# GRU vs LSTM: Detecção de Notícias Falsas

Diego Machado Cordeiro, Diogo Martins de Assis, Luiz Felipe Vieira,
Marcos Paulo Freitas da Silva, Renato Paganini Thürler Filho
*Instituto de Ciências Exatas e Informática*
*Pontifícia Universidade Católica de Minas Gerais (PUC Minas)*
Belo Horizonte, Minas Gerais, Brasil
Rua Cláudio Manoel, 1.162, Funcionários
Belo Horizonte – MG - Brasil
{diego.cordeiro, diogo.assis, lfvieira ,marcos.silva.1349199, renato.filho.1094209}@sga.pucminas.br,
@sga.pucminas.br

***Abstract*—A crescente disseminação de notícias falsas (*fake news*) demanda o desenvolvimento de métodos automáticos eficazes para sua detecção. Este estudo compara o desempenho de duas arquiteturas de redes neurais recorrentes — *Long Short-Term Memory* (LSTM) e Gated Recurrent Unit (GRU) — na tarefa de classificação binária de notícias como verdadeiras ou falsas. Utilizando o conjunto de dados *ISOT Fake News Detection Dataset*, o processo metodológico compreendeu pré-processamento textual, tokenização, vetorização e construção de modelos com camadas de *embedding* e redes recorrentes bidirecionais. A avaliação, baseada em métricas como acurácia, precisão, *recall*, F1-*score* e AUC, indicou que ambas as arquiteturas são altamente eficazes. O modelo LSTM com autoatenção alcançou 99,02% de acurácia no conjunto de teste, enquanto o modelo GRU obteve 98,93%, com ambos apresentando F1-*scores* elevados e consistentes. Estes resultados demonstram a robustez de ambas as abordagens, contribuindo para o desenvolvimento de soluções automatizadas no combate à propagação de notícias falsas.**

***Index Terms*—*Fake news*, GRU, LSTM, IOST**

## I. Introdução

Com o crescimento exponencial do acesso à informação por meio da internet e das redes sociais, a disseminação de notícias falsas tornou-se um fenômeno preocupante, com impactos significativos na sociedade, na política e na saúde pública. A facilidade com que conteúdos enganosos são compartilhados compromete a credibilidade de fontes legítimas e dificulta o acesso a informações confiáveis. Nesse contexto, torna-se essencial o desenvolvimento e a avaliação de métodos automáticos capazes de identificar e filtrar conteúdos falsos de forma eficiente. [1]

A detecção automática de notícias falsas é um problema recorrente de classificação de texto, no qual o objetivo é atribuir uma categoria (neste caso, "verdadeira" ou "falsa") a um determinado conteúdo textual. Esse tipo de tarefa pode ser abordado por meio de técnicas de aprendizado de máquina, que consistem em treinar algoritmos para reconhecer padrões nos dados e fazer previsões com base nesses padrões. [2]

Nos últimos anos, os avanços em aprendizado profundo (deep learning) — uma subárea do aprendizado de máquina — têm proporcionado resultados expressivos em tarefas de Processamento de Linguagem Natural (PLN). O aprendizado profundo utiliza redes neurais com múltiplas camadas para extrair representações complexas dos dados, sendo especialmente eficaz em tarefas que envolvem linguagem, imagem e som. [2]

Entre os modelos mais promissores para o tratamento de sequências de texto estão as redes neurais recorrentes (Recurrent Neural Networks – RNNs). Diferentemente das redes neurais tradicionais, as RNNs são projetadas para lidar com dados sequenciais, como frases e documentos, pois possuem uma estrutura que permite "memorizar" informações anteriores da sequência.

Duas variantes avançadas das RNNs são as arquiteturas LSTM (Long Short-Term Memory) e GRU (Gated Recurrent Unit). Ambas foram desenvolvidas para resolver limitações das RNNs tradicionais, como o problema do desvanecimento do gradiente, que dificulta o aprendizado de dependências de longo prazo em sequências.

A LSTM introduz mecanismos chamados de portas (gates), que controlam o fluxo de informações dentro da célula da rede. Essas portas decidem o que deve ser armazenado, esquecido ou passado adiante, permitindo que o modelo retenha informações relevantes por períodos mais longos. [3]

A GRU, por sua vez, é uma versão mais simplificada da LSTM. Ela também utiliza portas para controlar o fluxo de informação, mas com uma estrutura mais compacta e menos parâmetros, o que pode resultar em um treinamento mais rápido e eficiente, especialmente em conjuntos de dados menores ou menos complexos.

Ambas as arquiteturas são capazes de capturar dependências temporais e contextuais em sequências de texto, ou seja, conseguem entender como o significado de uma palavra pode depender das palavras anteriores e posteriores. Essa capacidade é essencial para tarefas como a detecção de *fake news*, onde o contexto e a forma como as informações são apresentadas podem ser determinantes para a classificação correta. [4]

## II. Método

A metodologia adotada neste estudo para a detecção automática de notícias falsas foi estruturada em etapas sequenciais, compreendendo: (i) coleta e pré-processamento do conjunto de dados; (ii) tokenização e vetorização dos textos; (iii) divisão dos dados em conjuntos de treinamento e teste; (iv)

construção, treinamento e avaliação de dois modelos principais de aprendizado profundo, um baseado na arquitetura Gated Recurrent Unit (GRU) e outro em Long Short-Term Memory (LSTM) com um mecanismo de autoatenção; e (v) avaliação da robustez dos modelos por meio de validação cruzada k-*fold*. Cada uma dessas etapas é detalhada a seguir.

### A. Conjunto de Dados

O conjunto de dados utilizado neste estudo foi o "*ISOT Fake News Dataset*". Este *dataset* é publicamente disponível e comumente empregado em pesquisas sobre detecção de notícias falsas, sendo composto por artigos de notícias rotulados como verdadeiros ou falsos. Para este trabalho, os textos foram organizados em um único conjunto, e rótulos binários foram atribuídos: o valor 1 para notícias consideradas verdadeiras e 0 para notícias falsas. A distribuição inicial do *dataset* combinado apresentou 21.417 notícias verdadeiras e 23.481 notícias falsas, indicando um conjunto de dados razoavelmente balanceado.

### B. Pré-processamento dos Dados

O pré-processamento textual é uma etapa fundamental em tarefas de Processamento de Linguagem Natural (PLN), pois visa padronizar e limpar os dados, melhorando a qualidade da entrada para os modelos de aprendizado e potencialmente reduzindo a dimensionalidade do vocabulário.

Para ambos os modelos (GRU e LSTM), foi realizado um processo de limpeza textual inicial. Este processo consistiu na conversão de todo o conteúdo textual para letras minúsculas, seguido pela remoção de caracteres especiais, números e pontuações, mantendo-se apenas caracteres alfabéticos e espaços.

Especificamente para o modelo LSTM com autoatenção, uma etapa adicional de lematização foi aplicada. Utilizou-se o *WordNetLemmatizer* da biblioteca NLTK para reduzir as palavras flexionadas à sua forma base ou lema (e.g., "correndo" para "correr"). Esta etapa busca normalizar o vocabulário e agrupar palavras com o mesmo radical semântico, o que pode auxiliar o modelo a capturar relações de significado de forma mais eficaz. Para o modelo GRU, a lematização não foi aplicada, mantendo-se o texto apenas com a limpeza básica e conversão para minúsculas.

### C. Tokenização e Vetorização

Após a etapa de limpeza, os textos processados foram convertidos em uma representação numérica adequada para entrada nas redes neurais. Este processo envolveu duas subetapas: tokenização e vetorização.

A tokenização consistiu em segmentar cada texto em unidades individuais, denominadas tokens (tipicamente palavras). Para esta tarefa, utilizou-se o Tokenizer da biblioteca Keras. Uma diferença no tratamento entre os modelos foi o tamanho máximo do vocabulário considerado:

- Para o modelo GRU, o vocabulário foi limitado às 5.000 palavras mais frequentes (ma_words=5000).
- Para o modelo LSTM com autoatenção, o vocabulário foi expandido para as 10.000 palavras mais frequentes (max_words=10000), visando acomodar um léxico potencialmente mais rico devido à etapa de lematização e à maior complexidade da arquitetura.

Em ambos os casos, foi definido um token especial (¡OOV¿) para representar palavras que não estavam entre as mais frequentes e, portanto, fora do vocabulário construído.

Subsequentemente, as sequências de tokens foram convertidas em sequências de inteiros. Como as redes neurais exigem entradas de tamanho fixo, aplicou-se a técnica de *padding* (preenchimento) utilizando a função pad_sequences do Keras. Todas as sequências foram padronizadas para um comprimento máximo de 200 tokens (max_len=200). Sequências mais curtas que 200 tokens foram preenchidas com zeros ao final, enquanto sequências mais longas foram truncadas.

### D. Divisão dos Dados

O conjunto de dados vetorizado e padronizado (matriz X contendo as sequências e vetor y contendo os rótulos) foi dividido em subconjuntos de treinamento e teste. Utilizou-se a função train_test_split da biblioteca scikit-learn, destinando-se 80% dos dados para o conjunto de treinamento e os 20% restantes para o conjunto de teste. A divisão foi realizada com random_state=42 para garantir a reprodutibilidade dos resultados.

### E. Arquiteturas e Treinamento dos Modelos

Neste estudo, foram desenvolvidos e comparados dois modelos principais baseados em redes neurais recorrentes, com foco nas arquiteturas GRU e LSTM com autoatenção.

*1) Modelo GRU:* A arquitetura do modelo baseado em *Gated Recurrent Unit* (GRU) foi definida utilizando a API Funcional do Keras, consistindo nas seguintes camadas:

- **Camada de *Embedding*:** Responsável por transformar os *tokens* de entrada (índices inteiros) em vetores densos de representação. Foi configurada com input_dim igual ao tamanho do vocabulário do GRU (5.000) e output_dim de 64, gerando um vetor de 64 dimensões para cada *token*.
- **Camada Bidirecional GRU:** Uma camada GRU com 32 unidades, envolvida por uma camada Bidirectional. A natureza bidirecional permite que a rede processe a sequência de entrada em ambas as direções (passado para futuro e futuro para passado), capturando informações contextuais de forma mais completa.
- **Camada de *Dropout*:** Aplicada após a camada Bidirecional GRU com uma taxa de 0.5 (50% dos neurônios desativados aleatoriamente) para mitigar o *overfitting*.
- **Camada Densa (ReLU):** Uma camada totalmente conectada com 32 unidades, utilizando a função de ativação ReLU (*Rectified Linear Unit*) para introduzir não-linearidade. Foi aplicada regularização L2 com fator de 0.01 nesta camada.

- **Camada de Saída (*Sigmoide*):** Uma camada densa final com uma única unidade e função de ativação sigmoide. A função sigmoide, definida como:

$$F(x) = \frac{1}{1 + e^{-x}}$$

que mapeia a saída para o intervalo [0, 1], que é interpretado como a probabilidade da notícia ser classificada como verdadeira (label 1).

O modelo GRU, com um total de 340.929 parâmetros treináveis, foi compilado utilizando o otimizador Adam e a função de perda binary_crossentropy, adequada para problemas de classificação binária. As métricas monitoradas durante o treinamento e avaliação foram a acurácia e a AUC (*Area Under the ROC Curve*).

*2) Modelo LSTM com Autoatenção:* Para o modelo baseado em *Long Short-Term Memory* (LSTM), foi incorporado um mecanismo de autoatenção para permitir que o modelo ponderasse a importância de diferentes partes da sequência de entrada ao gerar uma representação. A arquitetura, também construída com a API Funcional do Keras, compreendeu:

- **Camada de *Embedding*:** Com input_dim igual ao tamanho do vocabulário do LSTM (10.000) e output_dim de 100.
- **Camada Bidirecional LSTM:** Uma camada LSTM com 32 unidades, envolvida por Bidirectional e configurada com return_sequences=True para que a saída de cada passo de tempo fosse fornecida à camada de atenção subsequente.
- **Camada de *Dropout*:** Aplicada após a camada Bidirecional LSTM com taxa de 0.5.
- **Camada de Autoatenção (*SelfAttention*):** Uma camada customizada implementada com o objetivo de permitir que o modelo ponderasse dinamicamente a importância de diferentes segmentos da sequência de notícias .
- **Camada Densa (ReLU):** Uma camada totalmente conectada com 64 unidades, ativação ReLU e regularização L2 (fator 0.01).
- **Segunda Camada de Dropout:** Com taxa de 0.5.
- **Camada de Saída (Sigmoide):** Uma camada densa final com uma única unidade e função de ativação sigmoide.

O modelo LSTM com autoatenção, totalizando 1.042.498 parâmetros treináveis, foi compilado com os mesmos otimizador (Adam), função de perda (binary_crossentropy) e métricas (acurácia e AUC) do modelo GRU, para permitir uma comparação direta das condições de treinamento.

*3) Treinamento dos Modelos Principais:* Ambos os modelos principais foram treinados por um máximo de 5 épocas (epochs=5) com um tamanho de lote de 32 (batch_size=32). Durante o processo de treinamento, 20% dos dados de treinamento foram utilizados como conjunto de validação (validation_split=0.2) para monitorar o desempenho e evitar sobreajuste. Dois callbacks do Keras foram empregados:

- ***EarlyStopping*:** Configurado para interromper o treinamento se a perda de validação não apresentasse melhora por 2 épocas consecutivas (patience=2), com restore_best_weights=True para reter os pesos do modelo da época com melhor desempenho na validação.
- ***ReduceLROnPlateau*:** Configurado para reduzir a taxa de aprendizado pela metade (factor=0.5) caso a perda de validação não melhorasse por 1 época (patience=1), auxiliando na convergência para ótimos locais.

### *F. Validação Cruzada K-Fold*

Para avaliar a robustez e a capacidade de generalização dos modelos de forma mais rigorosa, foi empregada a técnica de validação cruzada k-*fold*. O conjunto de treinamento original (X_train, y_train) foi dividido em k=5 folds utilizando a função KFold do scikit-learn, com embaralhamento dos dados (shuffle=True) e random_state=42 para reprodutibilidade.

Ambos os modelos foram treinados em cada *fold* por 5 épocas (batch_size=32), utilizando o otimizador Adam, perda binary_crossentropy e métricas de acurácia e AUC. Os callbacks EarlyStopping e ReduceLROnPlateau foram omitidos nesta etapa para assegurar um treinamento uniforme entre os folds. As métricas de validação de cada fold foram então agregadas e promediadas para uma estimativa consolidada do desempenho.

### *G. Métricas de Avaliação*

O desempenho dos modelos foi avaliado utilizando um conjunto padrão de métricas para tarefas de classificação binária:

- **Acurácia (*Accuracy*):** Proporção de previsões corretas em relação ao total de amostras.
- **Precisão (*Precision*):** Capacidade do modelo de não classificar uma notícia falsa como verdadeira (verdadeiros positivos / (verdadeiros positivos + falsos positivos)).
- **Sensibilidade (*Recall*):** Capacidade do modelo de identificar todas as notícias verdadeiras (ou falsas, dependendo da classe positiva) (verdadeiros positivos / (verdadeiros positivos + falsos negativos)).
- **F1-*Score*:** Média harmônica entre precisão e recall, fornecendo uma métrica balanceada.
- **AUC (*Area Under the ROC Curve*):** Mede a capacidade do modelo de distinguir entre as classes positiva e negativa. Um valor próximo de 1 indica excelente poder de discriminação.
- **Matriz de Confusão:** Tabela que visualiza o desempenho da classificação, mostrando verdadeiros positivos, falsos positivos, verdadeiros negativos e falsos negativos.

As curvas de aprendizado (perda e AUC versus épocas) também foram analisadas para observar a convergência e o sobreajuste dos modelos durante o treinamento.

## III. Resultados

Nesta seção, são apresentados os resultados de desempenho dos modelos GRU e LSTM na tarefa de detecção de notícias falsas. A avaliação compreendeu tanto um treinamento com divisão única entre treino e teste, para os modelos principais (GRU e LSTM com autoatenção), quanto uma validação cruzada k-*fold* para aferir a robustez das arquiteturas.

### A. *Desempenho dos Modelos Principais no Conjunto de Teste*

Os modelos GRU e LSTM com autoatenção, conforme descritos na Seção 3.5 (Metodologia), foram treinados utilizando 80% dos dados e avaliados nos 20% restantes. O *callback* *EarlyStopping* selecionou os pesos da época com menor perda de validação durante o treinamento. Para o modelo GRU, a 4ª época foi a ótima (perda de validação: 0.0402, acurácia de validação: 0.9908). Para o modelo LSTM com autoatenção, a 4ª época também foi selecionada como a melhor (perda de validação: 0.0398, acurácia de validação: 0.9914).

A Tabela I sumariza o desempenho de ambos os modelos no conjunto de teste.

TABLE I: Resultados comparativos dos modelos GRU e LSTM no conjunto de teste (divisão 80/20).

| **Métrica** | **GRU** | **LSTM** |
|---|---|---|
| Acurácia | 0.9893 | 0.9902 |
| AUC | 0.9986 | 0.9980 |
| Perda (Loss) | 0.0448 | 0.0460 |
| **Classe '*Fake*' (0)** | | |
| Precisão | 0.99 | 0.99 |
| Recall | 0.99 | 0.99 |
| F1-Score | 0.99 | 0.99 |
| **Classe '*True*' (1)** | | |
| Precisão | 0.99 | 0.98 |
| Recall | 0.99 | 0.99 |
| F1-Score | 0.99 | 0.99 |

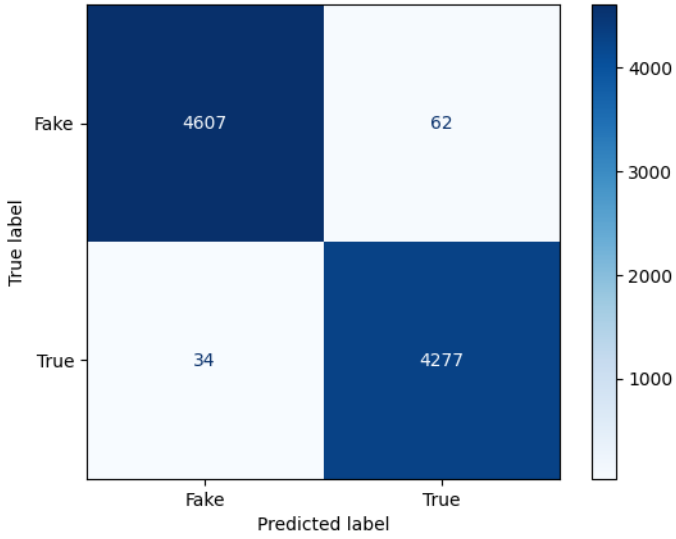



Fig. 1: Matriz de confusão para o modelo GRU no conjunto de teste.

Ambos os modelos demonstraram um desempenho excepcional no conjunto de teste. O modelo LSTM com autoatenção apresentou uma acurácia superior (99.02%) em comparação com o modelo GRU (98.93%). Em contrapartida, o modelo GRU obteve uma AUC ligeiramente maior (0.9986) que o LSTM com autoatenção (0.9980). As métricas de Precisão, *Recall* e F1-*Score* para as classes individuais ('*Fake*' e '*True*') foram idênticas e muito elevadas (0.99) para ambos os modelos, indicando excelente equilíbrio e capacidade de

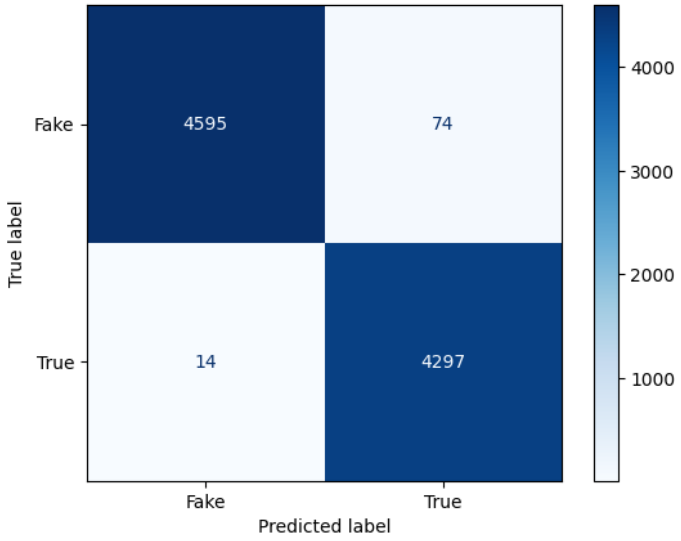
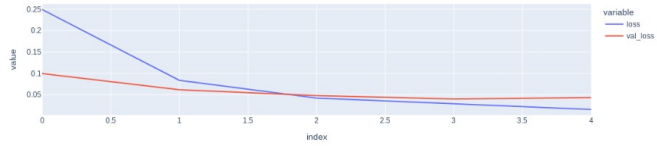
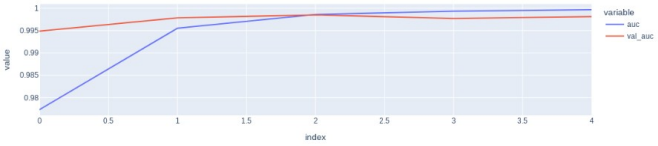



Fig. 2: Matriz de confusão para o modelo LSTM no conjunto de teste.

identificação correta para ambas as categorias. As matrizes de confusão (Figura 1) e (Figura 2) confirmaram as baixas taxas de erro para ambos os modelos.




(a) Curva de Perda (*Loss*) do GRU.

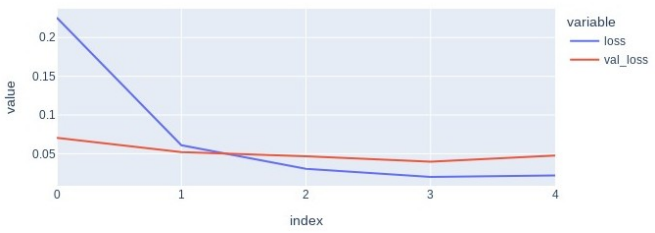



(b) Curva de AUC do GRU.

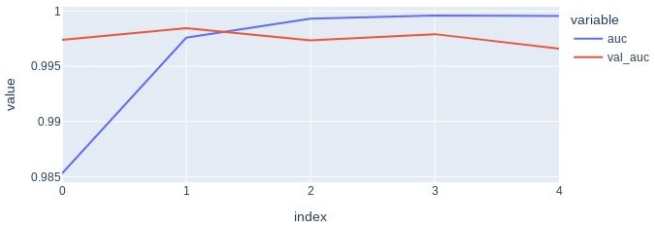
Fig. 3: Curvas de aprendizado para o modelo GRU durante o treinamento

As curvas para o modelo GRU (Figura 3), observa-se uma rápida redução da perda de treinamento (*loss*) e validação (val_loss) nas épocas iniciais. A perda de validação estabilizou-se em um valor baixo a partir da segunda época, indicando boa convergência e que o *EarlyStopping* foi eficaz ao selecionar os pesos da 4ª época, prevenindo *overfitting*. As curvas de AUC (Figura 3b) para o GRU também demonstram uma rápida ascensão, com a AUC de validação (val_auc) atingindo valores próximos de 0.998 já nas primeiras épocas e mantendo-se estável.

De forma similar, o modelo LSTM com autoatenção (Figura 4) exibiu uma diminuição acentuada da perda de treinamento e validação nas primeiras épocas, com a val_loss atingindo seu mínimo na 4ª época, cujos pesos foram restaurados pelo *EarlyStopping*. A AUC de validação (val_auc) para o LSTM




(a) Curva de Perda (Loss) do LSTM.




(b) Curva de AUC do LSTM.

Fig. 4: Curvas de aprendizado para o modelo LSTM durante o treinamento.

também alcançou rapidamente um patamar elevado, próximo de 0.997, e permaneceu estável, conforme ilustrado na Figura 4b.

Em ambos os modelos, as curvas de aprendizado indicam uma convergência eficiente e a efetividade do *EarlyStopping* em selecionar um ponto ótimo de treinamento, evitando *overfitting*, uma vez que as perdas de validação não apresentaram aumento substancial após atingirem seus mínimos."

### B. Resultados da Validação Cruzada K-Fold

Para avaliar a robustez, a técnica de validação cruzada k-*fold* (k=5) foi aplicada. Para o GRU, utilizou-se a arquitetura principal descrita na Seção 3.5.1. Para a abordagem LSTM, foi empregada a arquitetura com autoatenção.

A Tabela II apresenta as médias das métricas de desempenho obtidas nos *folds* de validação.

TABLE II: Resultados médios da validação cruzada k-*fold* (k=5) para os modelos GRU e LSTM.

| **Métrica Média** | **GRU** | **LSTM** |
|---|---|---|
| Acurácia de Validação | 0.9848 | 0.9835 |
| AUC de Validação | 0.9968 | 0.9947 |
| Perda de Validação | 0.0565 | 0.0697 |

Na validação cruzada, o modelo GRU apresentou uma acurácia média de validação de 98.48% e uma AUC média de 0.9968. O modelo LSTM alcançou uma acurácia média de validação de 98.35% e AUC média de 0.9947. Os relatórios de classificação por *fold* mostraram F1-*scores* consistentemente altos (0.98-0.99) para ambas as classes em ambos os modelos durante a validação cruzada. Estes resultados indicam que ambas as abordagens são robustas e generalizam bem para diferentes subconjuntos dos dados de treinamento.

### C. Comparativo de Desempenho

Analisando os resultados da avaliação no conjunto de teste principal (Tabela I), tanto o GRU quanto o LSTM atingiram um patamar de desempenho muito similar e extremamente alto, com acurácias em torno de 99% e AUCs superiores a 0.997. O LSTM obteve uma acurácia levemente superior (0.9902 vs 0.9893), enquanto o GRU teve uma AUC marginalmente melhor (0.9986 vs 0.9980).

Na validação cruzada k-*fold* (Tabela II), o GRU também demonstrou um desempenho ligeiramente superior em termos de acurácia média (98.48% vs 98.35%) e AUC média (0.9968 vs 0.9947).

Considerando o número de parâmetros treináveis ($\approx 341k$ ; $\approx 1.04M$), o modelo GRU, sendo mais simples, alcançou um desempenho comparável, e em algumas métricas ligeiramente superior, ao LSTM mais complexo. Isso sugere uma excelente eficiência do GRU para esta tarefa específica.

## IV. Conclusões

Este trabalho investigou o desempenho comparativo das arquiteturas de redes neurais recorrentes GRU e LSTM na tarefa de detecção de notícias falsas, utilizando o *dataset* ISOT. Os experimentos realizados demonstraram que ambas as abordagens alcançaram resultados expressivos, com métricas de desempenho elevadas e consistentes.

Na avaliação principal utilizando uma divisão de 80% dos dados para treino e 20% para teste, o modelo LSTM obteve uma acurácia de 99,02%, enquanto o modelo GRU alcançou 98,93%. Ambos os modelos apresentaram F1-*scores* de 0,99 para as classes "*Fake*" e "*True*", indicando um excelente equilíbrio entre precisão e *recall*. Na validação cruzada com $k = 5$ *folds*, o modelo GRU apresentou uma acurácia média de 98,48%, e um modelo LSTM obteve 98,35%, reforçando a robustez de ambas as arquiteturas.

Esses achados sublinham a eficácia das redes recorrentes para esta tarefa, mas também apontam para a importância de considerar o contexto de aplicação. A GRU, com sua arquitetura mais enxuta ($\approx 341k$ parâmetros), mostrou-se computacionalmente mais eficiente, sendo vantajosa em cenários com restrições de recursos, sem um sacrifício significativo de performance. Por outro lado, a LSTM, especialmente quando combinada com mecanismos como a autoatenção ($\approx 1.04M$ parâmetros), possui uma capacidade teórica maior para modelar dependências mais longas e complexas, o que pode ser benéfico em contextos textuais mais desafiadores.

Portanto, a escolha entre GRU e LSTM para a detecção de notícias falsas deve ser ponderada não apenas pelos resultados quantitativos de desempenho, mas também por fatores práticos como eficiência computacional, complexidade de implementação e a natureza específica dos dados. Esta análise comparativa contribui para uma tomada de decisão mais informada.

## References

[1] N. Chaudhuri, G. Gupta, M. Bagherzadeh, T. Daim, and H. Yalcin, "Misinformation on social platforms: A review and research agenda,"

*Technology in Society*, vol. 78, p. 102654, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0160791X24002021

[2] S. Khare, P. Singh, and P. Kumar, “An analysis of various classification algorithms for fake news detection,” in *Proc. 2025 2nd Int. Conf. Comput. Intell., Commun. Technol. Netw. (CICTN)*, 2025, pp. 268–272, doi: 10.1109/CICTN64563.2025.10932522

[3] T. S. Camelia, F. R. Fahim, and M. M. Anwar, “A regularized LSTM method for detecting fake news articles,” in *Proc. 2024 IEEE Int. Conf. Signal Process., Inf., Commun. Syst. (SPICSCON)*, 2024, pp. 01–06, doi: 10.1109/SPICSCON64195.2024.10941441.

[4] S. R. Tanuku, “Novel approach to capture fake news classification using LSTM and GRU networks,” in *Proc. 2022 Int. Conf. Futur. Technol. (INCOFT)*, Belgaum, India, 2022, pp. 1–4, doi: 10.1109/INCOFT55651.2022.10094467