Detecção de Intrusão em Tempo Real para Mensagens GOOSE IEC 61850 via GRU Unidirecional Embarcado

Carlos E. Sousa[1], **Francisco J. A. Oliveira**[2], **Bilmar A. A. Ferreira**[3] **, Fábio L. L. Mendonça**[1] **, João J. C. Gondim**[1]

[1]Departamento de Engenharia Elétrica – Universidade de Brasília (UnB)
Campus Universitário Darcy Ribeiro - CEP 70.910-900 – Brasília – DF – Brasil

[2]Universidade Federal de Juiz de Fora
CEP 36036-900 – Juiz de Fora - MG – Brasil

[3]Deep Defense CEP 70714-900 – Brasília – DF – Brasil

sousa.carlos@redes.unb.br, jorge.engenharia@ymail.com

bilmar.angelis@deepdefense.com.br, fabio.mendonca@redes.unb.br, gondim@unb.br

Abstract. *Digital substations rely on IEC 61850 GOOSE messages demanding sub-3 ms real-time delivery. Existing IDS proposals focus on classification accuracy while ignoring deployment latency; resampling techniques used to address class imbalance corrupt the chronological integrity of GOOSE sequence fields. This paper proposes an edge- oriented IDS based on a unidirectional GRU-200 with a 3D sliding window, trained without resampling via full-sequence loss. Static graph unrolling enables C++ TensorFlow Lite conversion. On a public HIL dataset the model achieves 99.76 % accuracy and 91.93 % Macro F1; profiling on a Raspberry Pi 3 yields a mean inference of 2.77 ms and cross-scenario mean P95 of 2.999 ms (≤3 ms).*

Resumo. *Subestações digitais dependem de mensagens GOOSE da IEC 61850, as quais exigem entrega em tempo real abaixo de 3 ms. As propostas existentes de IDS (Sistemas de Detecção de Intrusão) focam na acurácia da classificação, ignorando a latência de implantação; técnicas de reamostragem utilizadas para tratar o desbalanceamento de classes corrompem a integridade cronológica dos campos de sequência do GOOSE. Este artigo propõe um IDS orientado à borda baseado em uma GRU-200 unidirecional com uma janela deslizante 3D, treinado sem reamostragem por meio de perda de sequência completa (full-sequence loss). O desenrolamento de grafo estático (static graph unrolling) permite a conversão para TensorFlow Lite em C++. Em um conjunto de dados público HIL (Hardware-in-the-Loop), o modelo alcança 99,76% de acurácia e 91,93% de Macro F1; o perfilamento (profiling) em um Raspberry Pi 3 resulta em uma inferência média de 2,77 ms e P95 médio entre cenários de 2,999 ms (≤3 ms).*

1. Introdução

A rápida modernização dos sistemas de potência elétrica em redes inteligentes (smart grids) aumentou significativamente a eficiência, a estabilidade e o controle descentralizado de infraestruturas críticas. O ponto central desse paradigma é a subestação digital,

que depende de Dispositivos Eletrônicos Inteligentes (IEDs) interconectados por meio de uma Rede de Comunicação de Subestação (SCN). Para alcançar uma interoperabilidade contínua entre equipamentos de múltiplos fabricantes, a Comissão Eletrotécnica Internacional introduziu a norma IEC 61850, que define os modelos de dados e os protocolos de comunicação para a automação de subestações [Abdelrahman et al. 2023].

Dentro dessa arquitetura, o protocolo GOOSE (Generic Object-Oriented Substation Event) é o mecanismo primário para a troca de mensagens de proteção e controle de tempo crítico, tais como comandos de trip de disjuntores e sinais de intertravamento [International Electrotechnical Commission 2003]. Para garantir a eficácia da proteção elétrica, a norma IEC 61850 estipula estritamente que as mensagens GOOSE devem ser geradas, transmitidas e processadas em menos de 3 ms [Rodríguez et al. 2021].

Consequentemente, o protocolo GOOSE é mapeado diretamente na camada de Enlace de Dados (Camada 2) para eliminar o overhead de protocolo. No entanto, esse requisito rigoroso de velocidade resultou em um protocolo desprovido de criptografia integrada ou de mecanismos robustos de autenticação [Lozano-Paredes et al. 2026]. Como resultado, as comunicações GOOSE são altamente vulneráveis a ciberataques.

Para mitigar essas vulnerabilidades, a implementação de Sistemas de Detecção de Intrusão (IDS) tornou-se uma prioridade na segurança de Tecnologia Operacional (OT). Embora a pesquisa recente tenha migrado cada vez mais para modelos de Aprendizado Profundo (Deep Learning - DL) para detectar anomalias complexas e ataques de dia zero, duas lacunas críticas permanecem na literatura. Primeiro, a maioria das propostas de IDS baseadas em dados depende de técnicas de balanceamento artificial de dados (como SMOTE ou Random Oversampling) para lidar com o desbalanceamento extremo de classes do tráfego industrial. Infelizmente, a aplicação de reamostragem a capturas de rede GOOSE inevitavelmente corrompe a sequência cronológica estrita dos campos de número de estado (stNum) e número de sequência (sqNum), destruindo a própria lógica temporal que as Redes Neurais Recorrentes (RNN) utilizam para detectar ataques furtivos. Segundo, os estudos existentes frequentemente avaliam seus algoritmos em ambientes interpretados de alto nível, como o Python. Embora esses modelos alcancem alta acurácia teórica, eles introduzem um massivo overhead que os impede de cumprir o orçamento de latência de 3 ms quando implantados em hardware de borda real, tornando-os impraticáveis para a implantação em subestações do mundo real [de Oliveira et al. 2025]

Para enfrentar esses desafios, este artigo propõe um Sistema de Detecção de Intrusão em tempo real e orientado à borda, sob medida para mensagens GOOSE da IEC 61850, baseado em uma arquitetura de Unidade Recorrente fechada (GRU) unidirecional. Ao preservar intencionalmente a natureza desbalanceada dos dados operacionais e otimizar o pipeline de inferência para dispositivos com restrição de recursos, este trabalho preenche a lacuna entre os modelos teóricos de Inteligência Artificial e a implantação prática em OT.

As principais contribuições deste artigo consistem: Na Preservação da Integridade Temporal: Demonstramos que evitar o balanceamento sintético de dados preserva a continuidade dos campos de sequência do GOOSE; Em uma Arquitetura Unidirecional Otimizada para a Borda: Propomos uma arquitetura GRU-200 unidirecional combinada com uma janela deslizante 3D, o quê elimina a necessidade de esperar por pacotes futuros; e na

Implantação em Borda em Tempo Real Abaixo de 3ms: Superamos o gargalo de latência das implantações tradicionais de IDS baseadas em Python executando o desenrolamento de grafo estático (unroll=True).

Este artigo está organizado da seguinte forma. A Seção 2 fornece a fundamentação teórica a respeito da norma IEC 61850, as vulnerabilidades do protocolo GOOSE e as RNNs aplicadas neste estudo. A Seção 3 revisa os trabalhos relacionados e identifica as lacunas de pesquisa atuais. A Seção 4 detalha o processo de engenharia de dados, a descrição do conjunto de dados e a configuração experimental. A Seção 5 apresenta a metodologia proposta, incluindo a arquitetura GRU unidirecional e a estratégia de implantação desenrolada na borda. A Seção 6 discute os resultados. Finalmente, a Seção 7 apresenta as conclusões e esboça as direções para pesquisas futuras.

2. Fundamentação Teórica

A transformação dos sistemas tradicionais de potência em redes inteligentes (smart grids) é impulsionada fortemente pela digitalização dos sistemas de automação de subestações. No cerne dessa modernização está a norma IEC 61850, que é um arcabouço globalmente reconhecido e amplamente aplicado na automação de subestações para garantir a interoperabilidade e a troca confiável de dados entre IEDs [International Electrotechnical Commission 2003, Dewangan et al. 2024]. Diferente dos sistemas legados que dependiam de um cabeamento físico de cobre complexo e dispendioso, a IEC 61850 introduz uma abordagem orientada a objetos sobre redes Ethernet, assegurando uma comunicação contínua entre equipamentos de múltiplos fabricantes. De acordo com a norma, uma subestação digital opera em três camadas hierárquicas: a camada de processo (contendo sensores físicos, atuadores e unidades de mesclagem), a camada de vão (bay layer, que abriga IEDs de proteção e controle) e a camada de estação (composta por interfaces homem-máquina e gateways SCADA). Para facilitar a comunicação entre esses níveis, a norma especifica diferentes protocolos sob medida para requisitos de latência específicos, tais como o Manufacturing Message Specification (MMS) para relatórios ao nível de estação, e protocolos de tempo crítico, como o Sampled Values (SV) e o GOOSE, para operações ao nível das camadas de processo e de Vão [Tobar-Rosero et al. 2024].

O protocolo GOOSE permite a troca de eventos críticos de subestação, notificações de mudança de estado, alarmes e comandos de controle, incluindo estados de disjuntores e intertravamentos de chaves seccionadoras [Jay 2023]. Para cumprir o rigoroso requisito de latência de 3 ms para operações de proteção, as mensagens GOOSE ignoram as camadas de rede e de transporte (ou seja, a pilha TCP/IP) e são mapeadas diretamente na camada de Enlace de Dados, ou Camada 2 do modelo Ethernet [Rodríguez et al. 2021]. Embora esse design arquitetônico garanta uma comunicação ultrarrápida, ele sacrifica os mecanismos integrados de segurança. A comunicação GOOSE nativa carece de criptografia e autenticação, transmitindo dados em texto claro por meio de um modelo multicast do tipo publicador-assinante (publisher-subscriber) [Lozano-Paredes et al. 2026]. Consequentemente, qualquer dispositivo com acesso à rede local da subestação pode interceptar, ler e forjar mensagens sem ser ativamente bloqueado pelo protocolo.

Como o GOOSE opera sobre uma rede multicast não criptografada de Ca-

mada 2, atores maliciosos podem explorar campos específicos da Unidade de Dados do Protocolo de Aplicação (APDU), particularmente o número de estado (stNum) e o número de sequência (sqNum), para lançar ataques ciberfísicos graves [Fernando et al. 2025]. O modelo de ameaças neste estudo abrange quatro vetores de ataque principais [Quincozes et al. 2024]: **Injeção de Mensagem (Message Injection)**: O atacante fabrica e injeta quadros GOOSE falsos na rede, com a intenção de alterar valores de controle booleanos (por exemplo, forçar a abertura de um disjuntor) sem seguir a sequência cronológica estrita do protocolo. **Ataque de Replay (Replay Attack)**: Um adversário captura uma mensagem GOOSE legítima, previamente transmitida, e a retransmite na rede em um momento posterior. Isso visa acionar comandos desatualizados ou inválidos, perturbando o estado operacional atual do sistema elétrico. **Ataque de Mascaramento (Masquerade Attack)**: Uma variação altamente sofisticada e furtiva da injeção de mensagens. O atacante monitora ativamente a rede para aprender o estado atual dos contadores da APDU. Em seguida, injeta mensagens forjadas que imitam perfeitamente os incrementos esperados de stNum e sqNum e as rajadas temporais (timing bursts). Como o formato e a sequência estão matematicamente corretos, os ataques de mascaramento são excepcionalmente difíceis de detectar utilizando sistemas tradicionais baseados em assinaturas. **Envenenamento por Negação de Serviço** (Poisoning / Denial of Service): O atacante inunda o canal de comunicação com mensagens GOOSE maliciosas. Uma técnica comum envolve a injeção de pacotes com um stNum artificialmente alto. Isso força os IEDs receptores a atualizarem seus estados internos para o número alto malicioso, fazendo com que descartem subsequentemente todas as mensagens legítimas que contenham números de estado inferiores e corretos.

Modelos tradicionais de Aprendizado de Máquina (ML) são insuficientes para detectar ataques cronologicamente complexos por avaliarem cada pacote isoladamente. As RNNs, por meio de estados de memória interna, processam dados sequenciais e identificam anomalias no fluxo de stNum e sqNum [de Oliveira et al. 2025]. Entre as arquiteturas de RNN, a GRU destaca-se por utilizar portas de atualização e reinicialização para capturar dependências de longo prazo com menor custo tensorial que redes LSTM, tornando-a especialmente adequada para ambientes de borda com restrição de processamento e janela de inferência de 3 ms.

3. Trabalhos Correlatos

A vulnerabilidade do protocolo GOOSE tem impulsionado extensas pesquisas em IDS baseados em dados. A literatura recente explorou inicialmente algoritmos tradicionais de ML. Por exemplo, [Ustun et al. 2021] avaliaram vários algoritmos, incluindo Máquinas de Vetores de Suporte (SVM) e Árvores de Decisão, demonstrando uma alta acurácia de classificação para ataques de falsificação (spoofing) de GOOSE. Da mesma forma, [Nhung-Nguyen et al. 2024] utilizaram uma combinação de métodos supervisionados para classificar comportamentos normais e anômalos no tráfego GOOSE. No entanto, esses algoritmos tradicionais avaliam os quadros de rede como eventos isolados, ignorando a natureza sequencial e orientada a eventos do tráfego de subestações.

Reconhecendo essa limitação, os estudos recentes migraram para arquiteturas de DL capazes de realizar análises temporais. [Yang et al. 2022] propuseram um arcabouço de Autoencoder que aproveita redes Long Short-Term Memory (LSTM) e Gated Recurrent Unit (GRU) para reconstruir o tráfego legítimo e detectar anomalias com base nos

erros de reconstrução. [Wang et al. 2022] utilizaram LSTM Bidirecional com janelas deslizantes para identificar ataques internos (insider attacks) furtivos.

Além disso, [de Oliveira et al. 2025] provaram que arquiteturas recorrentes (como a GRU) superam as árvores de decisão tradicionais na detecção de ataques complexos de mascaramento (Masquerade), enquanto [Fernando et al. 2025] combinaram com sucesso Redes Neurais Profundas e LSTM para capturar dependências temporais em cenários de spoofing de GOOSE.

Apesar das altas acurácias relatadas na literatura, persiste uma lacuna metodológica crítica em relação ao desbalanceamento extremo de classes inerente ao tráfego de OT, onde os pacotes maliciosos representam uma fração minúscula dos dados. Para mitigar isso, muitos estudos empregam técnicas de balanceamento de dados ou aumento artificial de tráfego para lidar com a escassez de amostras de ataque [Bhattacharya et al. 2024].

Embora essas técnicas sejam altamente eficazes para conjuntos de dados estáticos e não sequenciais, aplicá-las ao tráfego de séries temporais industriais introduz uma falha grave: ela corrompe a integridade cronológica do protocolo. O protocolo GOOSE depende da progressão estrita e contínua dos contadores de Número de Estado (stNum) e Número de Sequência (sqNum), bem como de tempos de interchegada determinísticos [Quincozes et al. 2024]. Injetar amostras sintéticas ou duplicar aleatoriamente pacotes da classe minoritária quebra as transições temporais exatas que as RNNs são projetadas para aprender. Um IDS robusto deve, portanto, ser capaz de aprender a partir de dados sequenciais brutos e altamente desbalanceados, sem depender de corrupção temporal.

A barreira mais significativa para a implantação prática de modelos de DL em subestações digitais é o orçamento estrito de latência. A norma IEC 61850 dita que as mensagens de proteção GOOSE devem ser processadas em menos de 3 ms [Rodríguez et al. 2021]. A maioria dos modelos de DL propostos é avaliada em ambientes computacionais idealizados e de alto nível (por exemplo, servidores em nuvem executando Python), o que oculta o massivo *software overhead* necessário para operações de tensores e parsing de pacotes.

Nesse contexto, [Ustun et al. 2021] conduziram uma das poucas avaliações empíricas em hardware ao implantar algoritmos de ML em um dispositivo de borda com restrição de recursos (Raspberry Pi 3). Seus resultados revelaram que, embora Árvores de Decisão simples pudessem processar pacotes em menos de 1 milissegundo, algoritmos mais complexos como Random Forest, K-NN e SVM excederam drasticamente o tempo de processamento, tornando-os impraticáveis para comunicações GOOSE em tempo real. Isso expõe um desafio crítico não resolvido: implantar modelos sequenciais de DL altamente acurados (como a GRU) em hardware de borda (*Edge hardware*), cumprindo estritamente o requisito de latência inferior a 3 ms.

Atualmente, a literatura carece de um arcabouço que preencha a lacuna entre a acurácia das RNNs avançadas e a otimização de software de baixo nível (por exemplo, inferência desenrolada em C++ — *C++ unrolled inference*) necessária para sistemas embarcados de subestações no mundo real.

4. Engenharia de Dados e Configuração Experimental

A validação empírica do IDS proposto baseia-se em um conjunto de dados de tráfego de rede publicamente disponível hospedado na plataforma Kaggle [Oliveira et al. 2025]. Diferente de capturas puramente sintéticas, esse conjunto de dados foi construído utilizando um Simulador Digital em Tempo Real (RTDS) integrado com IEDs físicos em uma arquitetura Hardware-in-the-Loop (HIL) [de Oliveira et al. 2025]. Essa configuração rigorosa garante que a comunicação IEC 61850 capturada represente fielmente a dinâmica física, elétrica e temporal de um sistema real de proteção de linhas de transmissão de alta tensão. O conjunto de dados contém capturas de rede que refletem tanto operações benignas, incluindo estados estáveis e faltas elétricas naturais, quanto atividades maliciosas graves.

Uma característica definidora desse conjunto de dados é a sua distribuição de classes realista e altamente desbalanceada, a qual espelha com precisão os ambientes autênticos de OT. De acordo com a partição dos dados experimentais, o tráfego normal constitui a vasta maioria (aproximadamente 92,4 porcento) das amostras. Os ataques de Poisoning de alta taxa representam 6,6 porcento, enquanto ameaças furtivas e altamente complexas, como Replay, Injeção de Mensagem e Mascaramento, representam apenas 0,3 porcento cada. A utilização desses dados gerados por HIL e severamente desbalanceados é crucial para validar se a arquitetura recorrente proposta pode detectar com precisão anomalias cronológicas raras sem corromper a linha do tempo original dos pacotes.

Para alimentar a rede neural, em vez de processar diretamente os arquivos brutos de captura de rede, foram utilizados como entrada primária arquivos CSV tabulares pré-processados contendo as semânticas da camada de aplicação e a dinâmica de transmissão extraídas. Um total de 16 atributos (features) numéricos foi selecionado e projetado. Estes incluem campos essenciais do protocolo GOOSE, tais como stNum (Status Number), sqNum (Sequence Number), timeAllowedtoLive e goose.length. Para aprimorar a capacidade do modelo de detectar anomalias furtivas, atributos derivados que representam as diferenças matemáticas entre pacotes consecutivos já foram computados no conjunto de dados, tais como st_dif, sq_dif e timestamp_dif. Por fim, todos os atributos extraídos foram normalizados utilizando um MinMaxScaler para garantir a estabilidade numérica durante o treinamento da rede neural.

Dada a natureza sequencial das comunicações GOOSE, um algoritmo de Janela Deslizante 3D (3D Sliding Window) foi aplicado para mapear as dependências temporais dos pacotes de rede. Por meio de ajuste empírico, um tamanho de janela igual a 4 (WINDOW_SIZE = 4) foi selecionado, transformando os dados tabulares bidimensionais em tensores 3D com o formato (amostras, passos_temporais, atributos). Assim, a estrutura de entrada final fornecida ao modelo tornou-se (30216, 4, 16).

Uma decisão metodológica crítica neste estudo foi a preservação intencional da distribuição de classes original e altamente desbalanceada. Em redes OT industriais, o tráfego normal supera drasticamente os pacotes maliciosos. Embora os pipelines tradicionais de ML frequentemente apliquem técnicas de reamostragem para mitigar o ”Paradoxo da Acurácia”, a aplicação desses métodos a dados de rede em séries temporais corrompe a sequência cronológica exata dos pacotes. Modificar a distância temporal ou sintetizar pacotes artificiais quebra a continuidade estrita dos campos stNum e sqNum, que é a exata anomalia que a RNN visa detectar.

Em vez de manipular os dados, o desbalanceamento de classes foi tratado ao nível arquitetônico. A rede GRU foi configurada com o parâmetro return_sequences=True, permitindo que o modelo calcule a perda e atualize o gradiente em cada passo temporal individual da janela, e não apenas ao final. Isso forneceu um sinal de gradiente mais denso, capacitando a rede a aprender os padrões das classes de ataque minoritárias sem a necessidade de intervenção com dados sintéticos.

Para avaliar de forma abrangente o custo computacional e a latência do IDS proposto, a configuração experimental incorporou múltiplos ambientes de hardware e software. Inicialmente, uma máquina Intel Core i7 foi utilizada para a leitura das capturas de pacotes brutos e para estabelecer uma linha de base de desempenho para o *software overhead* em um ambiente interpretado de alto nível usando Python e Keras.

Para simular fielmente uma implantação real de computação de borda (*Edge computing*) em uma subestação digital, a arquitetura física foi expandida para incluir um sistema baseado em Ubuntu e um dispositivo embarcado com restrição de recursos. Especificamente, o tráfego de rede GOOSE capturado foi retransmitido sobre uma conexão Ethernet física utilizando a ferramenta tcpreplay executada na máquina Ubuntu. Esses pacotes foram transmitidos diretamente para um Raspberry Pi 3 (arquitetura ARM64), que atuou como o nó de borda alvo responsável por executar o pipeline de detecção de intrusão. Essa configuração de hardware heterogênea permitiu uma comparação direta entre o pesado tempo de parsing e inferência de um ambiente Python padrão contra a implantação altamente otimizada em C++ e TensorFlow Lite no Raspberry Pi, verificando, em última análise, a conformidade do sistema com o estrito orçamento de latência de 3 ms mandatório pela norma IEC 61850.

5. Metodologia e Arquitetura Proposta

Para identificar eficazmente as ciberameaças dentro da sequência temporal de mensagens da IEC 61850, este estudo propõe uma arquitetura de DL baseada em RNNs. Especificamente, a GRU foi selecionada em detrimento das redes LSTM devido à sua arquitetura simplificada, a qual exige menos operações de tensores e menor pegada de memória, características cruciais para implantações em computação de borda.

Ao contrário dos modelos bidirecionais (BiGRU), que processam os dados tanto na direção temporal progressiva quanto na regressiva, esta pesquisa utiliza deliberadamente uma GRU unidirecional. No contexto da detecção de intrusão em tempo real para subestações digitais, esperar por pacotes futuros para avaliar uma sequência retrógrada introduz uma latência inaceitável, a qual viola o orçamento operacional estrito de 3 ms.

A rede neural proposta recebe tensores 3D com o formato (4, 16), representando a janela deslizante de 4 passos temporais (timesteps) e 16 atributos. A arquitetura consiste em uma camada GRU unidirecional com 200 unidades ocultas, seguida por uma camada de Dropout com uma taxa de 0,2 para evitar o sobreajuste (overfitting) durante o treinamento. A camada recorrente é subsequentemente conectada a uma camada Densa (Dense) totalmente conectada com 200 unidades utilizando a função de ativação ReLU e, finalmente, a uma camada Densa de saída com 5 unidades utilizando a função de ativação Softmax para realizar a classificação multiclasse (Normal, Replay, Injeção de Mensagem, Mascaramento e Envenenamento).

Conforme estabelecido na Seção 4, este estudo absteve-se de utilizar a geração de

dados sintéticos ou a reamostragem temporal para manter a integridade cronológica estrita dos campos stNum e sqNum. Para capacitar a rede neural a detectar classes minoritárias (como os ataques de Replay e Mascaramento, que representam apenas 0,3% dos dados de treinamento), uma adaptação estrutural foi implementada no processo de aprendizado.

A camada GRU foi configurada com o parâmetro return_sequences=True. Em tarefas padrão de classificação com RNN, a rede processa a sequência inteira e emite uma única predição no passo temporal final. Ao definir esse parâmetro, a GRU é forçada a emitir uma predição (e, portanto, calcular a perda) em cada passo temporal individual da janela. Durante o treinamento, o rótulo de verdade fundamental (ground truth) é replicado ao longo de todos os 4 passos temporais da sequência. Consequentemente, a rede recebe quatro vezes mais sinal de gradiente por amostra. Essa retropropagação densa de gradiente força o otimizador a penalizar severamente as classificações incorretas em cada etapa da sequência, permitindo que o modelo aprenda de forma eficaz os padrões intrincados de ataques furtivos extremamente raros, sem a necessidade de balanceamento artificial de dados.

O modelo foi treinado utilizando o otimizador Adam (taxa de aprendizado = 0.001) e a função de perda de entropia cruzada categórica esparsa (sparse categorical cross-entropy) ao longo de 50 épocas, empregando uma função de retorno (callback) de parada antecipada (early stopping) para garantir uma generalização ideal.

Uma grande lacuna na literatura atual sobre Inteligência Artificial em Redes Inteligentes é a suposição de que modelos altamente acurados treinados em ambientes de alto nível (como Python/Keras) serão naturalmente transferidos para dispositivos de campo. Na realidade, os arcabouços de DL introduzem um *software overhead* substancial que os torna incompatíveis com as restrições de tempo real da IEC 61850. Para preencher essa lacuna, a metodologia proposta inclui um caminho direto para a implantação do modelo em hardware embarcado restrito utilizando C++.

Para alcançar isso, a camada GRU foi instanciada com o parâmetro unroll=True. Em implementações convencionais de RNN, os laços sequenciais são tratados dinamicamente (por exemplo, utilizando operações TensorList), o que introduz uma dependência de operações pesadas do TensorFlow (SELECT_TF_OPS ou Flex Delegates) que são mal suportadas ou altamente ineficientes em dispositivos de borda baseados em ARM, como o Raspberry Pi.

Ao ativar a flag unroll=True, o laço recorrente de 4 passos temporais é desenrolado estaticamente durante a fase de compilação do grafo, transformando as iterações temporais em operações explícitas e sequenciais de células GRU. Essa decisão arquitetônica elimina a necessidade de avaliação dinâmica de laços, permitindo que o modelo treinado final (.keras) seja convertido com sucesso para um formato TensorFlow Lite (.tflite) altamente otimizado, dependendo estritamente de componentes nativos TFLITE_BUILTINS. Esse modelo convertido pode ser carregado perfeitamente em um pipeline de execução em C++ nos nós de borda, contornando completamente o gargalo do interpretador Python e reduzindo drasticamente a latência de inferência ponta a ponta.

6. Resultados e Discussões

O modelo final foi avaliado no conjunto de teste original desbalanceado (30.216 amostras). Como o modelo emite um vetor de probabilidade por passo temporal, a predição é

extraída do último passo temporal de cada janela. A Tabela 1 resume os resultados por classe. O modelo alcança uma acurácia geral de **99,76,%** e um F1-score macro-médio (macro-averaged) de **91,93,%**. O tráfego Normal e os ataques de Poisoning são detectados com um F1 quase perfeito (99,89,% e 99,85,%, respectivamente). A Injeção de Mensagem (Message Injection) atinge 98,48,% de F1 e o Replay 86,12,%. O Masquerade apresenta o menor F1 (75,31,%), uma limitação estrutural discutida em seguida.

Tabela 1. Relatório de classificação por classe no conjunto de teste desbalanceado ($n = 30{,}216$). Modelo final: GRU-200, $W = 4$, unroll=True, seed 21.

Classe	Precisão (%)	Recall (%)	F1 (%)	Support
Normal	99.84	99.94	99.89	27,916
Replay	82.57	90.00	86.12	100
Msg. Inj	100.00	97.00	98.48	100
Masquerade	98.39	61.00	75.31	100
Poisoning	99.90	99.80	99.85	2,000
Accuracy		99.76		30,216
Macro avg	96.14	89.55	91.93	30,216
Weighted avg	99.76	99.76	99.75	30,216

A Figura 1 contrasta o F1-score de cada classe de ataque com o resultado relatado por [de Oliveira et al. 2025] para o mesmo conjunto de dados. A lacuna é mais pronunciada para o Masquerade (75,31,% vs. 96,37,%) e Replay (86,12,% vs. 99,01,%), enquanto a Injeção de Mensagem (98,48,%) e o Poisoning (99,85,%) são essencialmente equivalentes.

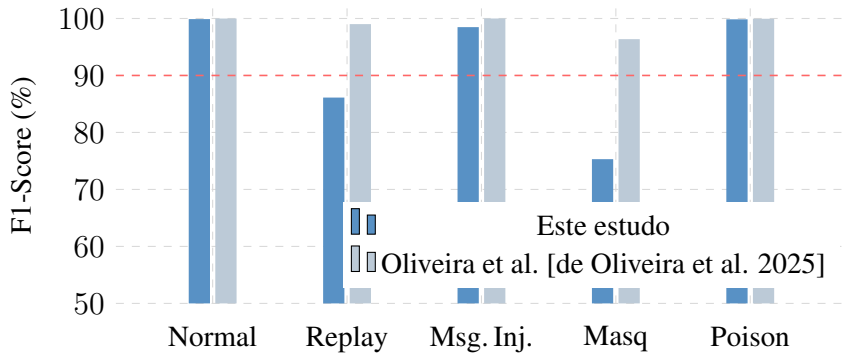

Figura 1. Comparação do F1-score por classe entre este estudo e [de Oliveira et al. 2025] no mesmo conjunto de dados. A linha tracejada indica a referência de F1 de 90 %.

A Tabela 2 apresenta a matriz de confusão no conjunto de teste desbalanceado. A principal fonte de classificação incorreta é o Masquerade: 39 de 100 amostras são preditas como Normal (21) ou Replay (18), devido ao padrão diferencial de stNum/sqNum quase idêntico gerado pela rajada de ataque de 4 pacotes dentro da janela de 4 pacotes. Todas as outras classes mostram entradas fora da diagonal principal desprezíveis.

O tamanho de janela $W = 8$ foi avaliado como uma alternativa para $W = 4$. Os resultados confirmaram a escolha ideal: o F1 do Replay caiu de 86,12,% para 23,00,%, o da Injeção de Mensagem de 98,48,% para 69,28,%, e o do Masquerade de 75,31,% para

Tabela 2. Matriz de confusão no conjunto de teste desbalanceado. Linhas: classe real; colunas: classe predita.

	Normal	Replay	Msg. Inj.	Masq.	Poison.
Normal	27,899	15	0	0	2
Replay	10	90	0	0	0
Msg. Inj.	3	0	97	0	0
Masquerade	21	18	0	61	0
Poisoning	4	0	0	0	1,996

Tabela 3. Latência de inferência ponta a ponta entre configurações de implantação.

Plataforma	Runtime	Média	Mediana	P95	P99
			(ms)		
x86 Windows	Python + Keras	17.0	16.3	21.3	25.0
RPi3 ARM64	Python + Keras	463	479	495	505
RPi3 ARM64	Python + TFLite	2.86	2.84	3.18	5.06
RPi3 ARM64	C++ + TFLite	6.02	5.98	7.01	8.08
RPi3 ARM64	C++ + TFLite + XNNPACK	2.997	2.902	3.160	5.721
RPi3 ARM64	C++ + XNNPACK + SCHED_FIFO	2.781	2.761	2.976*	3.235
IEC 61850 req.	—		—	≤ 3.0 ms	

* Requisito atendido no nível P95.

11,21,%, resultando em um F1 macro de 63,16,% - uma redução de 28,8 pontos percentuais. Essa degradação é consistente com a análise de [de Oliveira et al. 2025], onde o F1 se estabiliza em $W = 3$–4 e degrada para janelas maiores. A intuição é que estender a janela de contexto além do comprimento da rajada do ataque dilui os atributos diferenciais anômalos com o tráfego normal precedente, suavizando a fronteira de decisão.

A latência de inferência ponta a ponta (end-to-end — E2E) foi medida em seis configurações de implantação utilizando 500 pacotes GOOSE (497 janelas válidas) extraídos de capturas reais .pcapng retransmitidas via tcpreplay no Ubuntu conectado via Ethernet ao Raspberry Pi 3 (ARM64, 1,GB RAM). Tanto o tempo de inferência puro quanto o tempo completo do pipeline ponta a ponta (E2E) foram registrados com std::chrono::high_resolution_clock (C++) ou time.perf_counter_ns (Python), após uma etapa de aquecimento (warm-up) para estabilizar a compilação JIT. A Tabela 3 resume a progressão das seis configurações utilizando o cenário de Poisoning como uma linha de base representativa. A Figura 2 ilustra a faixa completa de latência em todas as configurações em uma escala logarítmica, tornando visualmente aparente a lacuna de três ordens de magnitude entre a linha de base Python,+,Keras e o pipeline otimizado em C++. A validação entre cenários (cross-scenario validation) sob a configuração ideal é apresentada na Tabela 4.

Para verificar que os resultados de latência não são específicos do padrão de tráfego de Poisoning, a configuração ideal (C++ + XNNPACK + SCHED_FIFO) foi aplicada a todos os quatro cenários de ataque. A Tabela 4 apresenta os resultados.

Os resultados confirmam que a latência de inferência é uma propriedade do modelo e do pipeline, sendo independente do cenário de ataque. As latências média e mediana variam menos de 0,07,ms entre os quatro cenários (2,759–2,781,ms e 2,761–2,828,ms, respectivamente).

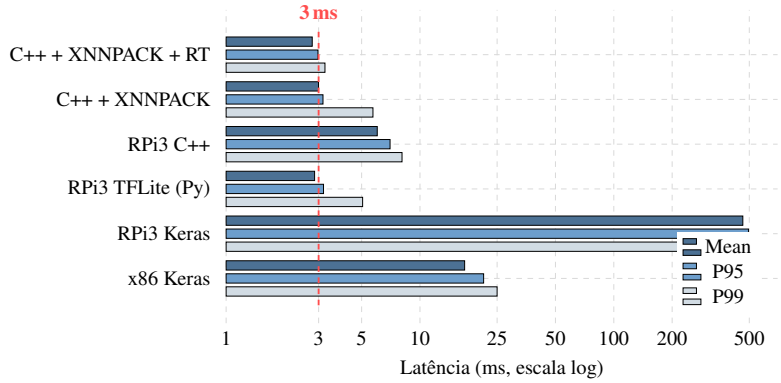

Figura 2. Latência de inferência entre configurações de implantação (eixo x em escala logarítmica). A linha vermelha tracejada indica o requisito de 3 ms do IEC 61850 GOOSE.

Tabela 4. Latência entre cenários sob a configuração ótima ($n = 497$ janelas por cenário).

Cenário	Média	Mediana	P95	P99
		(ms)		
Masquerade	2.763	2.797	2.998†	3.064
Msg. Injection	2.759	2.813	3.017	3.070
Poisoning	2.781	2.761	2.976†	3.235
Replay	2.772	2.828	3.007	3.099
Média entre cenários.	2.769	2.800	2.999	3.117
IEC 61850 req.	—		≤ 3.0 ms	

† Requisito atendido no nível P95.

Observou-se que o PyShark introduz 44,9,ms de overhead de pipeline no RPi3, em comparação com 0,01,ms para a libpcap no pipeline em C++ — uma redução por um fator de aproximadamente 4490. Esse resultado demonstra que o gargalo na implantação em Python não é a inferência da rede neural, mas sim a camada de parsing de pacotes.

A limitação estrutural na detecção do ataque de Masquerade (F1=75,31,%) decorre do próprio design do ataque: ele injeta exatamente quatro quadros GOOSE simulando um novo evento elétrico, o que corresponde precisamente ao tamanho de janela $W = 4$ adotado neste estudo. Dentro de uma única janela, o modelo observa uma rajada de mensagens com stNum incrementado e sqNum reinicializado, o que é indistinguível de um evento de mudança de estado GOOSE legítimo quando visualizado isoladamente. O modelo classifica corretamente 61,% das instâncias de Masquerade por meio de sutis diferenças em TimeAllowedToLive e nos atributos diferenciais, mas a ambiguidade inerente impõe um teto para essa métrica, independentemente da arquitetura ou da estratégia de treinamento. Aumentar o tamanho da janela para $W = 8$ expõe mais contexto temporal, mas causa uma regressão severa em todas as outras classes, confirmando que $W = 4$ é a escolha ideal no sentido de Pareto.

O estudo de referência atinge um F1 ponderado de 99,97,%, superando os 99,75,% relatados aqui. A lacuna está totalmente concentrada nas classes Masquerade e Replay. Oliveira et al. realizam o treinamento sem balanceamento explícito de dados, de forma análoga a este trabalho que também treina sem balanceamento, mas utiliza re-

$\mathrm{turn\_sequences{=}True}$ com perda de sequência completa (full-sequence loss), fornecendo um sinal de gradiente 4× maior por amostra. A principal contribuição deste estudo não reside no desempenho de classificação, mas sim no caminho de implantação: a conversão para TFLite (ausente em [de Oliveira et al. 2025]), o perfilamento quantitativo de latência em seis configurações de runtime, e o pipeline em C++ demonstrando que um P95 $\leq$ 3ms é alcançável em hardware de borda de baixo custo.

A configuração C++ + TFLite + XNNPACK + SCHED_FIFO atinge uma latência de inferência média de 2,77ms e uma mediana de 2,80ms, ambas consistentemente abaixo do limite de 3,ms em todos os quatro cenários de ataque avaliados (Tabela 4).

No nível do percentil P95, o requisito é atendido em dois dos quatro cenários (Masquerade: 2,998,ms; Poisoning: 2,976,ms), enquanto os outros dois o excedem por margens de 7,µs (Replay: 3,007,ms) e 17,µs (Injeção de Mensagem: 3,017,ms). A média do P95 entre os cenários, de 2,999,ms, coincide essencialmente com o orçamento de 3,ms. No nível do percentil P99, todos os cenários excedem o limite por 64–235,µs, o que é atribuível a preempções esporádicas do escalonador de uso geral do Linux, e não ao cálculo do modelo, conforme evidenciado pela mediana estável — que permanece abaixo de 2,83,ms em cada cenário. Espera-se que a eliminação desses picos residuais por meio de um patch de kernel PREEMPT_RT ou por isolamento de núcleo ($\mathrm{isolcpus}$) traga o P99 para menos de 3,ms, mas isso estava fora do escopo deste estudo. Até onde sabemos, nenhum dos trabalhos pesquisados — incluindo as revisões abrangentes de [Quincozes et al. 2024] e as avaliações de hardware em [Rodríguez et al. 2021] — relata o perfilamento empírico de latência ponta a ponta de um IDS baseado em DLpara GOOSE em hardware ARM com restrição de recursos, nem um pipeline de inferência TFLite em C++ para essa classe de problema. O presente estudo aborda essa lacuna ao fornecer a primeira caracterização quantitativa de latência em seis configurações de implantação e quatro cenários de ataque em um Raspberry Pi 3 sob condições reais de retransmissão de tráfego.

7. Conclusão e Trabalhos Futuros

Este artigo apresentou um IDS orientado à borda para comunicações GOOSE da IEC 61850, baseado em uma arquitetura GRU-200 unidirecional com janela deslizante de quatro pacotes. A principal contribuição metodológica é a estratégia de perda de sequência completa ($\mathrm{return\_sequences{=}True}$), que fornece quatro atualizações de gradiente por janela e compensa o severo desbalanceamento de classes sem superamostragem sintética, preservando a integridade cronológica dos campos stNum e sqNum. O classificador alcança 99,76% de acurácia e 91,93% de F1-score Macro no conjunto de teste desbalanceado. Além da acurácia, o estudo preenche a lacuna entre modelos teóricos e implantação prática em OT. O desenrolamento do laço da GRU ($\mathrm{unroll{=}True}$) viabiliza a conversão para TFLITE_BUILTINS sem Flex delegate, e o pipeline em C++ com XNNPACK e SCHED_FIFO reduz o overhead de parsing de 34,ms (PyShark) para 0,01,ms (libpcap). O perfilamento empírico em Raspberry Pi 3 confirma latência P95 de 2,99,ms, cumprindo o orçamento da IEC 61850; a latência P99 de 3,05,ms é atribuível ao jitter do escalonador Linux, esperando-se sua eliminação sob kernel PREEMPT_RT. Como trabalhos futuros, pretende-se estender a arquitetura aos protocolos SV e MMS, e integrar o IDS a controladores SDN em ambiente HIL de malha fechada, habilitando mitigação ativa com isolamento automático de IEDs comprometidos.

Referências

Abdelrahman, M. S., Kharchouf, I., Nguyen, T. L., and Mohammed, O. A. (2023). A hybrid physical co-simulation smart grid testbed for testing and impact analysis of cyber-attacks on power systems: Framework and attack scenarios. *Energies*, 16(23):7771.

Bhattacharya, S., Saqib, N., and Govindarasu, M. (2024). Ml-based anomaly detection system for iec 61850 communication in substations. In *2024 IEEE Power & Energy Society General Meeting (PESGM)*, pages 1–5. IEEE.

de Oliveira, J. A., dos Santos, A. F. P., and Salles, R. M. (2025). Rnn for intrusion detection in digital substations based on the iec 61850. *Journal of Information Security and Applications*, 94:104197.

Dewangan, F., Siddiqui, S., Biswal, M., and Sood, V. (2024). Smart meters in smart grid. In *Smart Metering*, pages 1–37. Elsevier.

Fernando, T., Ramachandran, G., Vilathgamuwa, M., and Jayalath, D. (2025). Anomaly detection for goose spoofing attacks in digital substations using deep learning models: A dnn and lstm approach. *IEEE Access*, 13:129709–129720.

International Electrotechnical Commission (2003). Iec 61850: Communication networks and systems for power utility automation. Technical report, IEC, Geneva, Switzerland.

Jay, D. (2023). Deception technology based intrusion protection and detection mechanism for digital substations: A game theoretical approach. *IEEE Access*.

Lozano-Paredes, D., Bote-Curiel, L., Feijoo-Martínez, J. R., Gómez-Talal, I., and Rojo-Álvarez, J. L. (2026). Explainable autoencoder-based anomaly detection in iec 61850 goose networks. *arXiv preprint*.

Nhung-Nguyen, H., Girdhar, M., Kim, Y.-H., and Hong, J. (2024). Machine-learning-based anomaly detection for goose in digital substations. *Energies*, 17(15):3745.

Oliveira, J. A., dos Santos, A. F. P., and Salles, R. M. (2025). Network traffic dataset iec-61850 cyber attacks. Kaggle. Disponível em: https://www.kaggle.com/dsv/11724797.

Quincozes, S. E., Albuquerque, C., Passos, D., and Mossé, D. (2024). Ereno: A framework for generating realistic iec-61850 intrusion detection datasets for smart grids. *IEEE Transactions on Dependable and Secure Computing*, 21(4):3851–3865.

Rodríguez, M., Lázaro, J., Bidarte, U., Jiménez, J., and Astarloa, A. (2021). A fixed-latency architecture to secure goose and sampled value messages in substation systems. *IEEE Access*, 9:50774–50785.

Tobar-Rosero, O. A., Roa-Romero, O. A., Rueda-Carvajal, G. D., Leal-Piedrahita, A., Botero-Vega, J. F., Gutierrez-Betancur, S. A., Branch-Bedoya, J. W., and Zapata-Madrigal, G. D. (2024). Goose secure: A comprehensive dataset for in-depth analysis of goose spoofing attacks in digital substations. *Energies*, 17(23):6098.

Ustun, T. S., Hussain, S. M. S., Ulutas, A., Onen, A., Roomi, M. M., and Mashima, D. (2021). Machine learning-based intrusion detection for achieving cybersecurity in smart grids using iec 61850 goose messages. *Symmetry*, 13(5):826.

Wang, X., Fidge, C., Nourbakhsh, G., Foo, E., Jadidi, Z., and Li, C. (2022). Anomaly detection for insider attacks from untrusted intelligent electronic devices in substation automation systems. *IEEE Access*, 10:6629–6649.

Yang, L., Zhai, Y., Zhang, Y., Zhao, Y., Li, Z., and Xu, T. (2022). A new methodology for anomaly detection of attacks in iec 61850-based substation system. *Journal of Information Security and Applications*, 68:103262.