A New Approach to River Flow Forecasting: LSTM and GRU Multivariate Models

G. de Melo, D. Sugimoto, P. Tasinaffo, A. Moreira, A. Cunha, and L. Dias

***Abstract*—Hydroelectric power stations are responsible for renewable energy generation, especially in countries with many rivers such as Brazil. It is very important to have good estimates of the hydrological flow in order to determine whether thermoelectric power plants should begin operation, an event that would increase the costs of electricity and also have a terrible environmental impact. The monthly flow of a river was estimated using two recurrent neural networks techniques: Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The results were compared with other articles that had the same structure and used the same data: the Rio Grande river in the Furnas and Camargos dam.**

***Index Terms*—Artificial neural networks, Long short term memory, River flow, Time series forecasting, Gated recurrent unit.**

I. Introdução

A VAZÃO dos rios que compõem a bacia hidrográfica constitui uma métrica essencial para o controle das comportas e da geração de energia de usinas hidroelétricas, sendo esse último fator crucial para a decisão do acionamento das termoelétricas, de operação mais cara e com maiores impactos ambientais. Dessa forma, o objetivo da previsão é reduzir o risco dessas tomadas de decisão. Os modelos estocásticos mais utilizados para a previsão da vazão de rios pertencem à classe de Modelos ARIMA (Média Móvel Integrada Autorregressiva) propostos por Box e Jenkins (1976) [1] que, apesar de apresentarem melhores resultados frente a modelos mais complexos quando a quantidade de dados é relativamente pequena (da ordem de cem), são superados ao se utilizar uma dimensionalidade e cardinalidade maiores em seus conjuntos de treinamento.

A utilização de redes neurais artificiais (ANN) na predição de séries temporais se encontra cada vez mais difundida na academia [2][3][4].

G. Adriano de Melo, Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, Brasil, gam@ita.br.

D. N. Sugimoto, Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, Brasil, dylan.ns@gmail.com.

P. M. Tasinaffo, Instituto Tecnológico de Aeronáutica, São José dos Campos, São Paulo, Brasil, tasinaffo@ita.br.

A. H. Moreira Santos, Escola Federal de Engenharia de Itajubá, Itajubá, Minas Gerais, Brasil, afonso@unifei.edu.br.

A. M. Cunha, Instituto Tecnológico de Aeronáutica, São José dos Campos - SP, cunha@ita.br.

L. A. V. Dias, Instituto Tecnológico de Aeronáutica, São José dos Campos - SP, vdias@ita.br.

Uma técnica amplamente utilizada é separar uma janela temporal para cada dado da entrada, caracterizando-se $N-1$ dados históricos para uma rede com múltiplas camadas perceptron (MLP) [5]. Dessa forma, o sistema consegue apenas considerar uma quantidade fixa no tempo. O modelo de ANN encontra uma boa performance em predição de séries temporais [6] em contraste com modelos estatísticos e em [7] para séries financeiras.

Este trabalho utiliza uma topologia específica de redes neurais artificiais recorrentes (RNN) denominada de *Long-Short Term Memory*, a fim de aproveitar o caráter sequencial dos dados de vazão dos rios. Utilizaram-se as medidas das usinas de Furnas e de Camargos como o estudo de caso do modelo. Ambas as usinas estão situadas no Rio Grande, assim, pode-se comparar com os resultados obtidos por [2] e [8] na caracterização empírica de MLP nos dados da usina de Furnas.

Embora várias técnicas de previsão de vazão de rios tenham sido realizadas por meio de redes neurais, a utilização de novas arquiteturas LSTM e GRU que consideram a sequência temporal em mais longos horizontes ainda não havia sido investigada. Espera-se que os modelos de aprendizado profundo sejam mais efetivos frente aos modelos clássicos à medida que mais dados são disponibilizados aos seus treinamentos.

Dessa forma, este trabalho foi dividido da seguinte maneira: a Seção II realizará o desenvolvimento teórico das redes neurais artificiais, evidenciando-se as arquiteturas *Long-Short Term Memory* (LSTM) [9] e *Gated Recurrent Unit* (GRU) [10]. A Seção III desenvolve a metodologia de análise da construção das redes neurais, enquanto a Seção IV compara e analisa os resultados obtidos. Por fim, realizam-se as considerações finais desse artigo na Seção V.

II. Modelo Teórico das Redes Neurais

As redes neurais podem ser interpretadas não apenas como aproximadores universais de funções, mas também como transformações aplicadas no espaço dos dados de entradas capazes de progressivamente, a cada camada, separar representações e encontrar variáveis latentes, isto é, cada camada de uma rede neural pode ser interpretada como aplicações sucessivas de funções na qual deve existir concordância da representação dos dados entre cada camada. [11].

Embora tenham a inspiração da biologia, os modelos de combinação linear e de aprendizado, na realidade uma otimização de parâmetros, se distanciam da simulação dos processos biológicos, de sua reprodução e também de suas dinâmicas, sendo as suas semelhanças amplamente debatidas [12] [13]. A álgebra matricial e o cálculo multivariado são as

principais ferramentas de construção, análise e otimização das redes neurais, sendo a analogia biológica conveniente para sua interpretação.

Dessa forma, o projeto de uma rede neural representa grandes desafios de engenharia, na estimação de hiperparâmetros de controle e na arquitetura e topologia da rede em si. Tal como a engenharia aeronáutica, no início do século XX, tinha os parâmetros de seus veículos mais pesados que o ar estimados por equações empíricas, devido à complexidade e a intratabilidade de soluções analíticas e até numéricas, o desenvolvimento das técnicas de aprendizado profundo também se encontra em sua infância, com grandes avanços, ainda que baseados em metodologia empíricas.

A. Redes Neurais Feedforward

A unidade básica de computação de uma rede neural é o neurônio, termos que foram inspirados pela biologia. Ocorre uma combinação linear das entradas do neurônio cujo sinal de saída é modulado por uma função de ativação não linear. Para um neurônio, os sinais de entrada são representados por um vetor dos sinais de saída da camada anterior, conforme observado na Fig. 1. A representação matemática está indicada pela Equação 1, na qual a indica a ativação resultante do neurônio, $\sigma(z)$ é uma função de ativação, x_i é a i-ésima entrada, w_i é o peso sináptico (*weight*) associado a i-ésima entrada e b é a polarização do neurônio (*bias*).

$$a = f(X) = \sigma(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n + b) \tag{1}$$

A representação completa se dá por meio da matriz que representa os pesos de cada camada. Dessa forma, a computação de uma camada de índice $N-1$ com 3 neurônios para outra de índice N com 4 neurônios pode ser escrita pela Equação 2. Nessa equação, $A^{(N)}$ representa as ativações do neurônio da N-ésima camada, $w_{ij}^{(N)}$ representa o peso sináptico entre os neurônios i da camada N e j da camada *N-1*, $b_i^{(N)}$ representa o limiar de polarização do i-ésimo neurônio da camada N e $\sigma(z)$ é a função de ativação que deve ser aplicada a cada elemento da matriz.

$$A^{(N)} = \sigma\left(\begin{bmatrix} w_{11}^{(N)} & w_{12}^{(N)} & w_{13}^{(N)} & b_1^{(N)} \\ w_{21}^{(N)} & w_{22}^{(N)} & w_{23}^{(N)} & b_2^{(N)} \\ w_{31}^{(N)} & w_{32}^{(N)} & w_{33}^{(N)} & b_3^{(N)} \\ w_{41}^{(N)} & w_{42}^{(N)} & w_{43}^{(N)} & b_4^{(N)} \end{bmatrix} \begin{bmatrix} a_1^{(N-1)} \\ a_2^{(N-1)} \\ a_3^{(N-1)} \\ 1 \end{bmatrix}\right) \tag{2}$$

É importante notar que o poder computacional da rede neural se dá por meio da não linearidade da função de ativação (ver [12] e [13]) uma vez que a combinação linear de funções lineares seria também uma função linear, o que restringiria a rede neural a apenas regressões lineares. Uma abstração desse fenômeno é conceber a não linearidade como uma regra IF-ELSE da programação imperativa, algo que fica claro nas funções de ativação Retificadas Linearmente (ReLU – *Rectified Linear Unit*), que apresentam uma verificação de sinal, isto é, definida pela equação $ReLU(x) = \max(0, x)$. Outra função de ativação comumente usada é a sigmoide, definida por $\sigma(x) = (1 + e^{-x})^{-1}$, que mapeia valores entre 0 e 1.

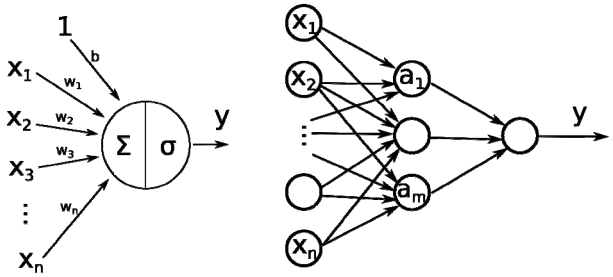

Fig. 1. Esquema de um neurônio e uma rede neural *feedforward* [5].

A função objetivo define o custo, o quão distante o resultado está. Ela deve ser minimizada. Isso caracteriza o que se chama de aprendizado supervisionado, isto é, quando o agente é treinado com valores de entrada já mapeados com valores de saída conhecidos. Dessa forma a função de custo (também chamada de *loss, cost, objective*) tem como valores de entrada os pesos e os limiares que definem a rede e tem como parâmetros fixos, os casos de treinamento. A Equação 3 define uma representação do custo como sendo o erro quadrático médio, isto é, a média entre os quadrados das diferenças entre o valor de saída esperado y_i e o valor de saída predito $\hat{y}_i$ no conjunto de N casos de treinamento. O custo, dessa forma, se torna uma função dos parâmetros W e B do modelo neural, que representam os pesos e polarizações neurais.

$$C(W, B) = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{3}$$

Dessa forma, o aprendizado consiste simplesmente em diminuir essa função de custo em todo o conjunto de treino. O método do *Gradient Descent* (GD) é de primeira ordem em contraste com Levenberg-Marquardt, Quasi-Newton e outros algoritmos utilizados por [19] que, apesar de convergirem em menos iterações por meio do cálculo da segunda derivada, levam mais tempo e gastam o quadrado de espaço, o que deixa a otimização intratável para redes profundas. Dessa forma no GD, a atualização de um determinado peso se dá pela derivada parcial do custo com relação a esse peso ponderados por uma taxa de aprendizado α, conforme apresentado na Equação 4.

$$\Delta w_{ij}^{(N)} = -\alpha \frac{\partial C}{\partial w_{ij}^{(N)}} = -\alpha \frac{\partial C}{\partial x_i} \frac{\partial x_i}{\partial w_{ij}^{(N)}} \tag{4}$$

Uma alternativa à utilização de todo o conjunto de treino é o *mini-batch*, uma forma de *Stochastic Gradient Descent* (SGD), no qual utiliza-se apenas um subconjunto do treino (*batches*) para realizar uma atualização preliminar do pesos. Ao percorrer uma vez todo o conjunto de treino, várias atualizações já terão ocorrido, diminuindo-se assim o tempo

de treinamento. A estocasticidade também promove uma busca aleatória na vizinhança dos parâmetros atuais da rede. Isso permite uma melhor tratabilidade, contribuindo para que a otimização não fique presa em mínimos locais. Há outras heurísticas que também melhoram a sua performance, com taxas de aprendizado adaptativo e o cálculo da quantidade de movimento da função, a exemplo do RMSProp, SGD-Nesterov Momentum, AdaDelta, AdaGrad e Adam [18].

Empregado neste trabalho, o *Adaptive Moment Estimation* (Adam) se baseia no gradiente do custo com relação aos pesos $\nabla_W C$ e calcula a suas médias móveis exponenciais (MME) expressas por V_{atual} na Equação 5.a e por S_{atual} que representa o MME do quadrado do gradiente elemento a elemento observado na Equação 5.b. Essas médias móveis são ponderadas respectivamente pelos parâmetros β_1 e β_2 valorados entre 0 e 1. O peso, expresso na Equação 5.c, é atualizado pelo cálculo da razão entre o momento de primeira e a raiz do momento de segunda ordem somada a uma pequena constante ε a fim de evitar divisões por zero. Matematicamente, a Equação 5 indica a formulação matemática do Gradiente Descendente, no qual os pesos são atualizados de forma proporcional ao gradiente do custo.

$$V_{atual} = \beta_1 V_{antigo} + (1-\beta_1)\nabla_W C \tag{5.a}$$

$$S_{atual} = \beta_2 S_{antigo} + (1-\beta_2)(\nabla_W C)^2 \tag{5.b}$$

$$W_{atual} = W_{antigo} - \alpha V_{atual} \big/ (\sqrt{S_{atual}} + \varepsilon) \tag{5.c}$$

A otimização do sistema, isto é, o aprendizado, se dá por meio de um tipo peculiar de descida por gradiente: a retropropagação (*backpropagation*). É um método que calcula as derivadas parciais da função de custo com relação aos pesos sinápticos. A vantagem da retropropagação é a redução da quantidade de computação e espaço necessário, uma vez que a variação dos pesos de uma camada é calculada apenas em função da camada anterior, o que acumula os cálculos de forma recursiva, como indicado na Equação 6.

$$\frac{\partial C}{\partial Z^{(K-1)}} = \frac{\partial C}{\partial Z^{(K)}} \frac{\partial Z^{(K)}}{\partial A^{(K-1)}} \frac{\partial A^{(K-1)}}{\partial Z^{(K-1)}} \tag{6}$$

B. Redes Neurais Recorrentes

O modelo de redes neurais recorrentes (RNN – *Recurrent Neural Network*) fornece uma arquitetura que encapsula as informações temporais. Isto é apropriado para previsão de séries temporais hidrológicas uma vez que se baseia nos valores anteriores da série, a depender do número de componentes de memória. Há ciclos de retroalimentação (*feedback*) nos neurônios de uma RNN, que de forma generalizada pode enviar sinais em qualquer direção de e para todas as camadas [1]. Assim, a saída da rede não depende apenas das entradas externas que recebe, mas também do estado da rede no período de tempo anterior, observado na Equação 7.

$$a = f(X,t) = \sigma(w_1 x_1 + \ldots + w_n x_n + b + \omega\ y(t-1)) \tag{7}$$

O algoritmo de retropropagação no tempo (*Back-Propagation Through Time*) considera as derivadas parciais não apenas relacionadas à entrada no mesmo instante de tempo que a saída, mas também com relação a todas as derivadas passadas, como representado pela Equação 8.

$$\frac{\partial v_v(t-q)}{\partial v_u(t)} = \sum_{l_1=1}^{n} \ldots \sum_{l_{q-1}=1}^{n} \prod_{m=1}^{q} \sigma'_{l_m}(a_{l_m}(t-m)) w_{l_m l_{m-1}} \tag{8}$$

A Fig. 2 ilustra como o modelo recorrente pode ser desdobrado na dimensão temporal. Dessa forma, pode-se caracterizar a rede recorrente como uma rede profunda, porém com pesos iguais entre as camadas temporais.

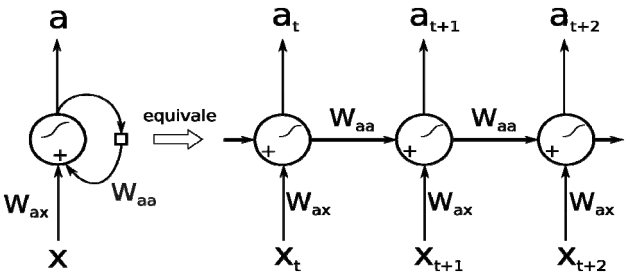

Fig. 2. Esquema de um neurônio em uma rede recorrente.

Assim, como arquiteturas específicas do modelo recorrente, podemos citar as LSTM e as GRU como os seus representantes mais notórios. Esses modelos especiais de redes recorrentes foram criados para tentar modelar uma dependência de mais longo prazo, uma vez que o treinamento e o funcionamento do modelo mais simples não conseguia um bom ajuste dos pesos iniciais dado que o sinal de retropropagação do gradiente diminuía a cada camada.

C. Arquitetura Long-Short Term Memory (LSTM)

As LSTM foram propostas inicialmente por [9] como uma primeira solução para as dependências de longo prazo que sofriam. A essência do funcionamento da LSTM está em um estado adicional que é o estado da célula (*cell state* c_t), que é praticamente inalterado no processo de iteração da unidade. Há três tipos diferentes de portas (*gates*): porta de entrada (*input gate* i_t), porta de saída (*output gate* o_t) e porta de esquecimento (*forget gate* f_t). Sua representação pode ser observada na Fig. 3.

A porta de esquecimento, representada na Equação 9.a, é responsável por apagar o estado da célula, para isso ela utiliza uma função de ativação sigmoide (tem sua imagem entre 0 e 1) cuja saída multiplica diretamente o estado da célula, que será atualizado para um valor menor ou igual ao valor anterior. Já a porta de entrada, na Equação 9.b, é responsável por somar o valor atual da entrada no estado da célula, fazendo uso de uma função de ativação sigmoide e ainda do estimador de estado $\tilde{c}_t$ com uma tangente hiperbólica, representada na Equação 9.c.

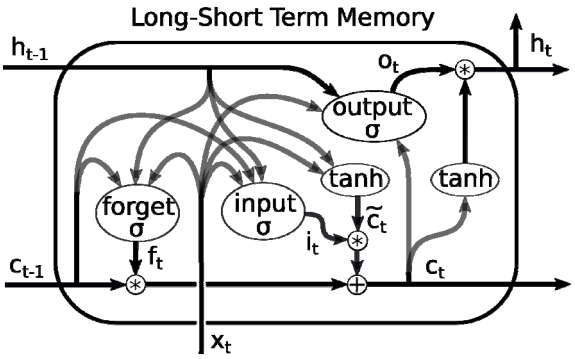

Fig. 3. Arquitetura interna da *Long-Short Term Memory* (LSTM).

No último passo da caracterização da LSTM, há a porta de saída na Equação 9.d, que representa a influência do estado da célula e dos valores de entrada sobre o valor de saída da unidade (Equação 9.e). Com relação ao estado da célula (Equação 9.f), realiza-se a soma de seu valor anterior ponderado pelo fator de esquecimento com o estimador do novo estado ponderado pelo fator de entrada.

Em termos matemáticos, as seguintes equações modelam o comportamento de uma célula *j* do tipo LSTM. Tal célula pode ser interpretada como um agregado de neurônios com uma topologia bem característica, definido pelo conjunto de estados representados no grupo de Equações 9.

$$f_t^j = \sigma(W_f x_t + U_f h_{t-1} + V_f c_{t-1})^j \tag{9.a}$$

$$i_t^j = \sigma(W_i x_t + U_i h_{t-1} + V_i c_{t-1})^j \tag{9.b}$$

$$\tilde{c}_t^j = \tanh(W_c x_t + U_c h_{t-1})^j \tag{9.c}$$

$$o_t^j = \sigma(W_o x_t + U_o h_{t-1} + V_o c_t)^j \tag{9.d}$$

$$h_t^j = o_t^j \tanh(c_t^j) \tag{9.e}$$

$$c_t^j = f_t^j c_{t-1}^j + i_t^j \tilde{c}_t^j \tag{9.f}$$

D. Arquitetura Gated Recurrent Unit (GRU)

De forma semelhante, as GRU, foram propostas como uma melhoria da eficiência computacional das LSTM por meio de simplificações, embora sem perdas de desempenho [10]. A unidade da GRU tem apenas duas portas e não apresenta o estado da célula. São as portas de reinício (*reset gate* r_t) e as portas de atualização (*update gate* z_t) que caracterizam esse modelo, observado na Fig. 4. a saída h_t se tornou o próprio estado da célula da Equação 10.a. que é uma combinação linear do estado antigo e do estimador novo.

A porta de atualização apresentada na Equação 10.b faz o papel simultaneamente do que seriam as portas de esquecimento e entrada na LSTM. A intuição é que ao esquecer um estado anterior também deve-se incluir um novo estado, e isso se dá em apenas um passo. Novamente, essa porta se relaciona com um estimador do novo estado $\tilde{h}_t$ da Equação 10.c que se utiliza de uma função tangente hiperbólica.

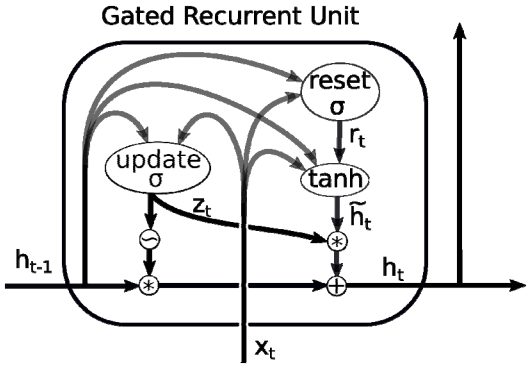

Fig. 4. Arquitetura interna da *Gated Recurrent Unit* (GRU).

Já a porta de reinício da Equação 10.d se assemelha também à porta de esquecimento, porém ela tem uma função de ativação tangente hiperbólica. Dessa forma, não há necessidade de uma porta de saída, uma vez que não há um estado da célula, apenas a sua própria saída. As definições dos seus estados estão expressas na Equação 10.

$$h_t^j = (1 - z_t^j) h_{t-1}^j + z_t^j \tilde{h}_t^j \tag{10.a}$$

$$z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j \tag{10.b}$$

$$\tilde{h}_t^j = \tanh(W x_t + U(r_t \otimes h_{t-1}))^j \tag{10.c}$$

$$r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j \tag{10.d}$$

III. CONSTRUÇÃO DAS REDES

Com base no trabalho de [14], a metodologia proposta neste trabalho consiste nos seguintes estágios fundamentais:

1. Determinação e seleção de variáveis de entrada.
2. Identificação da estrutura das redes neurais para dados de vazão de rios.
3. Determinação dos parâmetros da rede neural.

Dessa forma, cada uma das etapas propostas busca sistematicamente definir uma metodologia aplicável a qualquer tipo de dados de vazão de rios.

O conjunto de dados em relação aos rios é composto por duas séries históricas, cada uma representa a vazão média em m^3/s de um determinado mês das usinas hidroelétricas de Furnas e Camargos. Estas séries históricas apresentam alta correlação que pode ser explicada pelo fato de que ambas as usinas estão situadas no Rio Grande, na bacia do Rio Paraná. Os dados são de 82 anos do histórico de operação dessas duas usinas hidroelétricas, totalizando 984 amostras mensais desde janeiro de 1931 até dezembro de 2012, adquiridos pela ONS (Operador Nacional do Sistema Elétrico). Assim, a rede neural recebe dados simultaneamente das duas séries temporais e faz uma previsão também simultânea para as duas, caracterizando-se um modelo multivariado tal como em [15].

Em uma primeira análise, observou-se a autocorrelação da série temporal, importante para a detecção dos níveis de similaridade e periodicidade, tal qual [14]. Com relação ao modelo de predição de múltiplos passos, a utilização mapas auto-organizados (SOM) na previsão da vazão [16] também forneceu inspiração para a arquitetura proposta, por meio da utilização de entradas auxiliares para a rede neural. A definição da autocorrelação normalizada de uma série temporal discreta é dada pela Equação 11, onde f é uma série temporal discreta, $\overline{f}$ é o seu conjugado complexo e k é o atraso.

$$R_{ff}(k) = \sum_{n=0}^{N} f(n)\overline{f}(n-k) \Big/ \sum_{n=0}^{N} f(n)\overline{f}(n) \tag{11}$$

Observa-se na Fig. 5 o gráfico da autocorrelação da série temporal da vazão mensal de Furnas, notando-se os picos de aproximadamente 60% de correlação a cada intervalo de um ano. A sombra em azul-escuro representa o intervalo de confiança de 95% da hipótese de que a correlação observada seja significativa e não resultado aleatório. Além disso, a correlação com o atraso de apenas um mês é de aproximadamente 70%, o que estabelece uma base para os modelos.

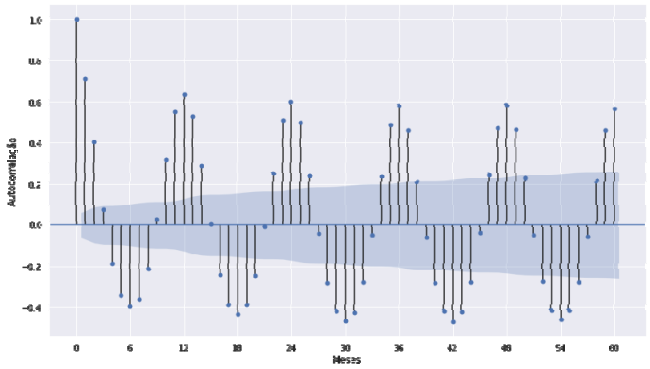

Fig. 5. Autocorrelação da série temporal das vazões do Rio Grande na usina de Furnas.

Além da autocorrelação, calculou-se a correlação entre as séries temporais das usinas de Furnas e de Camargos, com um valor de aproximadamente 0,95. Ambas as usinas estão localizadas na Bacia do Rio Paraná e estão a uma distância de aproximadamente 100 km, compartilhando não apenas a proximidade quanto também vários afluentes. Dessa forma, espera-se que o modelo multivariado seja capaz de utilizar esse dois dados com alta correlação para aplicar um filtro sobre eles, encontrando características internas sobre os períodos de chuva, por exemplo, e diminuindo o ruído de amostragem dos dados.

Utilizou-se a biblioteca *Keras* [17] para o desenvolvimento da rede neural em *Python3*, em um ambiente de desenvolvimento completamente *open-source*. Para fins didáticos, facilitando o entendimento e a visualização dos códigos e resultados, utilizou-se a plataforma *Jupyter Notebook*, também de código aberto, para o desenvolvimento dos modelos e observação dos gráficos gerados pela interface web.

Empregou-se o otimizador Adam, um algoritmo para otimização baseada em gradiente de primeira ordem de funções objetivas estocásticas, calculando-se as taxas individuais de aprendizagem adaptativa para diferentes parâmetros a partir das estimativas do primeiro e segundo momentos dos gradientes [18]. Os parâmetros de treinamento foram os recomendados pelo artigo original e implementados por padrão pela biblioteca Keras taxa de aprendizado 0,001, $\beta_1 = 0{,}9$; $\beta_2 = 0{,}999$; $\varepsilon = 10^{-8}$ e decaimento da taxa de aprendizado nulo. Os hiperparâmetros $\beta_1, \beta_2, \varepsilon$ representam, respectivamente, as taxas de decaimento dos momentos de primeira e segunda ordem, e o valor mínimo para evitar a divisão por zero.

Outro fator importante a ser levado em consideração é o superajuste (*overfitting*) do modelo. Uma das técnicas para evitar o superajuste é diminuir o número de parâmetros ou adicionar camadas de desligamento (*dropout*). Essa camada de desligamento terá neurônios desligados aleatoriamente durante o funcionamento da rede. Como observado em [19], é possível que na etapa de treinamento as redes neurais consigam se adequar ao ruído dos dados, o que diminui o erro dos dados do treinamento mas aumenta de sobremaneira o erro dos dados no qual a rede neural não foi otimizada, isto é, o seu objetivo último que é realizar previsões. Em outras palavras, em superajuste o modelo simplesmente decora os dados do treinamento, prejudicando a sua capacidade de generalização, ou seja, de obter bons resultados para valores não treinados. De forma semelhante ao *dropout*, também aplicou-se um ruído multiplicativo gaussiano sobre cada *batch* do conjunto de treinamento.

Uma entrada auxiliar que consiste na codificação unária do número do mês também foi utilizada. Dessa forma, uma representação dos meses de Janeiro a Dezembro foi concatenada a cada entrada que as redes neurais recebiam, tomando-se a devida atenção para que o ruído multiplicativo gaussiano aplicado nas vazões não fosse também inserido nessa entrada. Observou-se que essa entrada auxiliar foi capaz de reduzir significativamente o erro absoluto médio da rede, que mudou de cerca de 29 para aproximadamente 23 no caso da usina de Camargos, para o conjunto de validação. A utilização do modelo multivariado foi capaz de reduzir esse erro para aproximadamente 31.

Assim como em [20], utilizaram-se tantos os modelos LSTM e GRU na caracterização da série temporal, comparando-os e também tal qual [15], utilizou-se uma série multivariada, embora com todos os valores, sem a necessidade de uma nova célula para lidar com valores faltantes.

Em uma primeira modelagem de MLP, utilizou-se apenas 4 entradas com uma camada oculta de 16 neurônios. Já no segundo modelo MLP utilizaram-se 12 entradas, com 4 camadas ocultas, sendo que as duas primeiras apresentaram *dropout* de 10%, com respectivamente 24, 12, 12 e 6 neurônios. O modelo recorrente utilizou a mesma estrutura, com unidades recorrentes. Os modelos de LSTM e GRU foram construídos com uma primeira camada oculta de suas respectivas unidades, com 12 neurônios cada, e outras 2

camadas MLP com 12 e 6 neurônios respectivamente. A partir dessas definições iniciais, os modelos neurais tiveram os seus hiperparâmetros ajustados de forma a melhorar o seu desempenho no conjunto de validação.

Ainda como forma de aproveitar a maior capacidade que os modelos de aprendizado profundo (*deep learning*) LSTM e GRU apresentam, adicionaram-se também os dados climáticos referentes à precipitação e à evaporação nas cidades próximas à Bacia do Paraná. Dessa forma, estações meteorológicas das seguintes cidades foram utilizadas: Belo Horizonte, Caldas, Campos do Jordão, Lavras, Machado, Resende, São Carlos e São Lourenço. Os dados foram disponibilizados pelo INMET (Instituto Nacional de Meteorologia) e correspondem a um período desde janeiro de 1961. Apesar de diminuir o período considerado, o aumento da dimensionalidade propicia uma diminuição das métricas de erro tendo em vista a influência dos eventos atmosféricos passados sobre os fluxos pluviais futuros.

Dessa forma, fez-se uma evolução do conjunto dos dados de entrada para o treino dos modelos. Inicialmente, apenas a série individual da vazão do rio (entrada univariada); já no segundo momento os dois rios foram simultaneamente alimentados às redes (entrada multivariada); como terceira forma, inseriu-se também uma entrada de dimensão 12 referente ao mês; para a quarta e última análise, utilizaram-se dados meteorológicos, aumentando-se a entrada em mais 16 dimensões referentes à evaporação e precipitação em 8 cidades.

IV. Resultados e Análises

Executaram-se os modelos em um computador Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz, no sistema operacional CentOS 7, uma distribuição Linux. A execução de todos os modelos durou cerca de 1 minuto, já o seu treinamento levou cerca de 15 minutos.

A divisão do conjunto dos dados se deu de maneira diferente entre os modelos treinados. Em um primeiro momento, foram construídas janelas temporais de tamanho 12 para os modelos *feedforward* enquanto que os recorrentes tinham a disponibilidade de uma sequência de comprimento 72. Estes realizaram a retroprogação apenas no último passo de sua simulação, isto é, a leitura de uma sequência para predizer apenas um único valor.

Os dados foram divididos em um conjunto de treinamento, validação e teste (verificação final). O conjunto de verificação final foi separado como os últimos dez anos da série temporal, ficando isolado da simulação para evitar a contaminação dos modelos por hiperparâmetros. O restante dos dados foi dividido entre treinamento e 20% para validação, avaliando-se se os modelo estavam em superajuste. Assim, a saída esperada para cada mês se deu com a entrada dos meses anteriores, isto é, a previsão de apenas um mês à frente.

É importante destacar com relação aos modelos recorrentes que cada caso de treinamento é uma matriz cujas linhas são instantes temporais e as colunas são os diferentes atributos. A fim de otimizar o tempo de treinamento, as sequências tiveram um tamanho fixo, possibilitando matrizes de dimensões iguais (com 72 linhas) para todos os casos de treino e a construção de um tensor de ordem 3 para os *mini-batches*, que são um agrupamento dos casos de treinamento.

É importante notar que a utilização da métrica ERM como o objetivo do treinamento condiciona os modelos a apresentarem estimativas cuja média seja ligeiramente menores do que a média dos dados reais, uma vez que um erro de superestimativa é mais penalizado que subestimativas. Outra observação interessante é que tanto os modelos LSTM quanto GRU tiveram uma elevada correlação, inclusive realizando as mesmas subestimativas nos meses de janeiro.

A Fig. 6 mostra as estimativas obtidos pela rede neural LSTM em comparação com a GRU e os dados reais. Observa-se que ambos os modelos recorrentes profundos conseguiram capturar a dinâmica do sistema, sobretudo para o período seco, que compreende os seis meses de maio a outubro. O compartilhamento dos pesos na dimensão temporal permite que o treinamento seja mais eficiente, isto é, a própria arquitetura da rede guarda informação sobre a estruturação dos dados. Os resultados comparativos das diferentes métricas dos modelos estão dispostos na Tabela I.

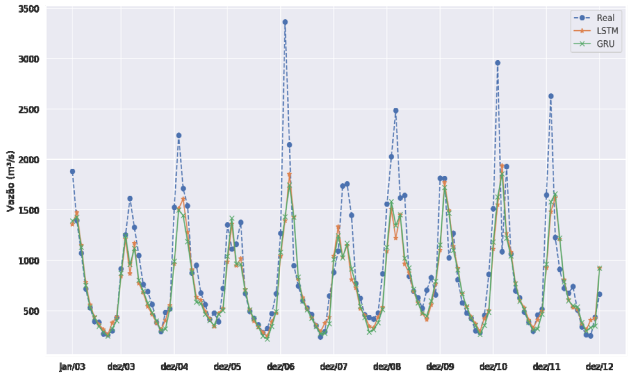

Fig. 6. Resultado para a previsão de Furnas utilizando-se LSTM em amarelo, GRU em verde e as vazões reais em azul.

TABELA I
RESULTADOS DO TESTE EM FURNAS PARA OS MODELOS

Modelo de estimativa	ERM_{Seco}	ERM	$ERM_{Úmido}$	R^2
ARIMA	17,5	22,1	26,7	0,69
Perceptron	12,9	18,5	24,0	0,67
MLP	13,0	19,3	25,7	0,64
Recorrente Simples	14,3	19,0	23,6	0,63
LSTM	13,0	19,1	25,2	0,61
GRU	13,0	18,9	24,9	0,63
D. B. De Lima [2]	-	19,5	-	-
B. O. Brito [8]	9,0	15,9	22,8	-

A Fig. 7 ilustra os resultados para a previsão da vazão da hidroelétrica de Camargos utilizando-se a arquitetura recorrente GRU com uma janela temporal de 60 meses como entrada. Observou-se um resultado semelhante a LSTM, apesar de que o modelo GRU tenha menos parâmetros a serem treinados uma vez que há apenas duas portas (*gates*) em detrimento de três.

A Tabela II apresenta os resultados comparativos para a vazão em Camargos, cuja vazão estimada é mais precisa do que a de Furnas, apesar de tal fenômeno ser menos significante nos seis meses úmidos, isto é, de novembro a abril.

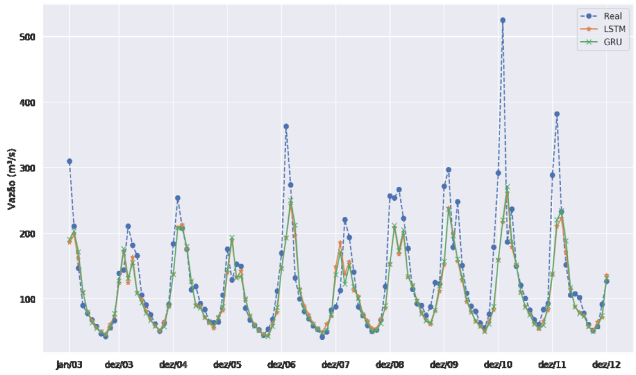

Fig. 7. Resultado para a previsão de Camargos utilizando-se LSTM em amarelo, GRU em verde e as vazões reais em azul.

TABELA II
RESULTADOS DO TESTE EM CAMARGOS PARA OS MODELOS

Modelo de estimativa	ERM_{Seco}	ERM	$ERM_{Úmido}$	R^2
ARIMA	11,5	17,5	23,5	0,70
Perceptron	10,0	17,2	24,3	0,60
MLP	10,9	17,7	24,5	0,59
Recorrente Simples	12,1	17,8	23,6	0,60
LSTM	9,8	17,0	24,2	0,59
GRU	9,2	16,4	23,5	0,60
D. B. De Lima [2]	-	16,9	-	-
B. O. Brito [8]	8,3	15,3	22,3	-

Assim, as métricas empregadas neste trabalho foram o erro relativo médio (ERM) apresentado na Equação 11.a e o coeficiente de determinação R^2 na Equação 11.b. Tais métricas foram relativas à quantidade N de amostras, o valor estimado $\hat{y}$, do valor real y e da média das amostras $\bar{y}$.

$$ERM(\%) = \frac{100}{N}\sum_{i=0}^{N}\left|\frac{\hat{y}_i - y_i}{y_i}\right| \quad (11.a)$$

$$R^2 = 1 - \sum_{i=0}^{N}(\hat{y}_i - y_i)^2 \Big/ \sum_{i=0}^{N}(y_i - \bar{y})^2 \quad (11.b)$$

Observou-se que o estimador estatístico ARIMA superou os modelos neurais no treinamento univariado e multivariado com a utilização apenas nas séries de vazão, mas apresentou um maior erro relativo médio em comparação com os modelos treinados com dados de maior dimensionalidade, conforme observado na Tabela II. Tal estimador foi calibrado com uma ordem (2, 0, 0), retirando-se a componente sazonal de período 12 utilizando-se os parâmetros (2, 0, 2, 12), encontrados por uma busca do menor erro de validação por meio da biblioteca Pmdarima.

Contudo, mesmo uma regressão linear com os últimos 12 meses das vazões de ambos os rios, representado por um único neurônio (perceptron com função de ativação tangente hiperbólica) conseguiu superar esse resultado. Ainda assim, o modelo de múltiplas camadas (MLP) não teve um desempenho superior aos modelos recorrentes, cuja arquitetura favorece o processamento de séries temporais.

De semelhante modo, os modelos de redes com mais parâmetros conseguiram capturar melhor a dinâmica à medida que a quantidade de dados aumentava, conforme observado na Tabela III. As colunas de meses e clima se referem respectivamente ao enriquecimento dos dados com as informações temporais e meteorológicas. Embora os modelos LSTM e GRU tenham apresentado desempenhos significativamente superiores tanto no treino quanto na avaliação sobre o conjunto de validação, sua performance no conjunto de teste foi aquém do esperado tanto por sua maior propensão à superajuste no caso do treino e pela seleção do modelo com menor erro na validação, que gera uma medida enviesada da performance no conjunto de validação, dado um espaço amostral de cerca das 500 épocas (*epoch*) para cada modelo, isto é, uma passada completa pelo conjunto de treino.

TABELA III
COMPARAÇÃO ENTRE O ERM DE FURNAS ENTRE TIPOS DE DADOS

Modelo de estimativa	Univariado	Multivariado	Meses	Clima
Perceptron	21,5	22,5	19,3	18,5
MLP	21,5	20,7	19,5	19,3
Recorrente Simples	19,5	20,4	19,6	19,0
LSTM	19,3	20,5	19,3	19,1
GRU	19,4	20,4	18,7	18,9

Nota-se que a utilização dos modelos propostos da última análise foi capaz de reduzir o erro relativo médio a um nível comparável aos trabalhos de [2] e [8], principalmente pela adição de mais dados, além da escolha das variáveis de entrada, como a sinalização dos meses e a utilização de mais de uma vazão.

V. Considerações Finais

As redes neurais oferecem várias vantagens em relação às abordagens convencionais. O aspecto mais importante é a capacidade de desenvolver uma solução generalizada para um problema a partir de um determinado conjunto de exemplos [1]. Assim, ocorre a combinação e adaptação desses modelos neurais a diferentes circunstâncias, com exposição a novas variações no problema por meio de um retreinamento, tal como ocorrido nas quatro análises.

Neste trabalho, observou-se que os modelos de redes GRU e LSTM foram superiores às redes MLP, que por sua vez também superaram as estimativas estatísticas mais comumente utilizadas, como o modelo ARIMA que se limita a frequências fixas. Contudo, isso se mostrou evidente apenas com a utilização de uma maior quantidade de dados. Tal fato se revela principalmente pelas correlações que as portas de reinício (*reset gates*) para a GRU e de esquecimento (*forget*) e entrada (*input*) para a LSTM aprendem a partir dos dados. Nas primeiras épocas de treinamento ocorreu um ajuste dessas portas que se assemelhou a um modelo linear responsável pela redução do viés de treino que, nas épocas seguintes, começou a reconhecer características não-lineares resultando na melhoria do erro. Todavia, devido à limitação do conjunto, há estruturas que não resultam de um padrão verdadeiro mas de ruídos que o modelo complexo começou a aprender e que

prejudicariam a sua capacidade de generalização caso o treinamento não fosse interrompido.

A utilização de um modelo multivariado para a previsão simultânea da vazão de duas hidroelétricas não mostrou um incremento estatisticamente significativo na melhoria do desempenho. Dinâmicas de mais longa duração como a detecção de ocorrência dos fenômenos *El Niño* e *La Niña* poderiam ser melhores levadas em consideração em modelos de mais longo prazo, como no caso das GRU e LSTM em contraste com os modelos clássicos, utilizando-se mais dados. Verificou-se a importância que a não imposição de janelas temporais têm no desempenhos das redes GRU e LSTM, que puderam realizar o seu treinamento em uma sequência muito mais longa de dados, e, além disso, nas etapas de validação e verificação que eram a continuidade da sequência temporal.

A utilização de ferramentas de código aberto possibilitou a reprodução do modelo proposto em outras máquinas por outros pesquisadores. O código e os dados que garantem a reprodutibilidade desse artigo estão disponíveis em https://github.com/Gabrui/papers/.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES). Agradecemos à Fundação Casemiro Montenegro Filho (FCMF) e ao Instituto Tenológico de Aeronáutica (ITA), por meio da Força Área Brasileira (FAB).

REFERÊNCIAS

[1] D. N. Kumar, K. S. Raju, and T. Sathish. “River Flow Forecasting using Recurrent Neural Networks”. *Water Resources Management*, vol. 18, pp. 143–161, 2004.

[2] D. B. De Lima, M. D. C. E Lima, and R. M. Salgado. “An Empirical Analysis of MLP Neural Networks Applied to Streamflow Forecasting”. *IEEE Latin America Transactions*, vol. 9, no. 3, June 2011.

[3] A. M. Kalteh, “Monthly river flow forecasting using artificial neural network and support vector regression models coupled with wavelet transform”, *Computers and Geosciences*, vol. 54, pp. 1–8, 2013.

[4] J. Zhang, Y. Zhu, X. Zhang, M. Ye, and J. Yang, “Developing a Long Short-Term Memory (LSTM) based model for predicting water table depth in agricultural areas”, *Journal of Hydrology*, vol. 561, pp. 918–929, 2018.

[5] C. V. Cardoso, and G. L. Cruz, “Forecasting Natural Gas Consumption using ARIMA Models and Artificial Neural Networks”, IEEE *Latin America Transactions*, vol. 14, no. 5, May 2016.

[6] R. L. U. Cazarez, C. L. Martín, “Neural Networks for predicting student performance in online education”, *IEEE Latin America Transactions*, vol. 16, no. 7, July 2018.

[7] J. A. Jaramillo, J. D. Velásquez, C. J. Franco, “Research in Financial Time Series Forecasting with SVM: Contributions from Literature”, *IEEE Latin America Transactions*, vol. 15, no. 1, Jan. 2017.

[8] B. O. Brito, R. M. Salgado, and L. A. Beijo, “Intelligent Modeling for Streamflow Forecasting”. *IEEE Latin America Transactions*, vol. 14, no. 8, Aug. 2016.

[9] S. Hochreiter, J. Schmidhuber, “Long short-term memory”, *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[10] J. Chung, C. Gulcehre, K. H. Cho, Y. Bengio, “Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling”. NIPS 2014 Deep Learning and Representation Learning Workshop.

[11] C. Olah, et al., “The Building Blocks of Interpretability”, Distill, 2018.

[12] M. A. Nielsen, “Neural Networks and Deep Learning”, Determination Press, 2015. Available in: <http://www.neuralnetworksanddeeplearning.com/>.

[13] I. Goodfellow, Y. Bengio, A. Courville, “Deep Learning”. MIT Press, 2016. Available in: <http://www.deeplearningbook.org/>.

[14] J. Jiménez, K. Donado, and C. G. Quintero, A Methodology for Short-Term Load Forecasting”. *IEEE Latin America Transactions*, vol. 15, no. 3, Mar. 2017.

[15] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu, “Recurrent Neural Networks for Multivariate Time Series with Missing Values”. *Scientific Reports*, vol. 8, 2018.

[16] R. Fonseca, P. Gómez, “Automatic Model Selection in Ensembles for Time Series Forecasting”, *IEEE Latin America Transactions*, vol. 14, no. 8, Aug. 2016.

[17] F. Chollet, et al., *Keras*. 2015. Available in: <http://keras.io/>.

[18] D. P. Kingma, Jimmy Ba, *Adam: A Method for Stochastic Optimization*. 3rd International Conference for Learning Representations, San Diego, 2015.

[19] T. V. da Silva, R. V. A. Monteiro, G. C. Guimarães, F. A. M. Moura, M. R. M. C. Albertini, and M. A. Tamashiro, “Performance Analysis of Neural Network Training Algorithms and Support Vector Machine for Power Generation Forecast of Photovoltaic Panel”, *IEEE Latin America Transactions*, vol. 15, no. 6, JUNE 2017.

[20] R. Fu, Z. Zhang, and L. Li, *Using LSTM and GRU Neural Network Methods for Traffic Flow Prediction*. 31st Youth Academic Annual Conference of Chinese Association of Automation, Wuhan, China, Nov. 2016.

Gabriel Adriano de Melo possui Graduação em Engenharia de Computação pelo Instituto Tecnológico de Aeronáutica (ITA, 2019) sendo mestrando em Engenharia de Computação pela mesma instituição. Possui experiência em desenvolvimento de aplicativos web, banco de dados e tem interesse em visão computacional. Atualmente pesquisa redes neurais no processamento de linguagem natural.

Dylan Nakandakari Sugimoto possui Graduação em Engenharia de Computação pelo Instituto Tecnológico de Aeronáutica (ITA, 2019). Possui experiência em programação de interfaces desktop e web, com ênfase em banco de dados. Atualmente trabalha com redes definidas por *software* (SDN) com testes de segurança.

Paulo Marcelo Tasinaffo Possui Graduação em Engenharia Mecânica pela Universidade Federal de Itajubá (UNIFEI, 1996), Mestrado também em Engenharia Mecânica pela Universidade Federal de Itajubá (UNIFEI, 1998) e Doutorado em Engenharia e Tecnologia Espaciais pelo Instituto Nacional de Pesquisas Espaciais (INPE, 2003). Atualmente é professor titular do Instituto Tecnológico de Aeronáutica (ITA). Tem experiência na área de Engenharia Aeroespacial e da Computação, com ênfase em

Redes Neurais Artificiais, atuando principalmente nos seguintes temas: modelagem de sistemas dinâmicos não-lineares, matemática computacional, inteligência artificial, agentes inteligentes, sistemas especialistas, sistemas de controle neural, computação evolutiva e processos estocásticos.

Afonso Henriques Moreira Santos Possui graduação em Engenharia Elétrica (1978) e Mestrado em Engenharia Mecânica (1981) pela Universidade Federal de Itajubá. Doutor em Planejamento de Sistemas Energéticos (1987) pela Universidade Estadual de Campinas e Pós-Doutor (1991) no Centre International de la Recherche sur l'Environnement et le développement (CIRED), na cidade de Paris – França, além de contar com 31 anos de experiência na área de Energia. Foi diretor da Agência Nacional de Energia Elétrica de 1997 a 2000.

Adilson Marques da Cunha possui: Bacharelado no Curso de Formação de Oficiais Aviadores pela Academia da Força Aérea - AFA - Brasileira (1970); Bacharelado em Administração de Empresas pelo Centro de Ensino Unificado de Brasília (1979); Mestrado em Information Systems pelo United States Air Force Institute of Technology - AFIT (1984); e Doutorado em Information Systems pela United States George Washington University - GWU (1987). Atualmente, é Professor Titular da Divisão de Ciência da Computação no Instituto Tecnológico de Aeronáutica – ITA.

Luiz Alberto Vieira Dias possui graduação em Engenheiro Eletricista pela Pontifícia Universidade Católica do Rio de Janeiro (1966), Mestrado em Ciência Espacial e da Atmosfera pelo Instituto Nacional de Pesquisas Espaciais (1968), Mestrado em Space Sciences - Rice University (1971) e Doutorado (PhD) em Space Physics and Astronomy - Rice University (1973). Atualmente é Professor Colaborador do Instituto Tecnológico de Aeronáutica - ITA.