



# Trabalho de Conclusão de Curso

## *MultiTask Learning* para previsão de Vazão

**Gabriel Dias de Abreu**

JUIZ DE FORA

NOVEMBRO, 2019

# *MultiTask Learning* para previsão de Vazão

Gabriel Dias de Abreu

Universidade Federal de Juiz de Fora
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Orientador: Luciana Conceição Dias Campos

JUIZ DE FORA
NOVEMBRO, 2019

# *MultiTask Learning* para previsão de Vazão

Gabriel Dias de Abreu

MONOGRAFIA SUBMETIDA AO CORPO DOCENTE DO INSTITUTO DE CIÊNCIAS EXATAS DA UNIVERSIDADE FEDERAL DE JUIZ DE FORA, COMO PARTE INTEGRANTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM CIÊNCIA DA COMPUTAÇÃO.

Aprovada por:

---

Luciana Conceição Dias Campos
Doutor

---

Leonardo Goliatt da Fonseca
Doutor

---

Heder Soares Bernardino
Doutor

JUIZ DE FORA
12 DE NOVEMBRO, 2019

*"Deep Learning is a superpower. With it you can make a computer see, synthesize novel art, translate languages, render a medical diagnosis, or build pieces of a car that can drive itself. If that isn't a superpower, I don't know what is.*

*Andrew Ng.*

# Resumo

A previsão da vazão em rios é fundamental para a manutenção do bem-estar social, dado que essas bacias fornecem desde recursos hídricos até energéticos e podem ainda causar graves tragédias como inundações e secas. Assim, prever a vazão de longo prazo, em estações de medição de uma bacia hidrográfica, com boa precisão, contribuem com a solução de uma miríade de problemas que afetam a sociedade e o gerenciamento de recursos.

O presente trabalho propõe o modelo MultiTask-LSTM que alia o modelo recorrente de *Deep Learning*, LSTM, com a transferência de aprendizagem *MultiTask Learning*, para prever e compartilhar informações adquiridas em toda a bacia do rio. Esse método é robusto a dados faltantes e ruídos, que são problemas comuns em séries temporais de vazão, e em específico, deste problema. O MultiTask-LSTM, quando aplicado nas 45 estações de medição do Rio Paraíba do Sul, conseguiu superar os modelos LSTM, que alcançaram os melhores resultados da literatura. Os experimentos utilizam três formas diferentes de imputação de dados faltantes para confirmar a robustez do MultiTask-LSTM a ruídos, que alcançou resultados estáveis mesmo variando a forma de imputar dados faltantes.

Por fim, o trabalho introduz um modelo que relaciona os dados presentes em todas as séries temporais das estações de medição da bacia, robusto a dados faltantes e ruídos, com treinamento mais rápido e melhor desempenho se comparado aos modelos LSTM.

**Palavras-chave:** Transferência de Aprendizagem, *Deep Learning*, séries temporais, previsão de Vazão.

# Abstract

Predicting river flow is critical to maintaining social well-being, as these watersheds provide from water to energy resources and can also cause severe tragedies such as floods and droughts.

Thus, predicting long-term flow at watershed metering stations with good accuracy helps us to solve a myriad of problems affecting society and resource management.

The present work proposes the MultiTask-LSTM model which combines the recurrent Deep Learning LSTM model with the MultiTask Learning transfer to predict and share information acquired across the river basin. This method is robust to missing data and noise, which are common problems inflow time series, and specifically of this problem. MultiTask-LSTM, when applied in the 45 measuring stations of Paraíba, do Sul River was able to surpass the results obtained in LSTM models, which obtained the best results in the literature. The experiments use three different ways of imputing missing data to confirm the robustness of the noisy MultiTask-LSTM, which achieved stable results even by varying how to impute missing data.

Finally, the paper introduces a model that relates the data present in the all-time series of basin measuring stations, robust to missing data and noise, with faster training and better forecasting performance compared to LSTM models.



**Keywords:** MultiTask Learning, Deep Learning, Time Series, River Flow Forecast.

# Agradecimentos

Aos meus pais pelo apoio e dedicação, ao meu irmão pelos ensinamentos que fizeram essa conquista possível.

A minha companheira de vida e profissão, sempre ao meu lado em todos os momentos, agradeço pelas inúmeras horas de trabalho e paciência, Letícia.

Aos professores do Departamento de Ciência da Computação pelos seus ensinamentos.

Ao professor Andrew Ng, pela sua grande iniciativa educacional, muito além de qualquer barreira sócio-educacional, que culminou nesse e em muitos outros trabalhos.

# Conteúdo

# Lista de Figuras

# Lista de Tabelas

# Lista de Abreviações

| | |
|---|---|
| DCC | Departamento de Ciência da Computação |
| UFJF | Universidade Federal de Juiz de Fora |
| LSTM | Long Short-Term Memory |
| ARIMA | Autoregressive Integrated Moving Average |
| MTL | Multi Task Learning |
| STL | Single Task Learning |

# 1 Introdução

## 1.1 Apresentação do tema

A previsão da vazão se faz necessária devido à dependência e fixação das sociedades nos entornos de bacias hidrográficas ao longo da história, uma vez que atividades fundamentais para a sociedade, como agricultura, pecuária, saneamento básico, geração de energia hidrelétrica, indústria e turismo são baseadas na disponibilidade hídrica, o que implicou no desenvolvimento de técnicas para identificar e prever o comportamento dessas bacias. Aliado a isso, tragédias podem ser evitadas como as decorrentes de enchentes, secas, rompimento de barragens e vetores de doenças [Yassuda, 1993].

Por uma ótica atual de sociedade, o aperfeiçoamento dessas técnicas é consoante com a melhoria na gestão dos recursos hídricos e a preservação ambiental, impactada negativamente pela acelerada expansão urbana, possibilitando o desenvolvimento sustentável e viabilizando a tomada de decisão e planejamento de risco de longo prazo por órgãos competentes [Rezende et al., 2013].

Os registros históricos contidos em séries temporais de fenômenos hídricos são muitas vezes custosos e difíceis de aferir, além de apresentarem ruídos e dados faltantes, o que prejudica o desempenho da previsão dessas séries temporais [Herschy, 2014].

Em específico, a bacia do rio Paraíba do Sul com 45 estações de medição de vazão possui muitos dados faltantes em todas as estações oriundos do desligamento da estação ou afins. Além de modificações hidro-geomorfológicas ou até falhas em sensores que resultam em ruídos nas séries temporais.

As variáveis hidro-geomorfológicas presentes em uma bacia, apresentam variações correlatas que indicam possíveis eventos, tais como alterações nos registros mensurados por uma estação de medição de vazão a montante, que influenciam na previsão da série temporal de uma estação de medição a jusante[1]. Portanto, faz-se necessário considerar

[1]Jusante é o fluxo normal da água, de um ponto mais alto para um ponto mais baixo. Montante é a direção de um ponto mais baixo para o mais alto. A jusante é o lado para onde se dirige a corrente de água e montante é a parte onde nasce o rio. Por isso, diz-se que a foz de um rio é o ponto mais a jusante

esses fenômenos para melhorar a capacidade preditiva. Por exemplo, caso uma barragem esteja instalada em uma região da bacia do rio, todo o fluxo a jusante dessa barragem será afetado, então as previsões das séries temporais de estações a jusante da barragem precisam considerar esse fenômeno.

As séries temporais de vazão são muito suscetíveis a fatores exógenos, como manutenções que exigem desligamento da estação de medição e falhas de medição em sensores. Somado a isso, as relações presentes nas séries temporais distribuídas ao longo do bacia do rio, quando não aproveitadas devidamente, configuram um potencial renegado para a previsão e desperdício dos recursos gastos para as medições de vazão. Logo, o estudo de previsão com métodos robustos a dados faltantes e/ou com ruídos em séries temporais de vazão se faz necessário.

## 1.2 Motivação

A bacia do Rio Paraíba do Sul tem grande importância no desenvolvimento econômico brasileiro e abastece 32 milhões de pessoas [Kelman, 2015]. Ela possui 45 séries temporais de vazão captadas em estações de medição com dados faltantes e ruídos que dificultam a previsão.

*MultiTask Learning* é uma abordagem para transferência de aprendizagem indutiva que aumenta a generalização fazendo uso da informação de tarefas relacionadas. Isso é feito pelo aprendizado em paralelo usando uma representação compartilhada. Ou seja, o que é aprendido em cada tarefa pode ajudar na melhoria do aprendizado das outras como definido em [Caruana, 1997].

O método de *MultiTask Learning* consegue ser resiliente aos dados faltantes e ruídos uma vez que, considera as relações presentes nas séries temporais de vazão ao longo da bacia do rio. Com isso, dados faltantes ou ruídos que prejudicariam o desempenho do modelo têm seu efeito negativo diminuído pelas relações presentes nos dados, unindo o aprendizado de cada série temporal em um único modelo.

O método de transferência de aprendizagem *MultiTask Learning* ainda consegue capturar informações implícitas nas relações entre todas as séries temporais de vazão ao

deste rio, e a nascente é o seu ponto mais a montante.

longo da bacia do rio, proporcionando melhor uso dos dados disponíveis em relação a aplicação dos modelos de previsão separadamente em cada estação de medição.

A motivação desse trabalho consistiu em aliar essas características da transferência de aprendizagem *MultiTask Learning* ao modelo *LSTM* de redes neurais recorrentes, que na literatura apresenta resultados promissores nas aplicações de previsão de vazão [Campos et al., 2019] para fazer a previsão de vazão de séries temporais da bacia do rio Paraíba do Sul, que apresenta 45 estações de medição com dados faltantes e ruidosos em suas séries temporais.

## 1.3 Trabalhos Relacionados

A previsão de séries temporais de vazão é muito utilizada para o planejamento e administração dos recursos hídricos como evidencia o trabalho em [Yaseen et al., 2015] que apresenta os modelos clássicos como os *ARIMA* e Regressão Linear, que são incapazes de capturar a não estacionariedade e não linearidade das séries temporais hidrológicas. Esse estudo ainda aponta o crescimento da atenção dada aos modelos direcionados a dados como as redes neurais que apresentam progresso na previsão de séries temporais não lineares, capturando a complexidade das séries temporais hídricas.

O *LSTM* é uma arquitetura específica de rede neural recorrente que possui a capacidade de aprender dependências temporais de longo prazo e ser robusto a ruídos. Isso o torna eficiente em problemas de previsão de séries temporais de recursos hídricos como explorado nos trabalhos em [Kratzert et al., 2018] que mostrou a eficácia do modelo *LSTM* em alternativa a modelos complexos que incluem conhecimentos prévios sobre o comportamento das afluências e o estudo em [Zhang et al., 2018] que mostrou a capacidade do *LSTM* de prever a profundidade de águas para irrigação a longo prazo, assim contribuindo com a gestão das águas para irrigação. Porém, os dois trabalhos deixam claro a necessidade de uma considerável quantidade de dados para o *LSTM* apresentar resultados satisfatórios.

O trabalho em [Campos et al., 2019] mostrou a eficiência do modelo *LSTM* para a previsão de vazão na bacia do Rio Paraíba do Sul quando comparado a outros modelos clássicos como o ARIMA e ainda apontou a importância da previsão de vazão de longo

prazo nessa bacia, e esse trabalho utilizou um subconjunto de 4 das 45 estações de medição de vazão do rio Paraíba do Sul.

O trabalho em [Shireen et al., 2018] mostrou que modelos utilizando *MultiTask Learning* conseguem captar informações de varias séries temporais ao mesmo tempo, com robustez a dados faltantes e ruídos, realizando inferências acerca de todas as séries temporais e suas relações no âmbito de painéis fotovoltaicos.

## 1.4 Objetivos

O objetivo principal do trabalho foi propor um modelo robusto a dados faltantes e ruídos e que consiga melhorar a previsão da vazão de longo prazo utilizando informações presentes nas séries temporais das estações de medição localizadas ao longo da bacia do rio Paraíba do Sul.

Em segundo plano, o trabalho alia técnicas de *Deep Learning*, como o *LSTM*, com a transferência de aprendizagem *MultiTask Learning*, para gerar métodos de previsão robustos a dados faltantes e/ou com ruídos que aproveitam as relações implícitas entre várias séries temporais, que no estudo de caso desse trabalho pertencem à mesma bacia hidrográfica.

## 1.5 Organização do Trabalho

O trabalho está organizado da seguinte forma, o Capítulo 2 oferece referencial teórico para que o leitor tenha maior entendimento do trabalho, o Capítulo 3 apresenta o modelo desenvolvido aplicando as técnicas de *MultiTask Learning* a um modelo de *Deep Learning* para fazer previsão de vazão, o Capítulo 4 apresenta o estudo de caso onde o modelo foi aplicado nas séries temporais de vazão coletadas nas estações de medição ao longo da bacia do Rio Paraíba do Sul. O Capítulo 5 apresenta a conclusão e os trabalhos futuros sugeridos.

# 2 Fundamentação Teórica

## 2.1 Série Temporal

Série Temporal é uma coleção de observações sequenciais feitas ao longo do tempo, dita contínua, quando as observações ocorrem em instantes irregulares de tempo, ou dita discreta, quando as observações são tomadas em intervalos iguais de tempo [Chatfield, 2016], por exemplo, séries temporais de sensores onde um sensor emite um sinal a cada minuto com a temperatura.

Quando a série temporal pode ser exatamente prevista, ela é determinística. Por outro lado, a maior parte das séries temporais são estocásticas, ou seja, componentes aleatórios influem no comportamento da série temporal, tornando previsões exatas impossíveis, sendo necessário modelos probabilísticos que tentem aproximar o comportamento da melhor maneira possível [Chatfield, 2016]. As séries temporais têm quatro componentes principais que são dissociáveis das observações, sendo eles [Adhikari and Agrawal, 2013]:

• Tendência, que é um movimento de longo prazo podendo ser de estagnação, crescimento ou decrescimento;

• Variações sazonais, onde existem padrões causados por efeitos externos que se repetem ao longo do tempo;

• Variações cíclicas configuradas por mudanças de médio prazo circunstanciais que se repetem;

• Variações irregulares decorrentes de fenômenos aleatórios imprevisíveis;

Quando as propriedades estatísticas como média e variância de uma série temporal não dependem do tempo, temos uma série estacionária. No entanto, as séries temporais que apresentam tendência central ou sazonalidade, como na Figura 2.1, não são estacionárias. Porém, algumas transformações de potenciação e diferenciação podem ser aplicadas em uma série temporal não estacionária para torná-la estacionária [Adhikari and Agrawal, 2013], vide Seção 2.2.4.

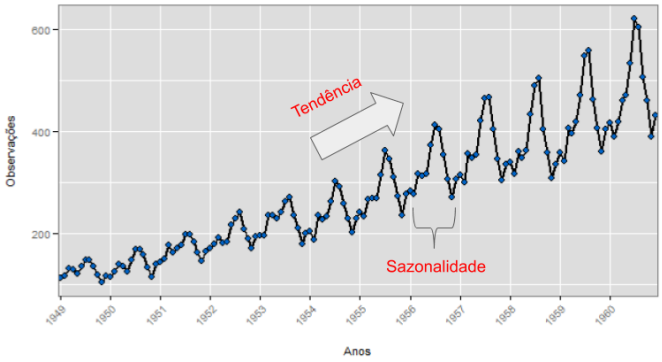


Figura 2.1: Representação da Tendência e sazonalidade - autor

## 2.2 Modelos ARIMA

Os modelos da família ARIMA são os principais métodos estatísticos utilizados para a previsão de séries temporais de fluxo na literatura [Yaseen et al., 2015]. Esses métodos possuem algumas vantagens como a linearidade e interpretabilidade.

### 2.2.1 Autoregressive Model (AR)

O modelo AR, da família ARIMA, expressa uma série temporal estocástica de forma finita com uma agregação linear dos últimos $p$ valores anteriores da série temporal e uma parte estocástica $\alpha_t$. Assim, tendo uma série temporal estocástica discreta denotada por $Z$, temos um modelo AR descrito pela equação 2.1:

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + ... + \phi_p Z_{t-p} + \alpha_t \tag{2.1}$$

onde p é a ordem do modelo, $\alpha_t$ é uma variável aleatória, os elementos $\phi_1, \phi_2, \ldots, \phi_p$ são os coeficientes da combinação linear e $Z_t$ é a observação obtida no tempo $t$ a partir das suas realizações passadas $Z_{t-1}$ até $Z_{t-p}$. Esse modelo é genericamente representado por ARIMA(p, 0, 0) [Box et al., 2015].

### 2.2.2 Moving Average Model (MA)

O modelo MA, da família ARIMA, também expressa uma série temporal Z estocástica, a partir da média ponderada dos erros aleatórios de $q$ períodos passados, representada pela equação 2.2:

$$Z_t = \alpha_t - \Theta_1\alpha_{t-1} - \Theta_2\alpha_{t-2} - ... - \Theta_q\alpha_{t-q} \tag{2.2}$$

onde o parâmetro $q$ indica a quantidade de valores de erros regressores, $\alpha_t, \ldots, \alpha_{t-q}$ são valores de erros nos tempos $t, t-1, ...t-q$, $\Theta_1, \ldots, \Theta_q$ são os parâmetros do modelo e $Z_t$ é a série temporal no tempo $t$. Esse modelo é genericamente representado por ARIMA(0, 0, q).

### 2.2.3 Autoregressive-Moving Average Model (ARMA)

O modelo ARMA, da família ARIMA, é a junção dos dois modelos descritos nas Seções 2.2.1 e 2.2.2, podemos defini-lo como:

$$Z_t = \phi_1 Z_{t-1} + \phi_2 Z_{t-2} + ... + \phi_p Z_{t-p} + \alpha_t - \Theta_1\alpha_{t-1} - \Theta_2\alpha_{t-2} - ... - \Theta_q\alpha_{t-q} \tag{2.3}$$

onde os parâmetros são obtidos da soma do modelo AR e o modelo MA[Box et al., 2015].

### 2.2.4 Autoregressive Integrated Moving Average Model (ARIMA)

Muitas das series temporais estocásticas que descrevem fenômenos reais são não estacionárias e, em particular, a média varia com o tempo. Nesses casos, definimos o operador $w_t$ [Box et al., 2015] que executa $d$ diferenciações na série temporal estocástica como:

$$w_t = (1 - B)^d Z_t \tag{2.4}$$

assim, representamos um modelo com comportamento não estacionário, por um modelo que faz um número $d$ de diferenciações para alcançar o comportamento estacionário. Contudo, encontram-se valores de $d$ como 0, 1 e 2 e note que $d = 0$ implica que já temos comportamento estacionário e $B$ é o operador de *backward shift* onde $B^n(Z_t) = Z_{t-n}$

definindo uma notação enxuta para realizações passadas da série temporal estocástica.

O Modelo ARIMA [Box et al., 2015], é definido como:

$$w_t = \Phi_1 w_{t-1} + \Phi_2 w_{t-2} + ... + \Phi_p w_{t-p} + \alpha_t - \Theta_1 \alpha_{t-1} - \Theta_2 \alpha_{t-2} - ... - \Theta_q \alpha_{t-q} \qquad (2.5)$$

sendo $w_t$ o valor da série temporal $Z_t$ diferenciada $d$ vezes.

## 2.3 Redes Neurais

O termo Rede Neural Artificial tem sua origem ao esforços matemáticos de simular o processamento de informação no cérebro humano que é altamente complexo, não-linear e paralelo, organizado em neurônios, com a capacidade de fazer generalizações, abstrações e reconhecer padrões [Haykin et al., 2009].

Os neurônios artificiais recebem como entrada $x_1 \dots x_m$ que são linearmente combinadas no $\Sigma$ com os pesos definidos por $w_{i,1}, w_{i,2} \dots w_{i,m}$ e o bias $\theta_i$. E esses pesos são responsáveis por codificar o aprendizado do neurônio. Uma função de ativação é aplicada resultando na saída $\hat{y}_i$ como representado na Figura 2.2

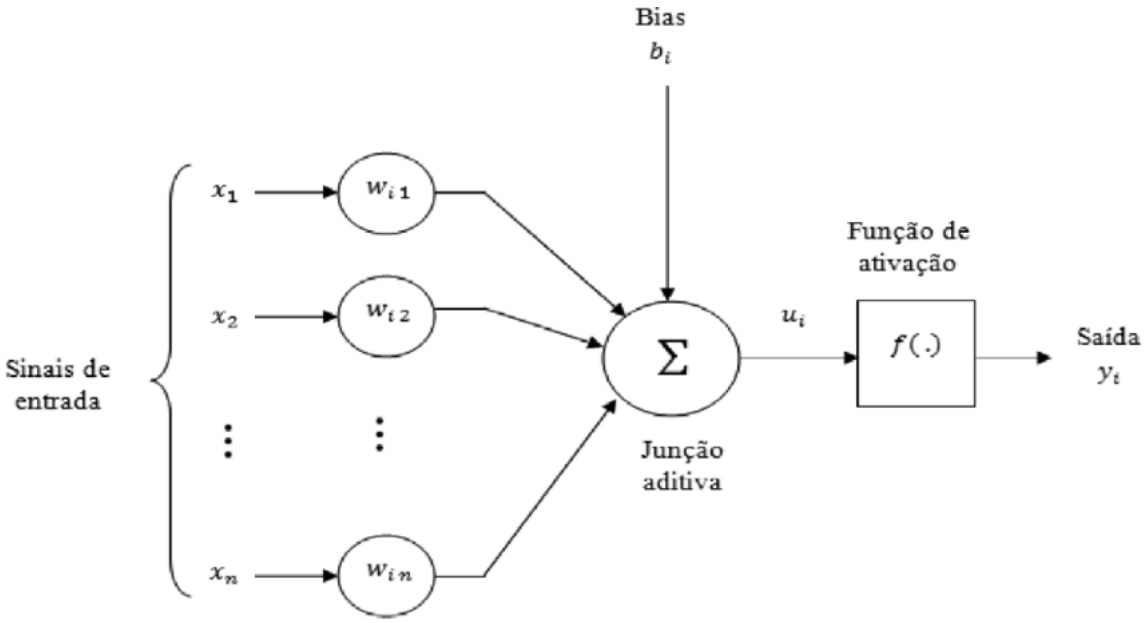


Figura 2.2: Neurônio artificial [Haykin et al., 2009]

Uma formulação matemática do neurônio artificial é apresentada na equação 2.6.

$$\hat{y}_i = \varphi(\sum_{j=1}^{m} \omega_{i,j} x_j + \theta_i) \tag{2.6}$$

onde $x_j$ representa a j-ésima componente de x, $\omega_{i,j}$ o peso sináptico que liga o sinal de entrada $x_j$ ao neurônio $i$, $\theta_i$ é o *bias* e $\varphi$ é uma função de ativação.

Na literatura, existem diversas funções de ativação $\varphi$ disponíveis, como por exemplo:

- A função de ativação linear, definida pela equação 2.7.

$$\varphi(v) = v \tag{2.7}$$

  onde $v$ representa a combinação linear resultante de $\Sigma$ apresentada na Figura 2.2.

- As funções de ativação sigmóides: função logística e função tangente hiperbólica, respectivamente definidas nas equações 2.8 e 2.9.

$$\varphi(v) = \sigma(v) \tag{2.8}$$

$$\varphi(v) = tanh(v) \tag{2.9}$$

- A função de ativação ReLU é definida pela equação 2.10. A ReLU produz ativação não-linear apesar de ser muito próxima de uma função linear, o que é uma propriedade interessantes para a generalização e a otimização com métodos baseados em gradiente.

$$\varphi(v) = max(0, v) \tag{2.10}$$

### 2.3.1 Rede Neural Feed-Forward

As redes neurais *Feed-Forward* são redes neurais artificiais em que os dados vão sempre adiante, sem voltas ou recorrências. A Figura 2.3 apresenta uma arquitetura de uma rede neural *Feed-Forward*, contendo 4 sinais de entrada na camada de entrada, 3 neurônios na camada oculta ou intermediária e 2 neurônios na camada de saída. Observe

que os pesos sinápticos conectam os elementos de uma camada com a outra e nunca elementos da mesma camada, ou retornam para camadas anteriores.

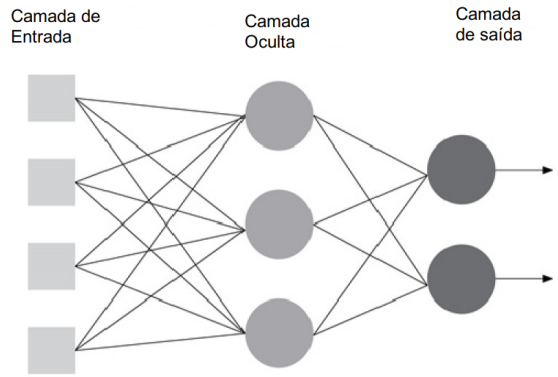


Figura 2.3: Rede neural Feed-forward com uma camada oculta [Haykin et al., 2009]

Essas redes neurais podem ser representadas como composições de diferentes funções, onde cada uma dessas funções é chamada de camada, formando uma função final descrita na equação 2.11, similar ao representado na Figura 2.3.

$$f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x))). \tag{2.11}$$

onde cada $f^n$ representa a camada n da rede neural e é composta por um ou mais neurônios. A função $f^{(1)}$ é chamada camada de entrada (*input layer*) enquanto a camada $f^{(n)}$ é chamada camada de saída (*output layer*), todas as demais camadas são chamadas de camadas ocultas (*hidden layers*) [Goodfellow et al., 2016].

As redes neurais Feed-Forward são o fundamento de *Deep Learning* e quando devidamente configuradas executam a função de aproximadoras para quaisquer funções [Goodfellow et al., 2016].

O aprendizado nessas redes neurais se dá de forma supervisionada onde dado um conjunto de dados de entrada, aqui denominados de $X$ e a saída denominada $Y$. Os pesos são ajustados em função desses exemplos $(x, y)$, na maioria das vezes utilizando alguma variante do algoritmo *Back-Propagation*, que oferece uma forma eficiente de calcular os gradientes [Haykin et al., 2009].

O algoritmo back-propagation é baseado na aprendizagem por correção de erro, onde a entrada $x$ é fornecida para a rede neural e a fase de propagação acontece da forma

apresentada na equação 2.11, e a partir da saída $\hat{y}$, produzida pela rede neural, podemos definir o erro como $y - \hat{y}$. Esse erro é retro-propagado na rede neural e utilizado para corrigir os pesos, com o intuito de obter erros menores. Cada iteração desse algoritmo é denominado época que se repete até um critério de parada ocorrer. Mais detalhes do funcionamento desse algoritmo são apresentadas em [Haykin et al., 2009, Goodfellow et al., 2016, Bishop, 2006, Goodfellow et al., 2016].

Um dos problemas enfrentados no aprendizado por gradiente, em especial usando *back-propagation*, é o *vanishing gradient*, que tem como principal consequência tornar difícil para o algoritmo de otimização definir um caminho de melhora para diminuir o erro. Esse problema está associado ao uso de funções de ativação como as sigmóides na Figura 2.4 que tem um comportamento como apresentado na figura, onde os valores saturam quando a entrada fica muito grande ou muito pequena. Essa característica é um empecilho, principalmente em modelos *Deep Learning*, onde um número maior de camadas pode diminuir o efeito da correção de erro até que seja ínfimo e não ocorra variação significativa dos pesos. A função de ativação *ReLU*, por outro lado não sofre desse problema [Goodfellow et al., 2016].

### 2.3.2 Rede Neural Recorrente

O funcionamento de uma rede neural recorrente é semelhante a uma *Feed-forward*, com a diferença de acrescentar como entrada da camada recorrente os resultados da saída dessa mesma camada no passo anterior [Graves, 2012], como pode ser observado na Figura 2.5, onde a saída da camada oculta é realimentada para uma outra camada anterior. Essa arquitetura de rede neural produz uma formulação profunda de compartilhamento de pesos [Goodfellow et al., 2016] como pode ser visto na Figura 2.6, em que a rede neural recorrente é desdobrada a cada passo da entrada.

Assim, essas redes neurais possuem memória, ou seja, mantém um estado dos resultados de passos passados sendo aplicáveis em problemas com dependência temporal, como em sequências. Esse tipo de rede neural obtém bons resultados em aplicações de processamento de linguagem natural, processamento de som, processamento de sinais, processamento de video, ente outras [Graves, 2012].

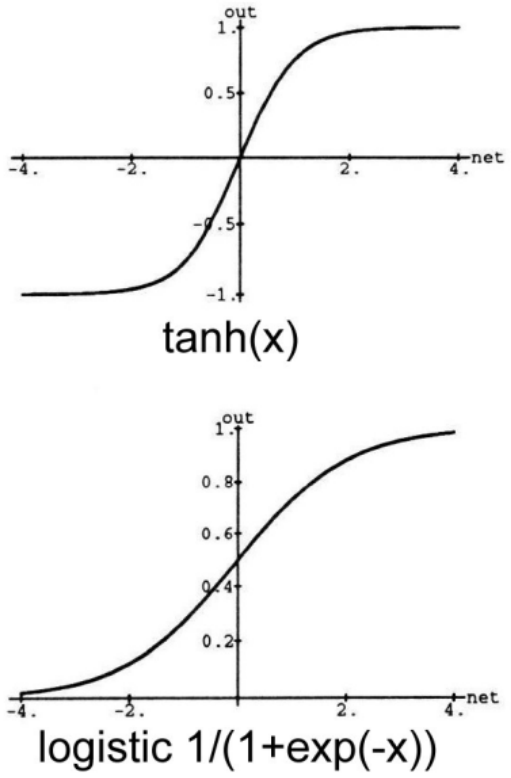


Figura 2.4: Comportamento das funções de ativação Sigmoides tangente hiperbólica e logística [Graves, 2012].

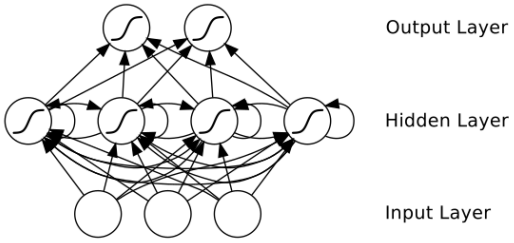


Figura 2.5: Exemplo de uma rede neural recorrente onde os neurônios da camada oculta tem sua saída realimentada [Graves, 2012].

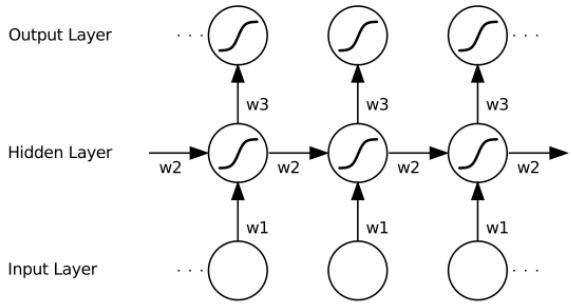


Figura 2.6: Exemplo de uma rede neural recorrente desdobrada em relação às entradas [Graves, 2012].

Essa abordagem de relaxar as restrições das redes neurais *Feed-Forward* para produzir Redes Neurais Recorrentes resulta em consequências poderosas como mapear um histórico de entradas anteriores em saídas, diferente das redes *feed-forward* que mapeiam uma entrada em um saída. Além disso, a aproximação universal de funções em redes *Feed-Forwards* também é válida em redes recorrentes, com a extensão de serem aproximadoras universais de funções que recebem uma sequência como entrada em saídas [Graves, 2012].

Uma rede neural recorrente tem como entrada um sequência denotada por $(x_1, x_2, ..., x_T)$ onde T é o tamanho da janela histórica, ou a ordem que o modelo dispõe do histórico passado de registros da série temporal, e uma saída denotada por $\hat{y}$. Redes neurais recorrentes podem ter como saída uma sequência denotada por $(\hat{y}_1, \hat{y}_2, ..., \hat{y}_T)$, porém, problemas de previsão univariada possuem somente uma saída $\hat{y}$ representando a previsão.

Uma camada recorrente, ou seja, um camada formada por neurônios recorrentes, podem ser descritas pelas equações 2.12 e 2.13

$$h(t) = \sigma(W1_{h,x}x(t) + W2_{h,h}h(t-1) + b_h) \tag{2.12}$$

$$\hat{y}(t) = \sigma(W3_{y,h}h(t) + b_y). \tag{2.13}$$

onde t é o passo temporal, $\sigma$ é uma função de ativação, $b_h$ é o bias no estado h, $W_{h,x}$ são os pesos no estado h da entrada x e $W_{h,h}$ são os pesos do estado h no estado h. A função h(t) representa o estado do neurônio recorrente no passo temporal t [Lipton et al., 2015].

Assim como as redes neurais *Feed-Forward*, as redes neurais recorrentes são comumente ajustadas usando uma variação do *back-propagation* [Lipton et al., 2015], o *backpropagation through time* (BPTT), que a partir da forma desdobrada apresentada na Figura 2.6, executa o ajuste dos pesos baseado no gradiente para cada passo temporal [Graves, 2012].

No entanto, o problema do *vanishing gradient*, presente nas redes neurais *Feed-Forward* é agravado em redes neurais recorrentes uma vez que sequências de sinais são realimentados e tem seu erro corrigido.

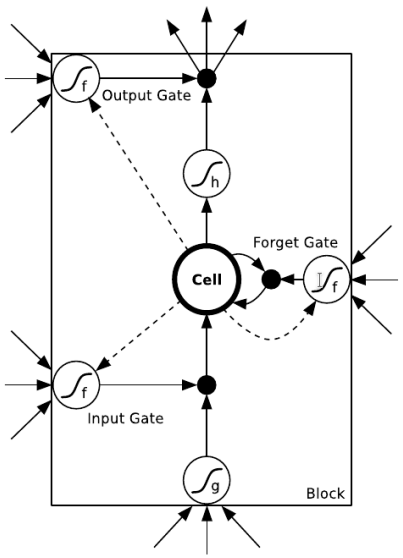


Figura 2.7: Uma célula do modelo LSTM [Graves, 2012]

### 2.3.3 Long Short-Term Memory (LSTM)

O *Long Short-Term Memory* acrescenta aos modelos recorrentes a capacidade de ignorar entradas irrelevantes e saídas armazenadas na memória interna irrelevantes, o que torna o LSTM robusto ao problema dos pesos diminuírem ao ponto da rede não conseguir se ajustar apropriadamente aos dados (*vanish gradient*) [Hochreiter and Schmidhuber, 1997].

Como apresentado em [Graves, 2012] e ilustrado na Figura 2.7, a arquitetura do LSTM é composta por um conjunto de sub-redes conectadas de forma recorrente, conhecidas como blocos de memória. Cada bloco consiste em uma ou mais células autoconectadas de memória e três unidades multiplicativas, que são os gates de entrada, gates de saída e gates de esquecimento, análogos a operações de escrita, leitura e reinicio.

Intuitivamente, o nome *Long short-term memory* foi inspirado na característica de que modelos simples de redes neurais recorrentes possuem memória de longo prazo na forma de pesos, que mudam pouco ao longo do treinamento, representando conhecimento geral sobre os dados, e possuem uma memória de curto prazo representada pelas ativações que passam de um nó para outro nó. No entanto, o LSTM introduz um tipo intermediário de memória na sua célula de memória **CELL** [Lipton et al., 2015], que pode ser visto na Figura 2.7.

Uma camada LSTM pode ser decomposta em *Gates*, uma particularidade in-

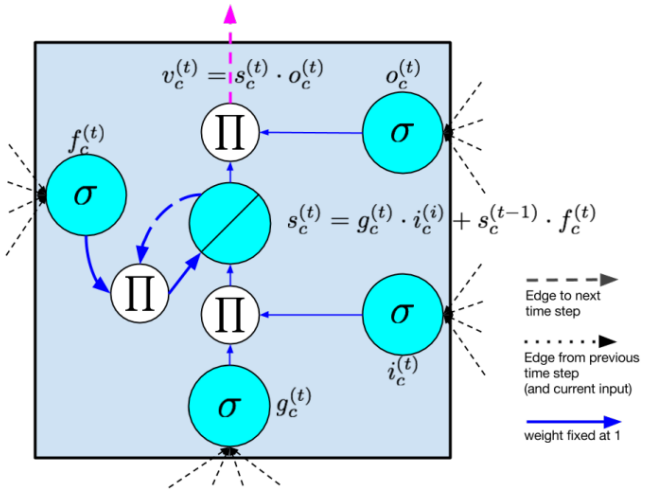


Figura 2.8: Uma camada LSTM detalhada [Lipton et al., 2015]

troduzida pelo LSTM, são unidades sigmoidais multiplicativas que funcionam como um *Input Node*, ou seja, ativam a entrada da informação que é fornecida a ele e o seu próprio estado no passo anterior. O termo *gate* vem de portão, e é assim denominado por barrar ou amenizar o sinal que passa por ali, por exemplo, um peso zero no gate significa que o sinal será impedido de propagar-se à frente.

O LSTM representado na Figura 2.8 é composto pelas seguintes unidades, segundo [Lipton et al., 2015]:

- *Input Node*: Essa unidade, denotada por $g_c$, é o nó que ativa a entrada $x_t$ e a recorrência da saída da camada oculta, assim como na rede recorrente comum[Lipton et al., 2015];

- *Input Gate*: Denotado por $i_c$, esse portão controla o sinal de entrada oriundo do *Input Node*;

- *Internal state*: Denotado por $s_c$, esse é o coração da célula de memória do LSTM que trata o problema do *vanishing gradient* em redes neurais recorrentes com uma entrada recorrente auto conectada de peso constante, fazendo com que o erro flua sem aumentar ou diminuir demais. Essa unidade é também chamada de carrossel de erro constante. Podendo-se denotar matematicamente pela equação 2.14.

$$s_c^{(t)} = g_c^{(t)} \odot i_c^{(t)} + s_c^{(t-1)} \tag{2.14}$$

onde $g_c^{(t)}$ é o *input node* no passo t e $i_t$ é o *input gate* e $\odot$ é o produto elemento a elemento ou *element-wise*;

- *Forget gate*: Denotado por $f_c$, esse portão tem a função de limpar o conteúdo do *Internal State* o que é essencial em sequências longas. Acrescentando-se o *forget gate*, o *Internal state* é descrito pela equação 2.15.

$$s_c^{(t)} = g_c^{(t)} \odot i_c^{(t)} + s_c^{(t-1)} \odot f_c^{(t)} \tag{2.15}$$

onde $g_c^{(t)}$ é o *input node* no passo t e $i_c^{(t)}$ é o *input gate* no passo t, $f_c^{(t)}$ é o *Forget gate* no passo t e $\odot$ é o produto elemento a elemento ou *element-wise*;

- *Output gate*: Denotado por $o_c$, esse portão tem a função de controlar o sinal que sai da celula LSTM. O $v_c$ produzido como saída final do célula de memoria LSTM é o valor do estado interno $s_c$ aplicado na função de ativação *tanh*, multiplicado pelo valor do *output gate* $o_c$;

### 2.3.4 Considerações sobre Deep Learning

Os modelos de *Deep Learning* são assim denominados por serem Redes Neurais Artificias com profundidade considerável, utilizando muitas camadas como definidas na Seção 2.3.1. Possíveis graças as melhorias algorítmicas e experimentais que diminuem o efeito do *Gradient Vanishing*, que é a aproximação dos sinais de correção do *backpropagation* a zero, o que torna os sinais de correção ínfimos, e a melhoria na capacidade de processamento do hardware [Goodfellow et al., 2016].

A vantagem de modelos *Deep Learning* sobre os outros modelos de *machine learning* é que a profundidade permite ao modelo aprender um programa de múltiplos passos onde cada camada pode ser entendida como o estado da memória do computador executando instruções paralelas. Partindo de cada camada o aprendizado de sinais cada vez mais complexos, até por fim aprender padrões comparáveis à capacidade de reconhecimento de padrões humana [Goodfellow et al., 2016]. No entanto, esses modelos conseguem esse desempenho quando possuem os recursos necessários; os modelos *Deep Learning* necessitam de muitos dados e estão alcançando resultados promissores graças à era do Big

Data, onde conseguimos processar muito mais dados e armazena-los [Goodfellow et al., 2016].

## 2.4 Transferência de aprendizagem

Transferência de aprendizagem é um conceito ligado ao ato de aproveitar um conhecimento aprendido para a realização de uma tarefa em uma outra tarefa. Por exemplo, considere que exista um modelo que consegue classificar felinos, podemos aproveitar o conhecimento prévio do modelo de classificação de felinos em um modelo cuja tarefa seja a classificação de gatos.

Suponhamos que um modelo de aprendizagem precisa cumprir duas ou mais tarefas diferentes, e que existam muitos fatores cuja variação na tarefa 1 influenciem na tarefa 2. Nesse caso, definimos uma aplicação de transferência de aprendizagem [Goodfellow et al., 2016]. Esse conceito é comumente aplicado no contexto de aprendizagem supervisionada onde a entrada é semelhante mas a saída é modificada [Goodfellow et al., 2016]. Como é ilustrado na Figura 2.9, o processo de transferência de aprendizagem mantém as

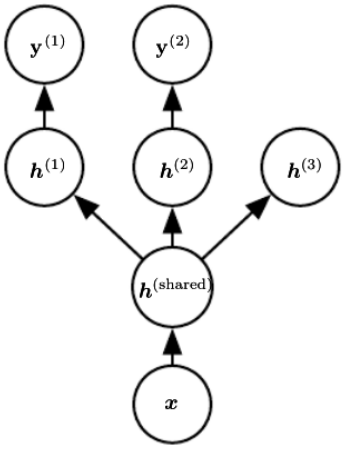


Figura 2.9: Representação compartilhada em um processo de transferência de aprendizagem [Goodfellow et al., 2016].

camadas de entrada e um camada escondida com representação compartilhada, de onde, o conhecimento previamente aprendido será utilizado. Assim, somente a parte final da rede neural é treinada para aprender as novas tarefas. Portanto, todo o treinamento feito posteriormente em $h^{(shared)}$ é aproveitado para contribuir em uma nova tarefa.

### 2.4.1 MultiTask Learning

O *MultiTask Learning (MTL)* é um mecanismo de transferência de aprendizagem cujo principal objetivo é aumentar o desempenho de generalização. Essa melhoria é alcançada com o compartilhamento de representação para as tarefas relacionadas e executadas paralelamente, como representado na Figura 2.11, ao contrário do representado na Figura 2.10 denominado *single task learning (STL)*, onde existe um modelo para cada tarefa [Caruana, 1997].

Assim, como aumentar o número de exemplos no treinamento de um modelo supervisionado, contribui com a capacidade de generalização. O *MultiTask Learning* através do aprendizado de várias tarefas relacionadas impõe restrições na representação da aprendizagem do modelo, de forma a aumentar a capacidade de generalização. Ou seja, o modelo fica mais restrito em direção a resultados melhores, uma vez que o compartilhamento de representação se justifique, através de tarefas relacionadas [Goodfellow et al., 2016].

Os modelos MTL podem ser divididos em duas partes em relação aos parâmetros associados [Goodfellow et al., 2016]:

- Parâmetros de uma tarefa específica, que se beneficiam somente dos exemplos específicos da sua tarefa para atingir a generalização. Esses são as camadas superiores na figura 2.9.
- Parâmetros genéricos, que são compartilhados por todas as tarefas, e que se beneficiam de todas os exemplos independente da tarefa, melhorando a generalização. Esses são as camadas inferiores na Figura 2.9.

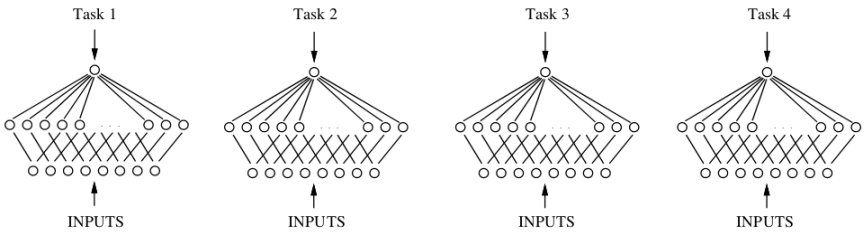


Figura 2.10: Quatro tarefas sem *MultiTask Learning* [Caruana, 1997]

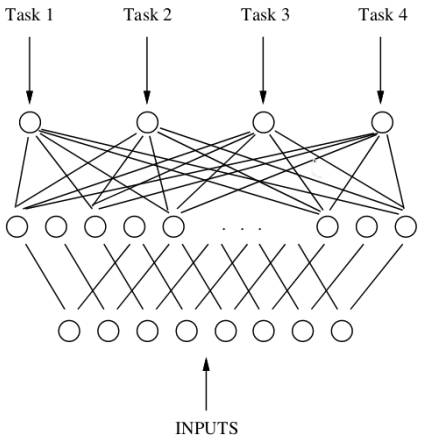


Figura 2.11: *MultiTask Learning* de quatro tarefas [Caruana, 1997]

# 3 MultiTask-LSTM

O modelo MultiTask-LSTM é composto por uma camada LSTM que alia as vantagens do LSTM como robustez a ruídos, tolerância ao problema de *vanishing gradient* e considerável performance de aprendizagem a partir dos dados históricos de séries temporais e a transferência de aprendizagem *MultiTask Learning* onde múltiplas tarefas são otimizadas ao mesmo tempo pelo modelo a fim de que a otimização alcançada em uma tarefa influencie positivamente a execução de outra tarefa.

O MultiTask-LSTM foi desenvolvido para o cenário de aplicação onde várias séries temporais de vazão estão dispostas ao longo da bacia capturando dados, como as séries temporais contém ruídos e dados faltantes e estão sujeitas a variações hidrogeomorfológicas como é comum no domínio de medição de vazão, as múltiplas fontes de dados serão consideradas no aprendizado da previsão. Esse modelo visa previsões de longo prazo.

O modelo funciona como ilustrado na Figura 3.1 onde um modelo LSTM prevê ao mesmo tempo as tarefas de $E$ estações de medição, compartilhando o aprendizado feito em cada tarefa, aumentando a capacidade de ignorar ruídos, pois mais fontes de informação são consideradas, a capacidade de generalizar o aprendizado, pois o modelo é obrigado a aprender relações que estão presentes em muitas séries e, por fim, capacidade de antecipar fenômenos que afetam a bacia como um todo como modificações na nascente.

Como pode ser observado na Figura 3.2, as $E$ séries temporais são fornecidas como entrada para o modelo, elas são divididas em janelas rolantes de tamanho $j$ e passo de tamanho 1, cada passo dessas séries temporais é concatenado com as $E$ estação de medição, formando uma matriz de $E$ linhas e $j$ colunas, e um vetor $y$ com tamanho $E$. Esses dados são então fornecidos ao LSTM que aprende a prever o comportamento futuro das séries temporais.

O modelo MultiTask-LSTM pode ser descrito como na equação 3.1:

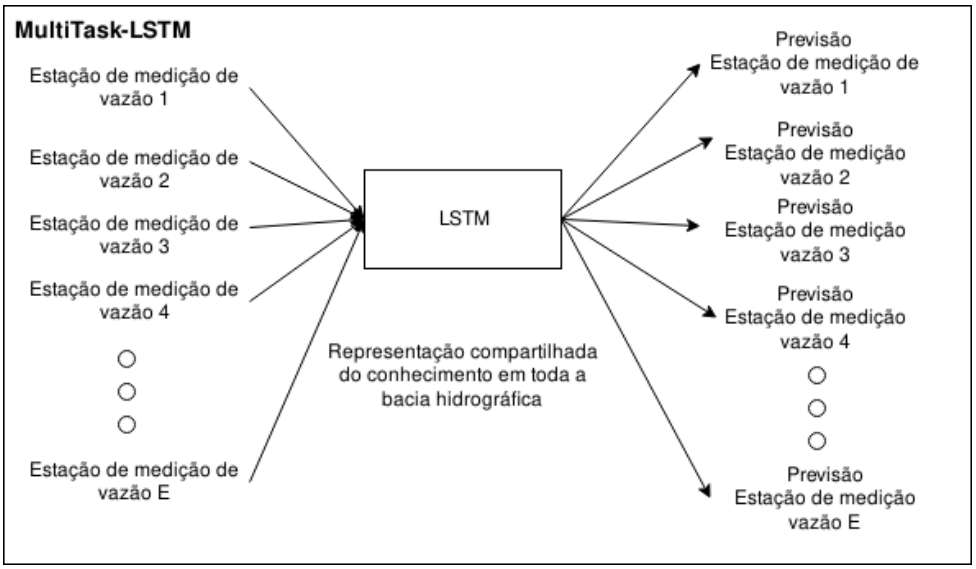


Figura 3.1: MultiTask-LSTM representado de forma simplificada - autor

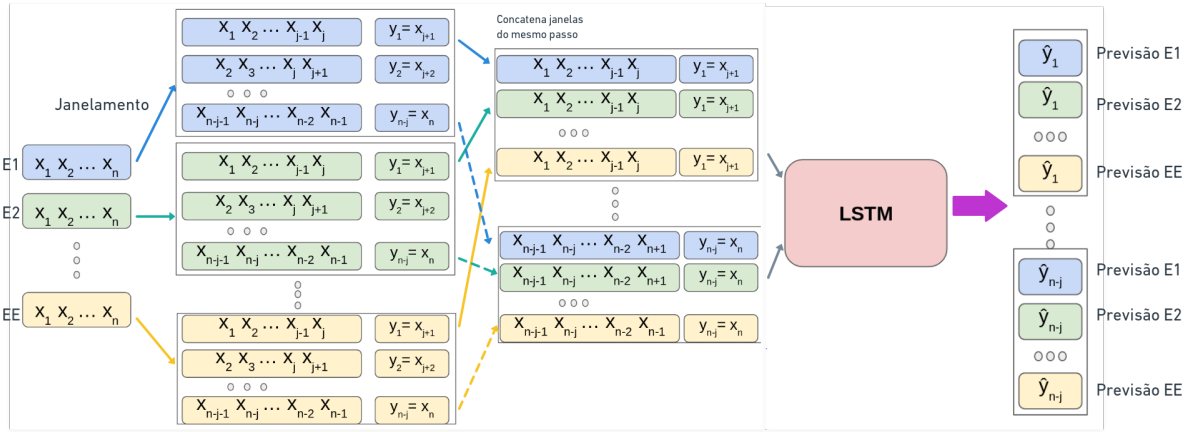


Figura 3.2: Modelo MultiTask-LSTM - autor

$$
\begin{aligned}
&\mathrm{Q}_{1,t+j} = F(Q_{1,t}, Q_{1,t-1}, \cdots, Q_{1,t-j-1}) \\
&\mathrm{Q}_{2,t+j} = F(Q_{2,t}, Q_{2,t-1}, \cdots, Q_{2,t-j-1}) \\
&\cdots \\
&\mathrm{Q}_{E,t+j} = F(Q_{E,t}, Q_{E,t-1}, \cdots, Q_{E,t-j-1})
\end{aligned}
\tag{3.1}
$$

onde $Q_{e,t+j}$ é a vazão na estação de medição de vazão $e$ e dia $t + j$, e $j$ é o tamanho da janela, e $F$ é a função estimada.

O componente LSTM teve o *kernel* inicializado com o algoritmo Xavier, as mini-batchs com tamanho 256 e 200 neurônios na camada oculta. Como a camada LSTM necessita de duas funções de ativação não lineares, elas foram determinadas como $hard-sigmoid$ e *hyperbolic tangent* ($tanh$) e a última camada foi definida como uma função de ativação linear. Esses hiper-parâmetros foram aplicados com sucesso em um subconjunto das estações de medição de vazão da bacia do Rio Paraíba do Sul no trabalho de [Campos et al., 2019].

Esse modelo foi otimizado pelo algoritmo ADAM com validação cruzada *hold-out* e *early-stopping* para prevenir *overfitting* [Campos et al., 2019].

# 4 Estudo de Caso

Os experimentos realizados na bacia do Rio Paraíba do Sul em suas 45 estações de medição para a comparação do modelo MultiTask-LSTM e LSTM são dispostos nesse capítulo.

Os experimentos foram executados no ambiente Google Colab[2] com 12gb de memória RAM em GPUs utilizando as bibliotecas Keras[3], numpy[4] e Tensorflow[5] em linguagem Python. Todos os resultados foram obtidos em relação a média de 30 execuções e a métrica MAPE foi a escolhida para comparar os resultados, definida por:

$$MAPE = \frac{100\%}{n} \sum_{t=1}^{n} \mid \frac{A_t - F_t}{A_t} \mid \tag{4.1}$$

onde $A_t$ é o valor histórico no tempo $t$, $F_t$ é o valor previsto no tempo $t$ e $n$ é o tamanho da série temporal.

O LSTM aplicado no modelo MultiTask-LSTM teve como hiper-parâmetros os demonstrados pelo trabalho de [Campos et al., 2019]. A utilização desses mesmos hiper-parâmetros ainda reforça a comparação ao LSTM quando executado de forma STL, ou seja, um modelo para cada série temporal das estações de medição.

O modelo MultiTask-LSTM utiliza janelas de 14 dias como no trabalho de [Campos et al., 2019], com 45 estações de medição e tem a forma apresentada na equação 4.2:

$$\mathrm{Q}_{1,t+14} = F(Q_{1,t}, Q_{1,t-1}, \cdots, Q_{1,t-13})$$
$$\mathrm{Q}_{2,t+14} = F(Q_{2,t}, Q_{2,t-1}, \cdots, Q_{2,t-13})$$
$$\cdots$$

[2] colab.research.google.com
[3] keras.io
[4] numpy.org
[5] www.tensorflow.org

$$Q_{45,t+14} = F(Q_{45,t}, Q_{45,t-1}, \cdots, Q_{45,t-13})$$

(4.2)

Um conjunto de treino com os primeiros 75% dos dados foi empregado para treinar o modelo, enquanto 10% dos dados seguidos foram usados no conjunto de validação para verificação dos hiper-parâmetros e 15% dos últimos dados da série temporal foram reservados para o teste. Cada experimento foi executado 30 vezes de onde calculamos a média da métrica MAPE para aferir o desempenho final do modelo.

## 4.1 Base de Dados

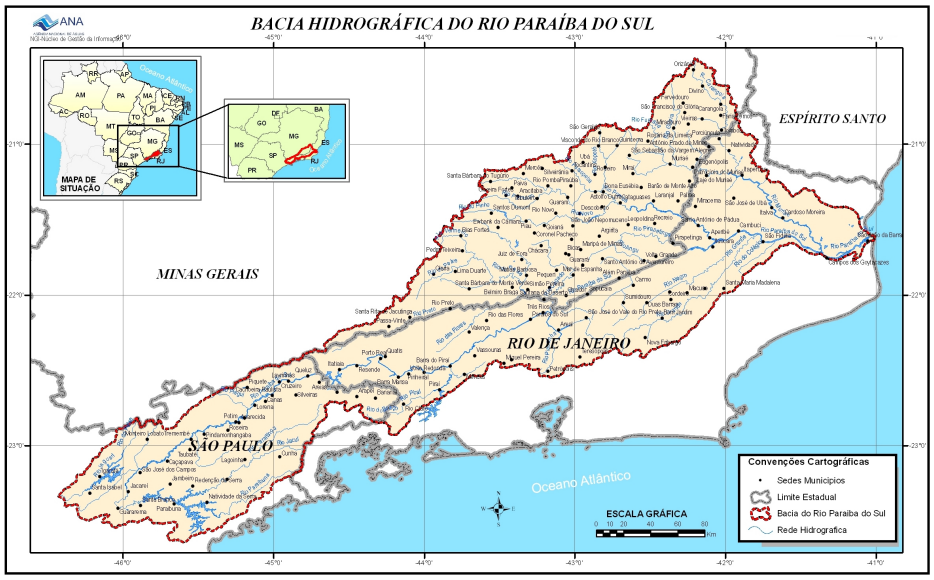


Figura 4.1: Imagem Bacia Rio Paraíba do Sul - ANA (www.ana.gov.br)

Os dados disponibilizados pela Agência Nacional de águas (ANA)[6], a base de dados contém 45 estações de medição de vazão ao longo da bacia do Rio Paraíba do Sul que pode ser vista na Figura 4.1, com registros diários no período de 1935 até 2016 e dados faltantes como pode ser observado na Figura 4.2. Um exemplo de série temporal pode ser visto na Figura 4.4.

[6] www.ana.gov.br

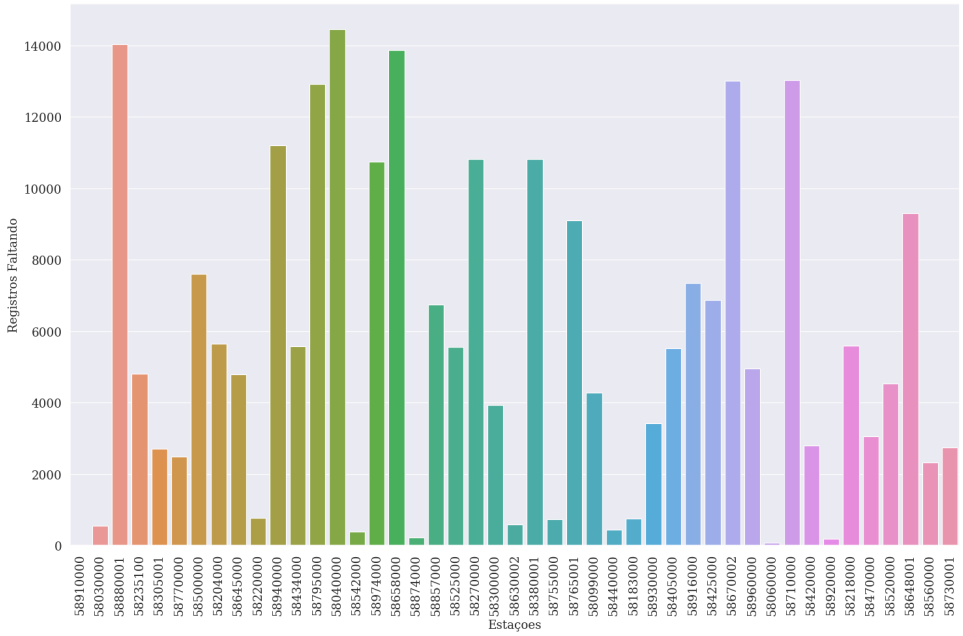


Figura 4.2: Quantidade de Dados Faltantes por estação de medição de vazão

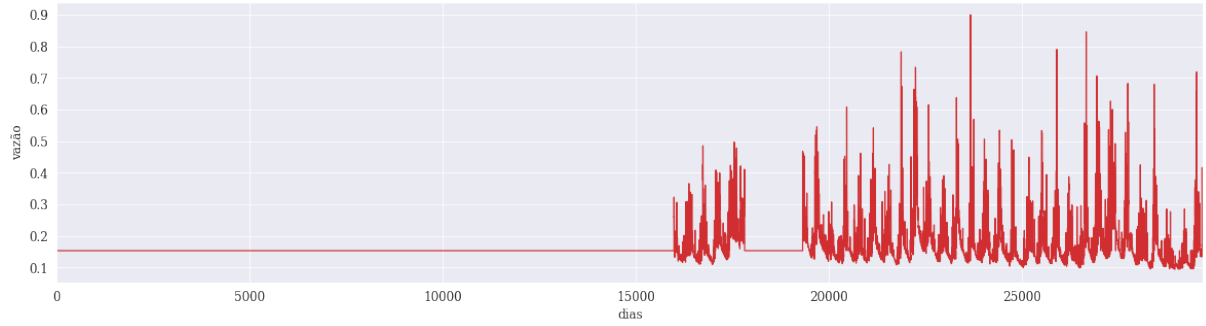


Figura 4.3: Série temporal da estação 58040000 com maior número de dados faltantes imputados pelas medianas

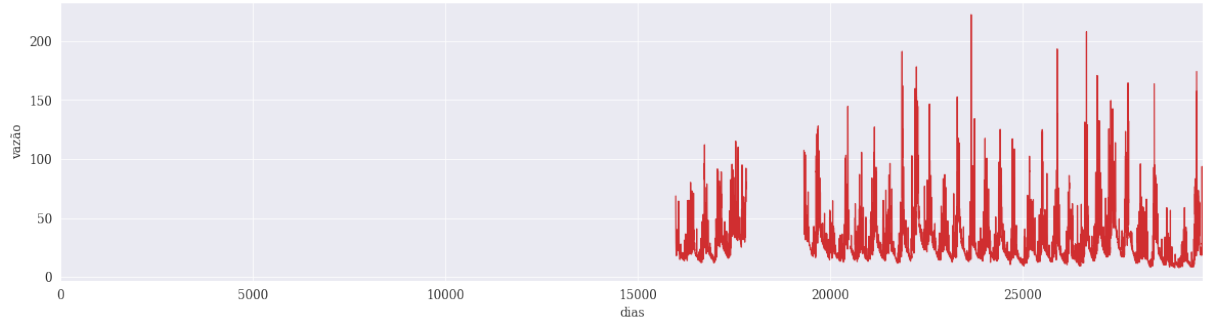


Figura 4.4: Série temporal da estação 58040000 com maior número de dados faltantes

Os dados faltantes e ruídos constituem um problema para a previsão de séries temporais, uma vez que ruídos implicam em erros no aprendizado do comportamento da série temporal, enquanto os dados faltantes inibem o modelo de entender o que aconteceu durante o período em que os dados não estão dispostos, e por isso, afetam a continuidade

da previsão do modelo. A quantidade de registros faltando por estação de medição de vazão pode ser observada na Figura 4.2.

Algumas técnicas foram aplicadas para imputar os dados faltantes, enquanto o ruído foi tratado com a aplicação do modelo MultiTask-LSTM, que é a combinação de duas técnicas robustas a ruídos, o *MultiTask Learning* e o LSTM. Pode-se ainda entender que o processo de imputação de valores nos dados faltantes cria um ruído que será tratado pelas características de aprendizagem por correlação de séries temporais semelhantes do MultiTask-LSTM.

Os Dados faltantes foram imputados com três técnicas diferentes ARIMA, Média dos dias em todos os anos e mediana da série.

### 4.1.1 Mediana

Nesta técnica, cada série temporal de vazão produzida pelas estações de medição tiveram sua mediana calculada e todos os dados faltantes daquela série temporal foram substituídos por ela, como pode ser visto na Figura 4.3.

O preenchimento dos dados faltantes com a mediana foi a técnica mais simples de imputação aplicada. Ela permite entendermos se o MultiTask-LSTM realmente consegue obter uma melhora em relação ao LSTM já que o ruído acrescentado pelo valor constante da mediana proporciona uma aproximação muito distante do real e assim, para obter bons resultados, o modelo MultiTask-LSTM é forçado a utilizar os dados aprendidos de séries temporais similares, e aprender a identificar e ignorar o ruído.

### 4.1.2 ARIMA

Nesta técnica de imputar os dados faltantes, o modelo ARIMA é aplicado aos dados de cada série temporal de vazão das estações de medição. Em seguida geram-se os dados faltantes com o modelo ARIMA construído e o resultado pode ser visto na Figura 4.5. Os hiper-parâmetros p, q e d são escolhidos por busca exaustiva como no trabalho de [Campos et al., 2019].

O ARIMA foi utilizado para preencher os dados faltantes, dado que é uma técnica clássica de previsão de séries temporais de vazão, ocasionando na diminuição do compo-

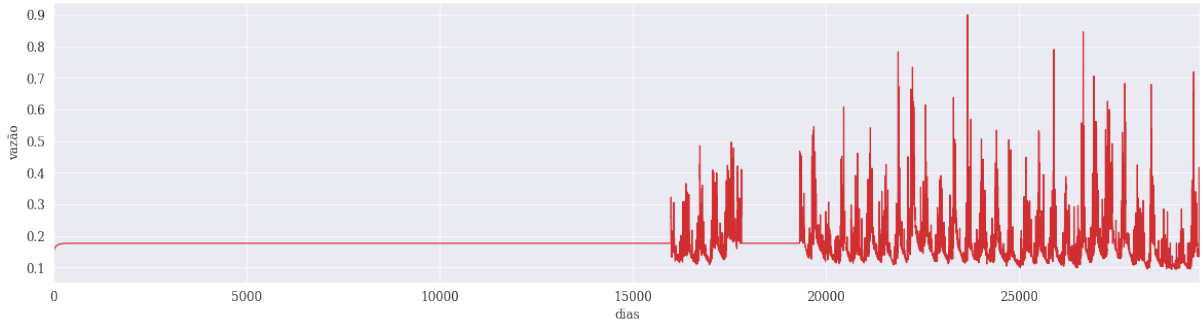


Figura 4.5: Série temporal da estação 58040000 com maior número de dados faltantes imputados pelo modelo ARIMA

nente de ruído na série temporal, o que implica em comparar o MultiTask-LSTM e o LSTM dando maior atenção a capacidade de extrair as informações geomorfológicas presentes nas séries temporais das estações de medição de vazão.

### 4.1.3 Média dos dias de cada Ano

Nesta técnica cada dado faltante é imputado pela média dos resultados obtidos para o respectivo dia e mês daquele dado faltante durante o ano. Essa forma de imputar os dados supõe que a forte sazonalidade anual presente nessa série temporal referente a fenômenos hídricos será determinante para os valores faltantes como pode ser visto na Figura 4.9.

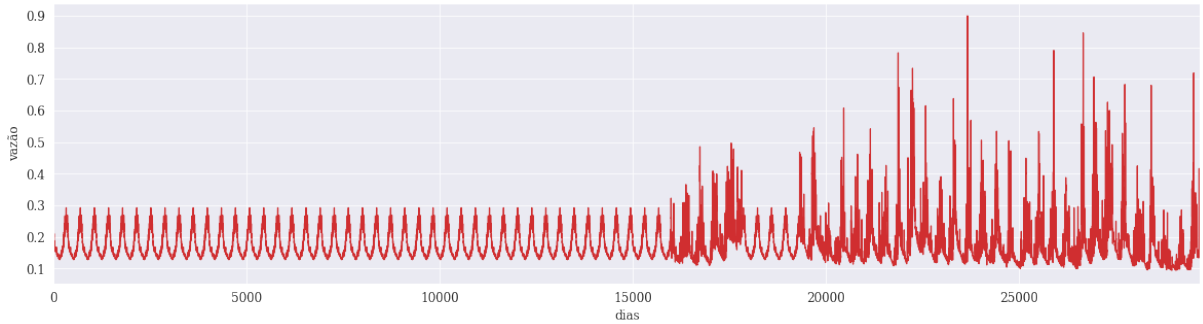


Figura 4.6: Série temporal da estação 58040000 com maior número de dados faltantes imputados pelas medias de cada dia dos anos.

O preenchimento dos dados faltantes com essa técnica representou bem a sazonalidade presente na série temporal e permitiu averiguar melhor o desempenho do LSTM em relação ao MultiTask-LSTM. Uma vez que a melhor imputação com ruídos menores e mais controlados tende a favorecer o LSTM sem transferência de aprendizagem MultiTask

e assim comparar a característica de melhor aproveitamento das relações geomorfológicas das séries temporais.

## 4.2 Experimentos

Como pode ser observado na Figura 4.7, O MultiTask-LSTM, quando aplicada a imputação por mediana, obtém resultados consideravelmente melhores na maioria das estações de medição a não ser pela 58218000. O que demonstra a capacidade do modelo MuliTask-LSTM de relacionar os dados hidrogeomorfólogicos presentes ao longo da bacia do Rio Paraíba do Sul para inferir em meio ao ruído acrescentado pela imputação da mediana (constante).

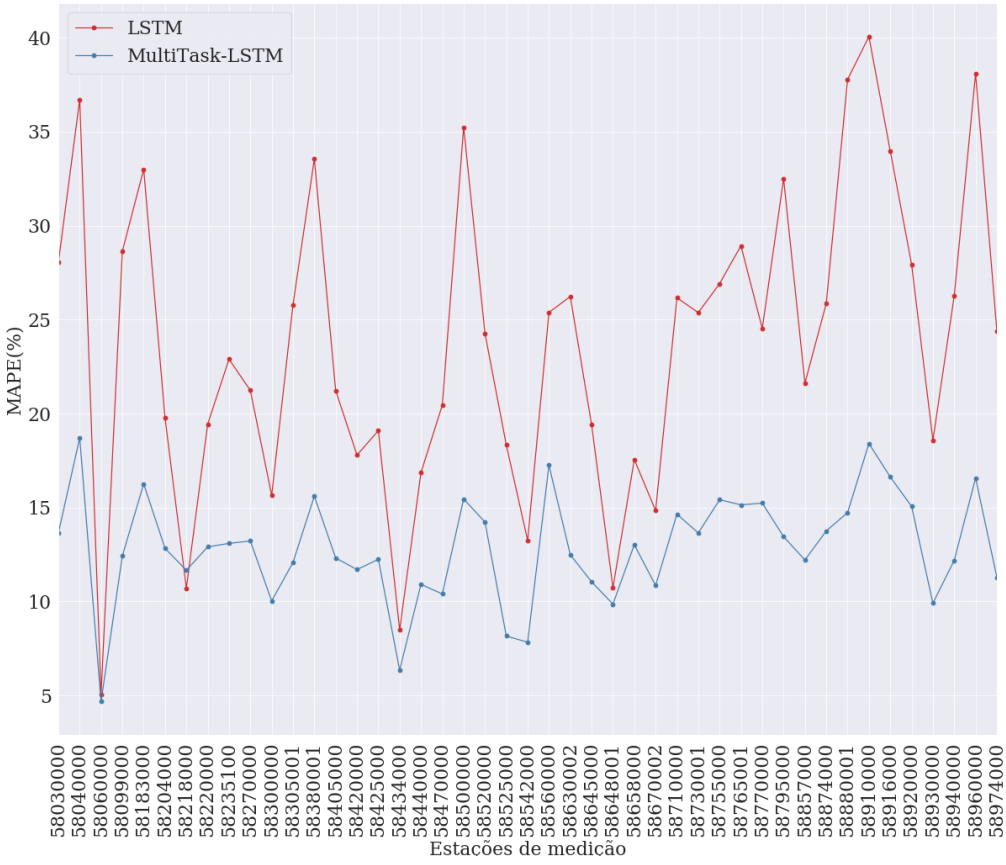


Figura 4.7: Comparação MultiTask-LSTM e LSTM nas 45 estações da bacia do RPS com mediana imputada

Como pode ser observado na Figura 4.8, muito semelhante ao exemplo da imputação com a mediana quando imputados os dados utilizando o modelo ARIMA. O

MultiTask-LSTM obtém resultados consideravelmente melhores em todas as estações de medição o que reforça a robustez do modelo MuliTask-LSTM de relacionar os dados hidro-geomorfólogicos presentes ao longo da bacia do Rio Paraíba do Sul para inferir em meio ao ruído acrescentado pela imputação do modelo ARIMA.

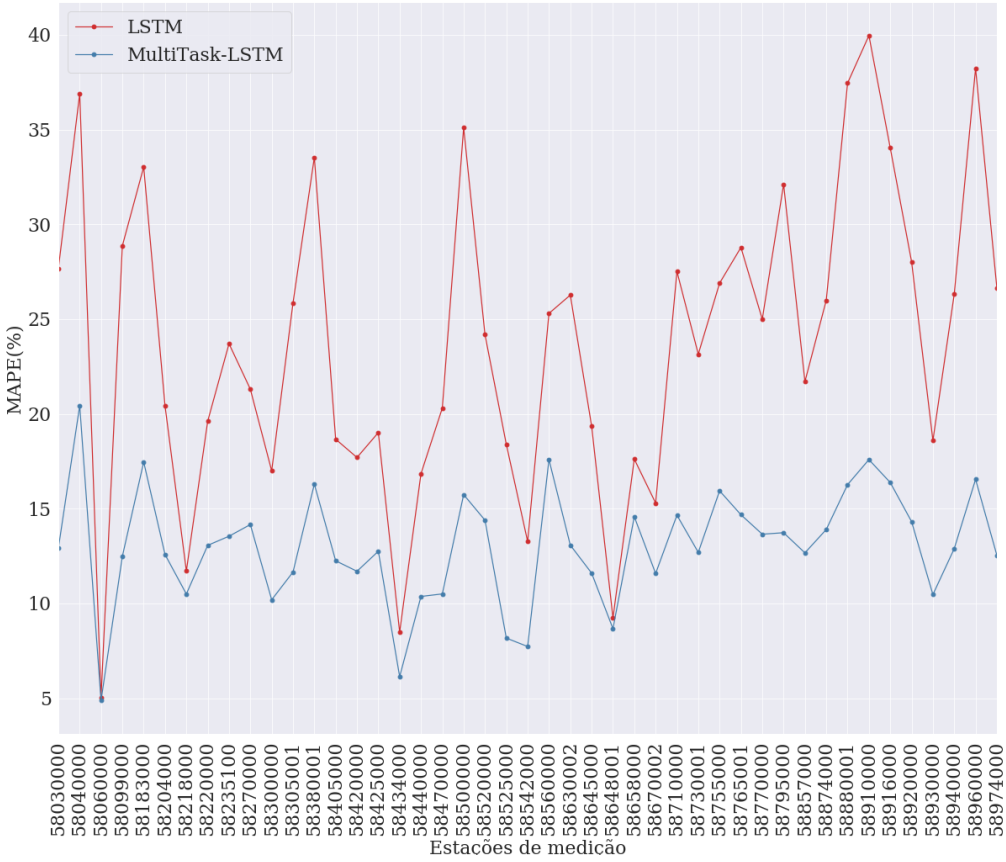


Figura 4.8: Comparação MultiTask-LSTM e LSTM nas 45 estações da bacia do RPS com dados imputados pelo ARIMA

Como pode ser observado na Figura 4.9, o MultiTask-LSTM, quando aplicada a imputação por média dos dias de cada ano, supera em todas as estações o LSTM, como esse experimento manteve fixa a característica sazonal da série temporal ao imputar pela média dos dias de cada ano. Observa-se que o MultiTask-LSTM aprende melhor que o LSTM a sazonalidade e como essa imputação teve maior semelhança a série histórica, podemos apontar uma tendência a ser investigada do desempenho superior do MultiTask-LSTM caso todos os dados estivessem presentes.

Como podemos observar na Figura 4.10, o LSTM foi robusto obtendo MAPEs com pouca variação mesmo com imputações diferentes, o que demonstra que o desempenho do

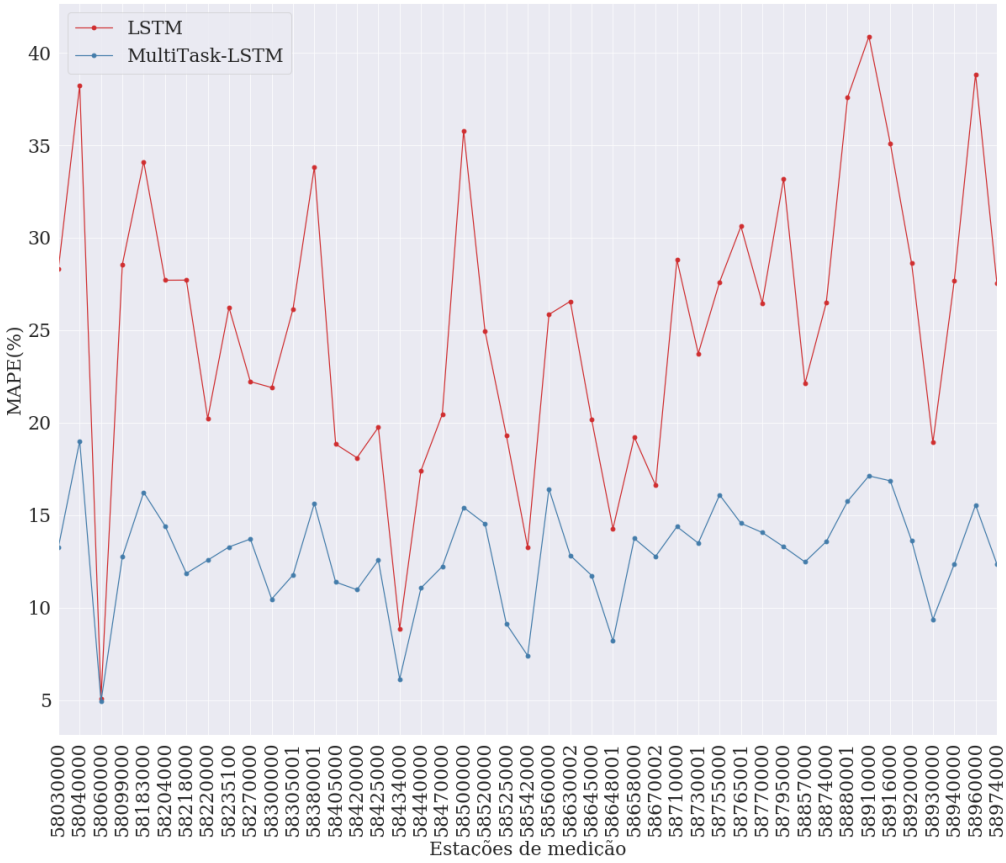


Figura 4.9: Comparação MultiTask-LSTM e LSTM nas 45 estações da bacia do Rio Paraíba do Sul com dados imputados pela médias dos dias de cada ano

modelo não está preso a imputação utilizada e ficou aparente que a imputação das médias dos dias implicou em um efeito negativo para o LSTM.

Como pode ser observado na Figura 4.11, o MultiTask-LSTM também apresentou baixa variabilidade em relação à técnica de imputação. No entanto, ao contrário do LSTM, seu desempenho foi melhorado com a imputação média mês o que reforça a capacidade do modelo com padrões sazonais e conhecimento aprendido de toda a bacia hidrográfica.

Por fim, a Tabela 4.1 resume os resultados encontrados e permite constatar que o *MultiTask-LSTM* alcança MAPEs que são no geral quase duas vezes menores que os alcançados com o LSTM, enquanto o LSTM obteve MAPEs de até 40%, o *MultiTask-LSTM* esteve em até 20.44%.

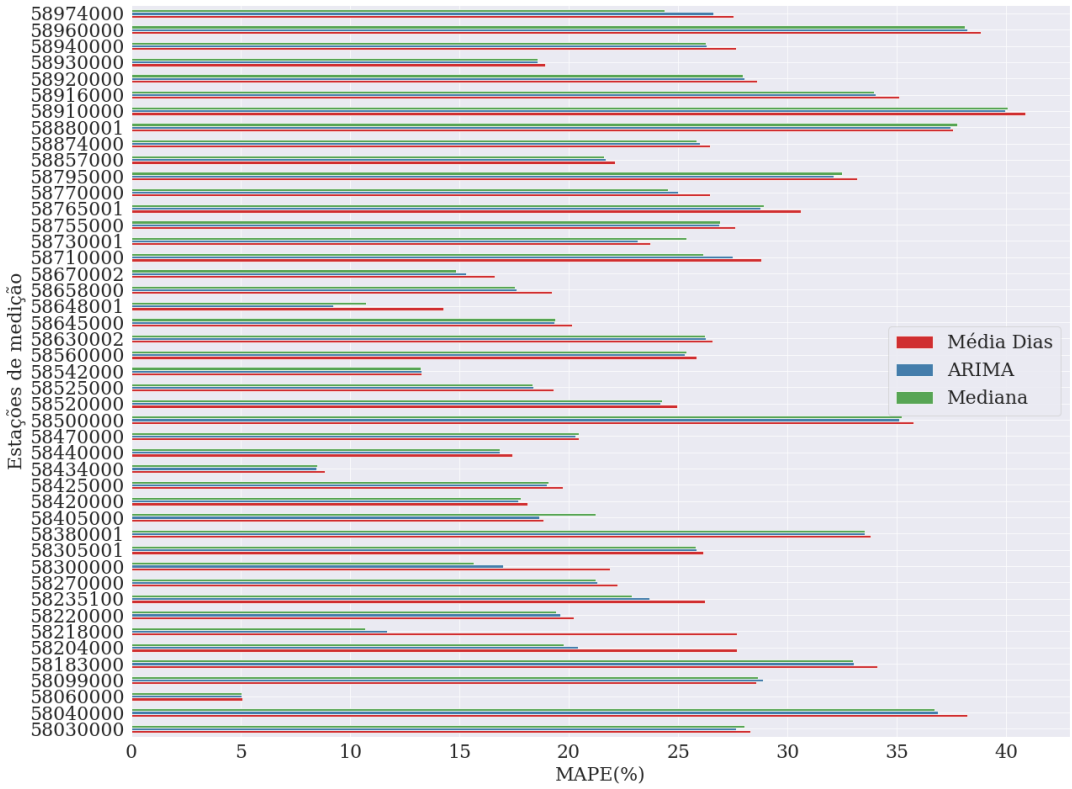


Figura 4.10: MAPEs LSTM nas 45 estações da bacia do Rio Paraíba do Sul variando os métodos de imputação

### 4.2.1 Comparação do tempo de treino dos modelos

Como pode ser observado na figura 4.12, o MultiTask-LSTM possui a grande vantagem de ter seu tempo treino mais rápido uma vez que coloca todas as estações de medição de vazão no mesmo modelo. Por outro lado, o modelo contendo apenas LSTM é consideravelmente mais lento por ter de treinar cada série temporal separadamente.

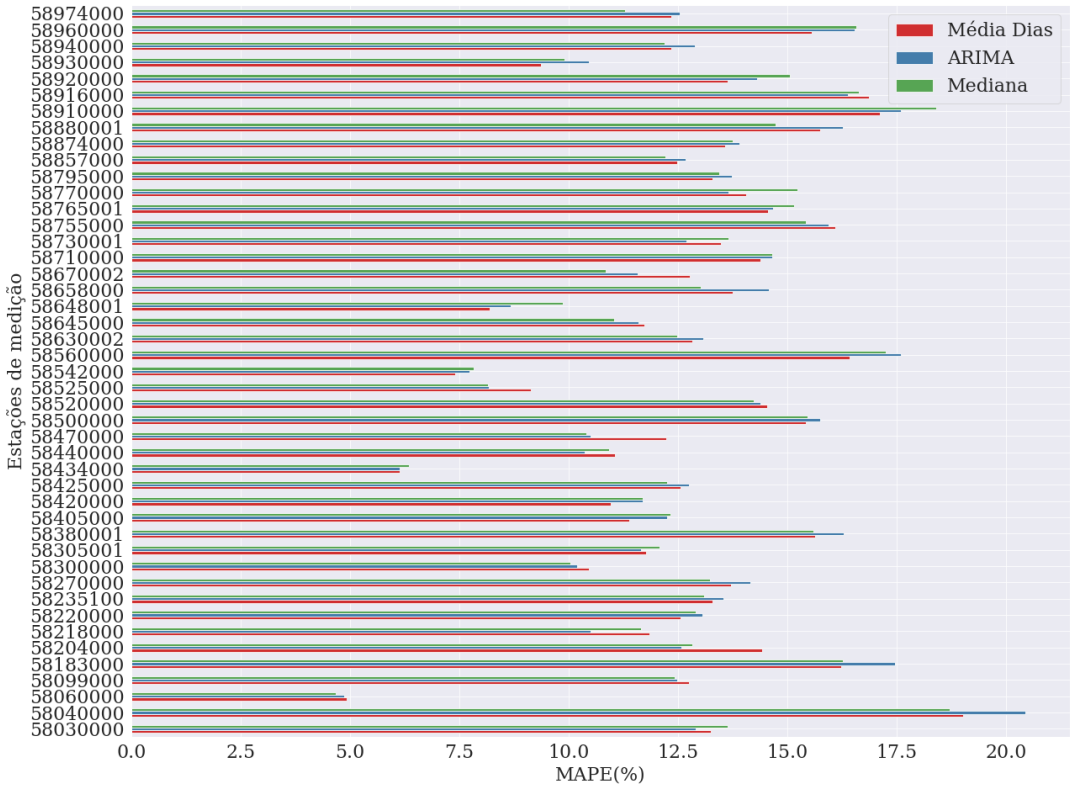


Figura 4.11: MAPEs MultiTask-LSTM nas 45 estações da bacia do Rio Paraíba do Sul variando os métodos de imputação

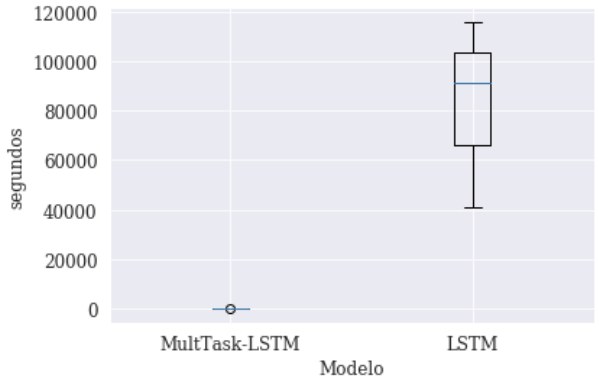


Figura 4.12: Comparação do tempo de execução gasto pelo MultiTask-LSTM e o LSTM em todo o experimento

| | **ARIMA** | | **Média Dias** | | **Mediana** | |
|---|---|---|---|---|---|---|
| **Estações de medição** | LSTM | MultiTask-LSTM | LSTM | MultiTask-LSTM | LSTM | MultiTask-LSTM |
| **58030000** | 27.67 | **12.91** | 28.31 | **13.25** | 28.04 | **13.64** |
| **58040000** | 36.90 | **20.44** | 38.25 | **19.01** | 36.72 | **18.71** |
| **58060000** | 5.03 | **4.87** | 5.08 | **4.92** | 5.04 | **4.67** |
| **58099000** | 28.87 | **12.48** | 28.56 | **12.75** | 28.65 | **12.43** |
| **58183000** | 33.03 | **17.46** | 34.11 | **16.23** | 33.01 | **16.26** |
| **58204000** | 20.42 | **12.58** | 27.70 | **14.41** | 19.77 | **12.83** |
| **58218000** | 11.71 | **10.49** | 27.71 | **11.85** | **10.70** | 11.66 |
| **58220000** | 19.61 | **13.06** | 20.22 | **12.56** | 19.44 | **12.91** |
| **58235100** | 23.70 | **13.54** | 26.23 | **13.28** | 22.89 | **13.09** |
| **58270000** | 21.30 | **14.16** | 22.22 | **13.71** | 21.23 | **13.22** |
| **58300000** | 16.99 | **10.19** | 21.90 | **10.46** | 15.65 | **10.03** |
| **58305001** | 25.85 | **11.65** | 26.15 | **11.76** | 25.80 | **12.08** |
| **58380001** | 33.53 | **16.29** | 33.81 | **15.63** | 33.56 | **15.60** |
| **58405000** | 18.66 | **12.25** | 18.84 | **11.38** | 21.23 | **12.32** |
| **58420000** | 17.70 | **11.69** | 18.10 | **10.97** | 17.80 | **11.70** |
| **58425000** | 19.01 | **12.75** | 19.75 | **12.56** | 19.09 | **12.24** |
| **58434000** | 8.48 | **6.14** | 8.85 | **6.14** | 8.48 | **6.35** |
| **58440000** | 16.84 | **10.37** | 17.41 | **11.06** | 16.86 | **10.91** |
| **58470000** | 20.31 | **10.50** | 20.45 | **12.22** | 20.45 | **10.40** |
| **58500000** | 35.10 | **15.74** | 35.79 | **15.41** | 35.24 | **15.45** |
| **58520000** | 24.18 | **14.39** | 24.95 | **14.53** | 24.26 | **14.23** |
| **58525000** | 18.39 | **8.17** | 19.32 | **9.13** | 18.33 | **8.16** |
| **58542000** | 13.27 | **7.72** | 13.26 | **7.40** | 13.25 | **7.83** |
| **58560000** | 25.30 | **17.58** | 25.85 | **16.42** | 25.38 | **17.25** |
| **58630002** | 26.27 | **13.07** | 26.56 | **12.82** | 26.23 | **12.49** |
| **58645000** | 19.35 | **11.60** | 20.17 | **11.74** | 19.40 | **11.04** |
| **58648001** | 9.23 | **8.67** | 14.27 | **8.20** | 10.73 | **9.86** |
| **58658000** | 17.61 | **14.57** | 19.22 | **13.74** | 17.56 | **13.02** |
| **58670002** | 15.29 | **11.58** | 16.61 | **12.77** | 14.84 | **10.85** |
| **58710000** | 27.51 | **14.65** | 28.81 | **14.38** | 26.16 | **14.64** |
| **58730001** | 23.15 | **12.69** | 23.74 | **13.49** | 25.37 | **13.65** |
| **58755000** | 26.90 | **15.94** | 27.61 | **16.09** | 26.92 | **15.41** |
| **58765001** | 28.78 | **14.68** | 30.62 | **14.56** | 28.93 | **15.14** |
| **58770000** | 24.99 | **13.65** | 26.46 | **14.06** | 24.53 | **15.24** |
| **58795000** | 32.11 | **13.73** | 33.19 | **13.29** | 32.50 | **13.45** |
| **58857000** | 21.70 | **12.67** | 22.11 | **12.47** | 21.62 | **12.21** |
| **58874000** | 25.99 | **13.90** | 26.48 | **13.57** | 25.86 | **13.75** |
| **58880001** | 37.45 | **16.27** | 37.59 | **15.75** | 37.78 | **14.73** |
| **58910000** | 39.96 | **17.59** | 40.88 | **17.12** | 40.07 | **18.41** |
| **58916000** | 34.03 | **16.38** | 35.11 | **16.85** | 33.98 | **16.63** |
| **58920000** | 28.03 | **14.31** | 28.63 | **13.63** | 27.95 | **15.06** |
| **58930000** | 18.59 | **10.46** | 18.93 | **9.36** | 18.58 | **9.91** |
| **58940000** | 26.32 | **12.89** | 27.66 | **12.34** | 26.27 | **12.19** |
| **58960000** | 38.23 | **16.54** | 38.84 | **15.55** | 38.10 | **16.57** |
| **58974000** | 26.63 | **12.53** | 27.54 | **12.35** | 24.38 | **11.28** |

Tabela 4.1: Média dos MAPEs para cada estação de medição de vazão por método de imputação e modelo.

# 5 Conclusão

A previsão de vazão na bacia de rios é um tema caro ao bem-estar e desenvolvimento social, com isso, o estudo de modelos que proporcionem melhoria na previsão da vazão de longo prazo são necessários. Ainda mais em séries temporais que possuem muitos dados faltantes, ruídos e mudanças hidro-geomorfológicos como as séries temporais de vazão.

A aplicação de transferência de aprendizagem *MultiTask Learning*, junto ao modelo de *Deep Learning*, *LSTM*, permite utilizar todos os dados presentes nas séries temporais de uma bacia, o que implica em reaproveitar o conhecimento aprendido em uma série temporal de uma estação de medição de vazão em outras. O modelo foi aplicado na Bacia do Rio Paraíba do Sul e pode ser estendido para outras bacias onde múltiplas estações de medição de vazão estejam coletando dados.

O estudo fez uso de três técnicas de imputação de dados faltantes para averiguar a robustez do *MultiTask-LSTM* a ruídos, onde o modelo foi melhor em todos os cenários de imputação de dados faltantes, obtendo MAPEs consideravelmente melhores quando comparado ao LSTM, que foi o modelo aplicado com sucesso na previsão de longo prazo em estações de medição de vazão do Rio Paraíba do Sul, como pode ser visto na Figura 4.8, Figura 4.7 e Figura 4.9. O *MultiTask-LSTM* ainda apresentou um tempo menor de treino quando comparado ao LSTM como pode-se observar na Figura 4.12.

Portanto, o incremento de transferência de aprendizagem ao LSTM, presente no modelo *MultiTask-LSTM* permitiu a melhora da previsão de longo prazo, com dados de todas as estações de medição de uma bacia hidrográfica e ainda demonstrou robustez à forma de imputação de dados faltantes mantendo uma performance estável com as diferentes imputações.

Sugere-se como trabalhos futuros aplicar o modelo *MultiTask-LSTM* em outras bacias hidrográficas e séries temporais relacionadas entre si, como temperatura, precipitação e energia eólica em uma mesma região.

# Bibliografia

Ratnadip Adhikari and Ramesh K Agrawal. An introductory study on time series modeling and forecasting. *arXiv preprint arXiv:1302.6613*, 2013.

Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time series analysis: forecasting and control*. John Wiley & Sons, 2015.

Luciana Conceição Dias Campos, Leonardo Goliatt da Fonseca, Tales Lima Fonseca, Gabriel Dias de Abreu, Letícia Florentino Pires, and Yulia Gorodetskaya. Short-term streamflow forecasting for paraíba do sul river using deep learning. In *EPIA Conference on Artificial Intelligence*, pages 507–518. Springer, 2019.

Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

Chris Chatfield. *The analysis of time series: an introduction*. Chapman and Hall/CRC, 2016.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Alex Graves. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*, pages 5–13. Springer, 2012.

Simon S Haykin et al. *Neural networks and learning machines/Simon Haykin*. New York: Prentice Hall,, 2009.

Reginald W Herschy. *Streamflow measurement*. CRC Press, 2014.

Sepp Hochreiter and Jürgen Schmidhuber. Lstm can solve hard long time lag problems. In *Advances in neural information processing systems*, pages 473–479, 1997.

J. Kelman. Water supply to the two largest brazilian metropolitan regions. *Aquatic Procedia*, 5:13 – 21, 2015. ISSN 2214-241X. doi: https://doi.org/10.1016/j.aqpro.2015.10.004. URL `http://www.sciencedirect.com/science/article/pii/S2214241X15002825`. At the Confluence Selection from the 2014 World Water Week in Stockholm.

Frederik Kratzert, Daniel Klotz, Claire Brenner, Karsten Schulz, et al. Rainfall-runoff modelling using long short-term memory (lstm) networks. 2018.

Zachary C Lipton, John Berkowitz, and Charles Elkan. A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*, 2015.

Osvaldo Moura Rezende, Marcelo Gomes Miguez, and Aline Pires Veról. Manejo de águas urbanas e sua relação com o desenvolvimento urbano em bases sustentáveis integradas: estudo de caso dos rios pilar-calombé, em duque de caxias/rj. *Revista Brasileira de Recursos Hídricos*, 18(2):149–163, 2013.

Tahasin Shireen, Chenhui Shao, Hui Wang, Jingjing Li, Xi Zhang, and Mingyang Li. Iterative multi-task learning for time-series modeling of solar panel PV outputs. *Applied Energy*, 212(December 2017):654–662, 2018. ISSN 03062619. doi: 10.1016/j.apenergy.2017.12.058. URL https://doi.org/10.1016/j.apenergy.2017.12.058.

Zaher Mundher Yaseen, Ahmed El-shafie, Othman Jaafar, Haitham Abdulmohsin Afan, and Khamis Naba Sayl. Artificial intelligence based models for stream-flow forecasting: 2000-2015, nov 2015. ISSN 00221694. URL https://www.sciencedirect.com/science/article/pii/S0022169415008069.

Eduardo Riomey Yassuda. Gestão de recursos hídricos: fundamentos e aspectos institucionais. *Revista de Administração pública*, 27(2):5–18, 1993.

Jianfeng Zhang, Yan Zhu, Xiaoping Zhang, Ming Ye, and Jinzhong Yang. Developing a long short-term memory (lstm) based model for predicting water table depth in agricultural areas. *Journal of hydrology*, 561:918–929, 2018.