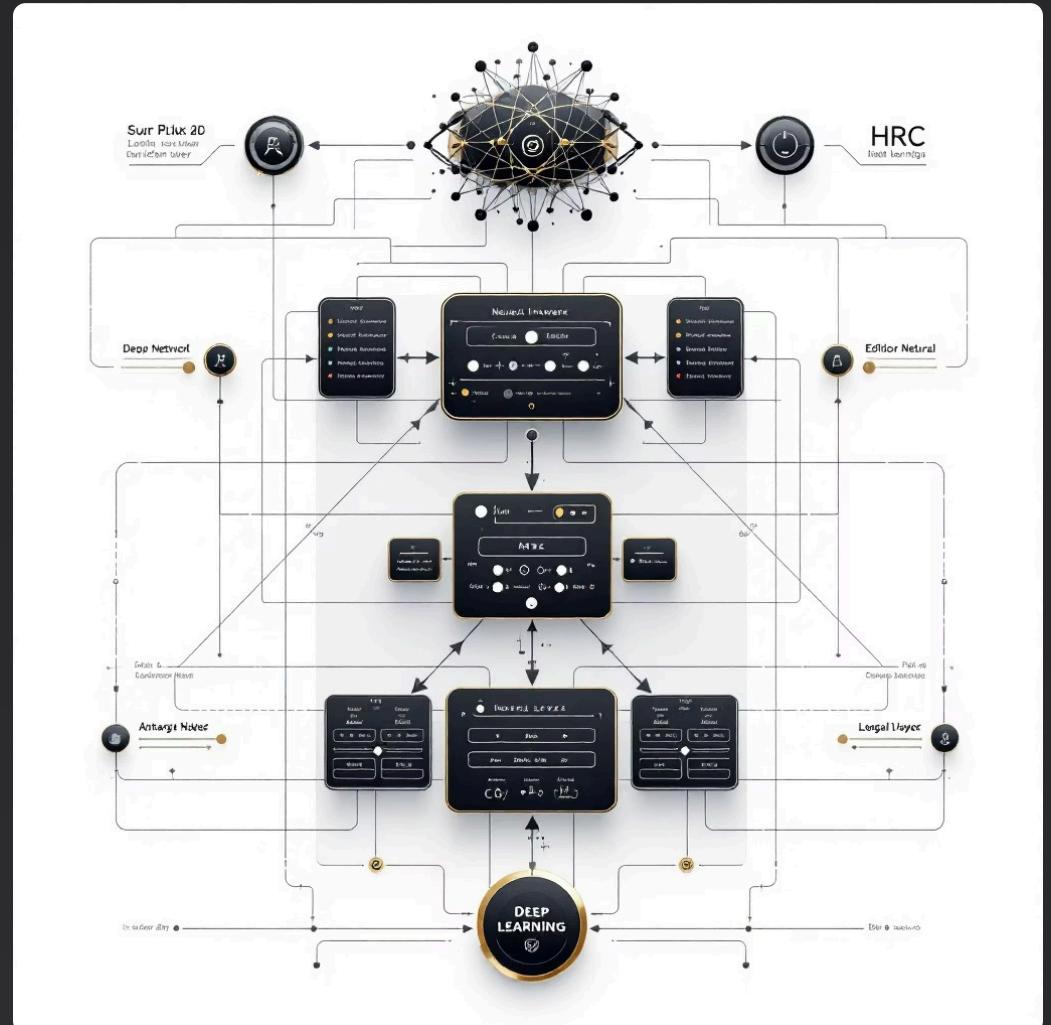
Adiciona Ruído Branco Gaussiano ao sinal.

# A Arquitetura PLS-DNN

Nossa Rede Neural Profunda (DNN) é projetada para decodificar os bits transmitidos, operando como o receptor "Bob" e aprendendo a mitigar o ruído do canal.

- **Camadas Densas:** Três camadas lineares para extração de características.
- **Ativação ReLU:** Introduz não-linearidade para aprender padrões complexos.
- **Ativação Sigmoid:** Produz probabilidades de bits entre 0 e 1.

```
 nn.Linear(input_size, 512)  
  
nn.ReLU()  
  
nn.Linear(256, input_size)  
  
nn.Sigmoid()
```



# Treinamento do Modelo: O Loop de Aprendizagem

O modelo PLS-DNN é treinado iterativamente para minimizar o erro de decodificação de bits, usando a função de perda de entropia binária cruzada e o otimizador Adam.



## Geração de Bits

Alice envia bits originais.



## Modulação e Canal

Sinais transmitidos e corrompidos por ruído.



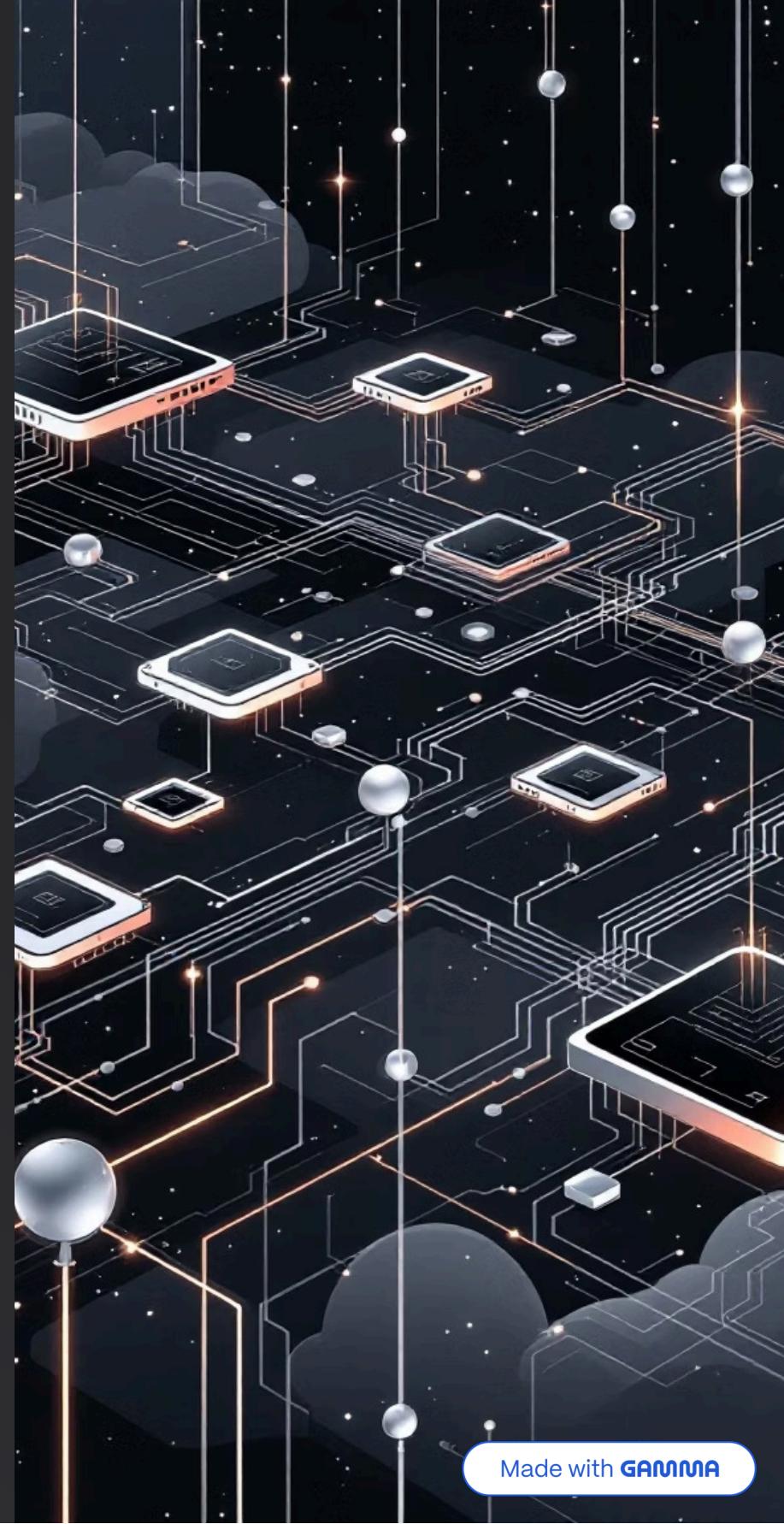
## Previsão DNN

Bob tenta decodificar os bits.



## Cálculo de Perda e Otimização

Ajuste de pesos para reduzir erros.

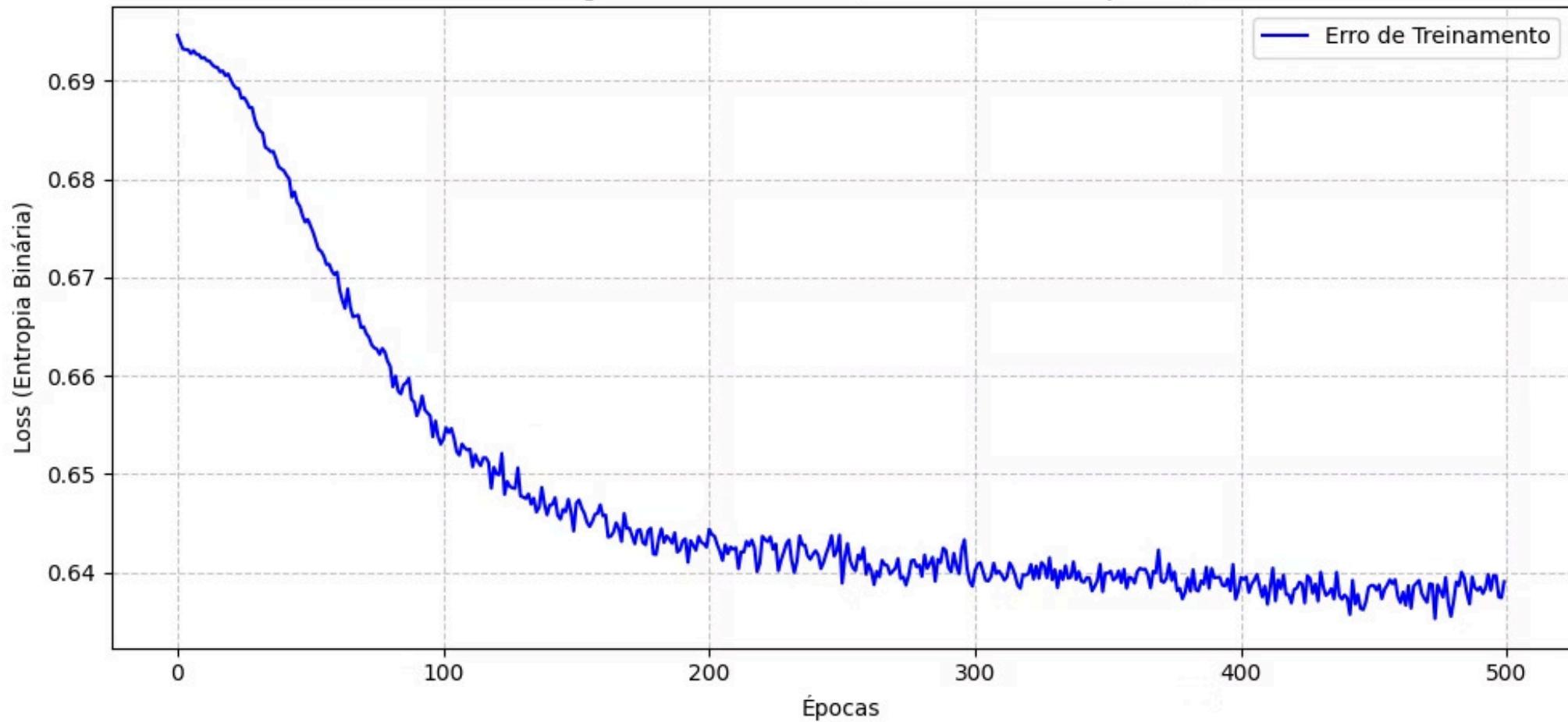


# Convergência do PLS-DNN

A perda de treinamento diminui ao longo das épocas, indicando que o modelo está aprendendo a decodificar os bits de forma eficaz, mesmo na presença de ruído.

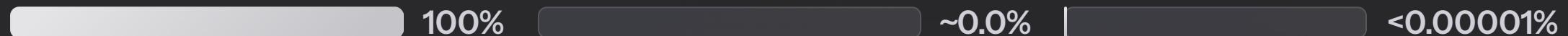
O gráfico de perda demonstra a eficácia do treinamento, com a curva decaindo consistentemente. Isso valida a capacidade do PLS-DNN de aprender a mapear sinais ruidosos para bits originais.

Convergência do PLS-DNN (SNR=10dB - 500 Épocas)



# Resultados Finais e Perspectivas

A validação do modelo revela uma baixa Taxa de Erro de Bits (BER), confirmando o potencial do PLS-DNN para garantir a segurança na camada física de futuras redes 6G.



## Total Bits

Avaliados na fase de teste.

## Bits Errados

Ínfimo número, demonstrando precisão.

## BER

Taxa de Erro de Bits extremamente baixa.

## Próximos Passos:

Explorar cenários com interferência e Eve (eavesdropper) para testar a robustez do sistema.